PROCEEDINGS OF THE WORKSHOP ON DISCOVERING MEANING ON THE GO IN LARGE HETEROGENEOUS DATA 2011 (LHD-11)

Held at
The Twenty-second International Joint Conference on Artificial Intelligence
(IJCAI-11)
July 16, 2011
Barcelona, Spain

THE UNIVERSITY of EDINBURGH
informatics

YAHOO! RESEARCH

W3C Semantic Web

This workshop was supported by the Office of Naval Research Global. The content of the information does not necessarily reflect the position or the policy of the United States Government and no official endorsement should be inferred.

Organising Committee

Fiona McNeill (University of Edinburgh)
Harry Halpin (Yahoo! Research)
Michael Chan (University of Edinburgh)

Programme Committee

Marcelo Arenas (Pontificia Universidad Catolica de Chile)
Krisztian Balog (University of Amsterdam)
Paolo Besana (University of Edinburgh)
Roi Blanco (Yahoo! Research)
Paolo Bouquet (University of Trento)
Ulf Brefeld (Yahoo! Research)
Alan Bundy (University of Edinburgh)
Ciro Cattuto (ISI Foundation)
Vinay Chaudhri (SRI)
James Cheney (University of Edinburgh)
Oscar Corcho (Universidad Politécnica de Madrid)
Shady Elbassuoni (Max-Planck-Institut für Informatik)
Jerome Euzenat (INRIA Grenoble Rhone-Alpes)
Eraldo Fernandes (Pontifícia Universidade Católica do Rio de Janeiro)
Aldo Gangemi (CNR)
Pat Hayes (IHMC)
Pascal Hitzler (Wright State University)
Ivan Herman (W3C)
Tom McCutcheon (Dstl)
Shuai Ma (Beihang University)
Ashok Malhotra (Oracle)
Martin Merry (Epimorphics)
Daniel Miranker (University of Texas-Austin)
Adam Pease (Articulate Software)
Valentina Presutti (CNR)
David Roberston (University of Edinburgh)
Juan Sequeda (University of Texas-Austin)
Pavel Shvaiko (Informatica Trentina)
Jamie Taylor (Google)
Eveylne Viegas (Microsoft Research)

Invited Speaker

Fausto Giunchiglia (University of Trento)

Panel Members

John Callahan (ONR Global)
Tom McCutcheon (Dstl)
Peter Mika (Yahoo! Research)
Thomas Steiner (Google)

Programme

16th July 2011

0830-0900	**Registration**
Session 1 (Chair: Fiona McNeill)	
0900-0945	**Short presentation panel: Matching**
	It makes sense... and reference: A bag-of-words model for meaning grounding and sharing in multi-agent systems. Theodosia Togia and Fiona McNeill.
	Towards "On the Go" Matching of Linked Open Data Ontologies. Isabel Cruz, Matteo Palmonari, Federico Caimi and Cosmin Stroe.
	Scale Semantic Matching: Agrovoc vs CABI. Aliaksandr Autayeu.
0945-1015	On the Investigation of Similarity Measures for Product Resolution. Krisztian Balog.
1015-1100	**Short presentation panel: Searching**
	Ranking Query Results from Linked Open Data using a simple cognitive heuristic. Arjon Buikstra, Hansjoerg Neth, Lael Schooler, Annette Ten Teije and Frank Van Harmelen.
	RULIE: Rule Unification for Learning Information Extraction. Dylan Seychell and Alexiei Dingli.
	Understanding Web Data Sources for Search and Exploration. Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Emanuele Della Valle and Silvia Quarteroni.
1100-1130	**Coffee break**

Session 2 (Chair: Michael Chan)	
1130-1200	Towards a Theory of Diagnosis of Faulty Ontologies. Alan Bundy
1200-1230	Enriching Wikipedia Vandalism Taxonomy via Subclass Discovery. Si-Chi Chin and W. Nick Street.
1230-1300	When owl:sameAs isn't the Same Redux: A preliminary theory of identity and inference on the Semantic Web. Harry Halpin, Patrick J. Hayes and Henry S Thompson.
1300-1430	**Lunch**
Session 3 (Chair: Alan Bundy)	
1430-1500	Extending a geo-catalogue with matching capabilities. Feroz Farazi, Vincenzo Maltese, Biswanath Dutta and Alexander Ivanyukovich.
1500-1530	Sense Induction in Folksnonomies. Pierre Andrews, Juan Pane and Ilya Zaihrayeu.
1530-1630	**Invited Talk: Managing diversity in knowledge** Fausto Giunchiglia
1630-1700	**Coffee break**
Session 4 (Chair: Harry Halpin)	
1700-1830	**Panel session: Big Society meets Big Data: Industry and Government Applications of Mapping Meaning** Peter Mika (Yahoo!), Tom McCutcheon (Dstl), John Callahan (ONR Global), Thomas Steiner (Google).
Social event	

Table of Contents

PREFACE

The problem of semantic alignment - that of two systems failing to understand one another when their representations are not identical - occurs in a huge variety of areas: Linked Data, database integration, e-science, multi-agent systems, information retrieval over structured data; anywhere, in fact, where semantics or a shared structure are necessary but centralised control over the schema of the data sources is undesirable or impractical. Yet this is increasingly a critical problem in the world of large scale data, particularly as more and more of this kind of data is available over the Web.

In order to interact successfully in an open and heterogeneous environment, being able to dynamically and adaptively integrate large and heterogeneous data from the Web “on the go” is necessary. This may not be a precise process but a matter of finding a good enough integration to allow interaction to proceed successfully, even if a complete solution is impossible.

Considerable success has already been achieved in the field of ontology matching and merging, but the application of these techniques - often developed for static environments - to the dynamic integration of large-scale data has not been well studied.

Presenting the results of such dynamic integration to both end-users and database administrators - while providing quality assurance and provenance - is not yet a feature of many deployed systems. To make matters more difficult, on the Web there are massive amounts of information available online that could be integrated, but this information is often chaotically organised, stored in a wide variety of data-formats, and difficult to interpret.

This area has been of interest in academia for some time, and is becoming increasingly important in industry and - thanks to open data efforts and other initiatives - to government as well. The aim of this workshop is to bring together practitioners from academia, industry and government who are involved in all aspects of this field: from those developing, curating and using Linked Data, to those focusing on matching and merging techniques.

Extending a geo-catalogue with matching capabilities

Feroz Farazi
DISI
University of Trento
farazi@disi.unitn.it

Vincenzo Maltese
DISI
University of Trento
maltese@disi.unitn.it

Biswanath Dutta
DISI
University of Trento
bisu@disi.unitn.it

Alexander Ivanyukovich
Trient Consulting Group S.r.l.
Trento, Italy
a.ivanyukovich@trientgroup.it

Abstract

To achieve semantic interoperability, geo-spatial applications need to be equipped with tools able to understand user terminology that is typically different from the one enforced by standards. In this paper we summarize our experience in providing a semantic extension to the geo-catalogue of the Autonomous Province of Trento (PAT) in Italy. The semantic extension is based on the adoption of the S-Match semantic matching tool and on the use of a specifically designed faceted ontology codifying domain specific knowledge. We also briefly report our experience in the integration of the ontology with the geo-spatial ontology GeoWordNet.

1 Introduction

To be effective, geo-spatial applications need to provide powerful search capabilities to their users. On this respect, the INSPIRE[1] directive and regulations [EU Parliament, 2009; EU Commission, 2009] establish minimum criteria for the *discovery services* to support search within INSPIRE metadata elements. However, such services are often limited to only syntactically matching user queries to metadata describing geographical resources [Shvaiko *et al*., 2010]. In fact, current geographical standards tend to establish a fixed terminology to be used uniformly across applications thus failing in achieving semantic interoperability. For example, if it is decided that the standard term to denote a harbour (defined in WordNet as "*a sheltered port where ships can take on or discharge cargo*") is *harbour*, they will fail in applications where the same concept is denoted as *seaport.*

As part of the solution, domain specific geo-spatial ontologies need to be adopted. Unfortunately, existing geo-spatial ontologies are limited in coverage and quality [Giunchiglia *et al*., 2010b]. This motivated the creation of GeoWordNet[2] - a multi-lingual geo-spatial ontology providing knowledge about geographic classes, geo-spatial entities (locations), entities' metadata and part-of relations between them. It represents a significant improvement w.r.t. the state of the art, both in terms of quantity and quality of the knowledge provided. As such, it currently constitutes the best candidate to provide semantic support to geo-spatial applications.

One of the purposes of the Semantic Geo-Catalogue (SGC) project [Ivanyukovich *et al*., 2009] - promoted by the PAT - was to extend the geographical catalogue of the PAT with semantic search capabilities. The main requirement was to allow users to submit queries such as *Bodies of water in Trento*, run them on top of the available geographical resources metadata and get results also for more specific features such as *rivers* and *lakes*. This is clearly not possible without semantic support.

In this paper we report our work in providing full support for semantic search to the geo-catalogue of the PAT. This was mainly achieved by integrating in the platform the S-Match[3] semantic matching tool [Giunchiglia *et al*., 2010a] and by adopting a specifically designed faceted ontology [Giunchiglia *et al*., 2009] codifying the necessary domain knowledge about geography and including *inter-alia* the administrative divisions (e.g., municipalities, villages), the bodies of water (e.g., lakes, rivers) and the land formations (e.g., mountains, hills) of the PAT. Before querying the geo-resources, user queries are expanded by S-Match with domain specific terms taken from the faceted ontology. To increase the domain coverage of both resources, we integrated the faceted ontology with GeoWordNet. We conducted an evaluation of the proposed approach to show how simple queries can be semantically expanded using the tool.

The rest of this paper is organized as follows. Section 2 describes the overall system architecture by focusing on the semantic extension. Section 3 describes the dataset containing the locations within the PAT and how we cleaned it. Sections 4, 5 and 6 provide details about the construction of the faceted ontology, its population and integration with GeoWordNet, respectively. The latter step allows supporting multiple languages, enlarging the background ontology and increasing the coverage of locations and corresponding metadata such as latitude and longitude coordinates. Section 7 provides an evaluation showing the effectiveness of the proposed approach. Section 8 provides a generalization of

[1] http://inspire.jrc.ec.europa.eu/

[2] A significant part of GeoWordNet is in RDF and freely available at http://geowordnet.semanticmatching.org/

[3] Freely available at http://sourceforge.net/projects/s-match/

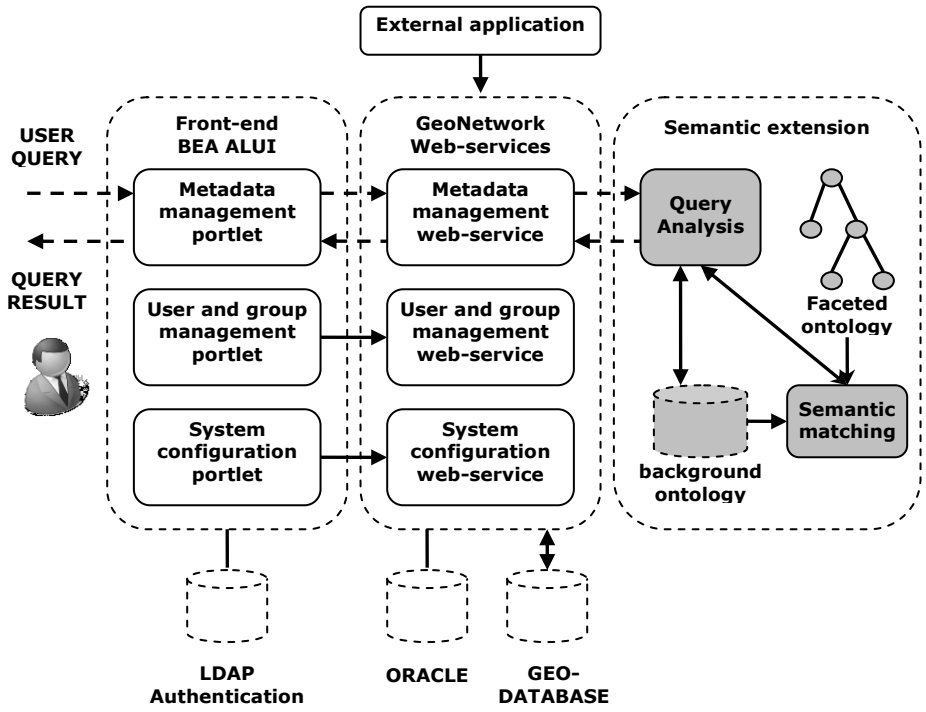


Fig. 1 – The architecture of the semantic geo-catalogue

the work done for the design of the faceted ontology of the PAT in the direction of a faceted ontology for the whole world. Section 9 concludes the paper.

2 The architecture of the geo-catalogue

The overall architecture is constituted by the front-end, business logic and back-end layers as from the standard three-tier paradigm [Shvaiko *et al.*, 2010]. The geo-catalogue is one of the services of the existing geo-cartographic portal[4] of the PAT. It has been implemented by adapting available open-source tool[5] conforming to the INSPIRE directive and taking into account the rules enforced at the national level. Following the best practices for the integration of the third-party software into the BEA ALUI framework[6] (the current engine of the geo-portal), external services are brought together using a portlet[7]-based scheme, where GeoNetwork is used as a back-end. Fig.1 provides an integrated view of the system architecture. At the front-end, the functionalities are realized as three portlets for:

- ***metadata management***, including harvesting, search and catalogue navigation functionalities;
- ***user/group management***, to administer access control on the geo-portal;
- ***system configuration***, which corresponds to the functionalities of the GAST (GeoNetwork's Administrator Survival Tool) tool of GeoNetwork.

These functionalities are mapped *1-to-1* to the back-end services of GeoNetwork. Notice that external applications, can also access the back-end services of GeoNetwork.

The GeoNetwork catalogue search function was extended by providing ***semantic query processing*** support. To provide this support we used the S-Match open source *semantic matching operator*. Given two graph-like structures semantic matching operators identify the pairs of nodes in the two structures that are semantically similar (equivalent, less or more specific), where the notion of semantic similarity is both at the node level and at the structure level [Giunchiglia *et al.*, 2008]. For instance, it can identify that two nodes labeled *stream* and *watercourse* are semantically equivalent because the two terms are synonyms in English. This allows similar information to be identified that would be more difficult to find using traditional information retrieval approaches.

Initially designed as a standalone application, S-Match was integrated with GeoNetwork. As explained in [Shvaiko *et al.*, 2010], this was done through a wrapper that provides web services to be invoked by GeoNetwork. This approach mitigates risks of failure in experimental code while still following strict uptime requirements of the production system. Another advantage of this approach is the possibility to reuse this service in other applications with similar needs.

In order to work properly, S-Match needs domain specific knowledge. Knowledge about the geographical domain is codified into a faceted ontology. A faceted ontology is an ontology composed of several sub-trees, each codifying a different aspect of the given domain. In our case, it includes (among others) the administrative divisions (e.g., municipalities, villages), the bodies of water (e.g., lakes, rivers) and the land formations (e.g., mountains, hills) of the PAT.

[4] http://www.territorio.provincia.tn.it/
[5] GeoNetwork Open Source, http://geonetwork-opensource.org
[6] http://download.oracle.com/docs/cd/E13174_01/alui/
[7] http://jcp.org/en/jsr/detail?id=168

The flow of information, starting from the user query to the query result, is represented with arrows in Fig.1. Once the user enters a natural language query (which can be seen as a classification with a single node), the query analysis component translates it into a formal language according to the knowledge in the background ontology[8]. The formal representation of the query is then given as input to the semantic matching component that matches it against the faceted ontology, thus expanding the query with domain specific terms. The expanded query is then used by the metadata management web-service component to query GeoNetwork and finally access the maps in the database.

3 Data extraction and filtering

The first step towards the construction (Section 4) and population (Section 5) of the faceted ontology was to analyze the data provided by the PAT, extract the main geographical classes and corresponding locations and filter out noisy data. The picture below summarizes the main phases.

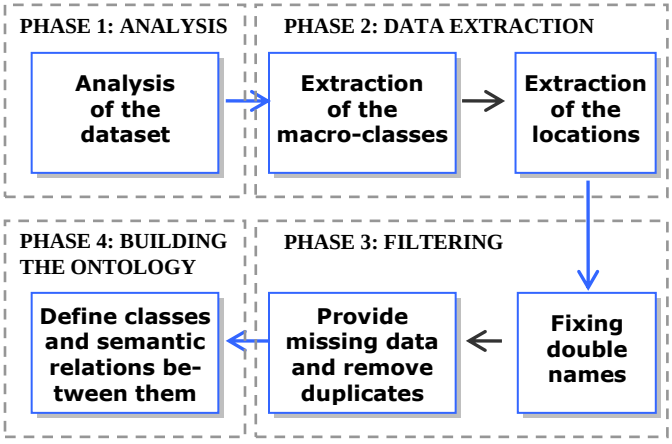


Fig. 2 – The phases for the dataset processing

The dataset of the PAT

The data are available in four files and are gathered from the PAT administration. The *features* file contains the main 45 geographical classes; the *ammcom* file contains 256 municipalities; the *localita* file contains 1,507 wards and ward parts, that we generically call populated places; the *toponimi* file contains 18480 generic locations (including *inter-alia* villages, mountains, lakes, and rivers). *Comune*, *frazione* and *località popolata* are the Italian class names for municipality, ward and populated place respectively.

Data extraction

We retrieved the PAT classes, that we call **macro-classes**, from the *features* file. Each class is associated an id (e.g., P110) and an Italian name (e.g., *Monti principali*). Names of the macro-classes need to be refined as they are too generic and represent many kinds of locations grouped together. As this file lacks classes for the provinces, municipalities, wards and populated places, we manually created them.

We imported all the **locations** into a temporary database by organizing them into the part-of hierarchy *province* > *municipality* > *ward* > *populated place* (and other location kinds). The entity representing the Province of Trento is not explicitly defined in the dataset but it is clearly the root of the hierarchy, so we manually created it. A few locations from the files are not connected to any place and therefore we directly connected them to the province. Each location was temporary assigned to the corresponding macro-class.

Locations are provided with latitude and longitude coordinates in Cartesian WGS84 (World Geodetic System 1984) format, a standard coordinate reference system mainly used in cartography, geodesy and navigation to represent geographical coordinates on the Earth[9]. Since in GeoWordNet we store coordinates in WGS84 decimal format, for compatibility we converted them accordingly.

Filtering

A few location names are double names, e.g., *Cresta di Siusi Cresta de Sousc*. The first (*Cresta di Siusi*) is in Italian and the second (*Cresta de Sousc*) is in Ladin. Ladin is a language spoken in a small part of Trentino and other Alpine regions. The combination of the two is the official name of the location in the PAT. In the temporary database, we put the Italian and Ladin names as alternative names.

While importing the entities in the temporary database, we found that 8 municipalities and 39 wards were missing in the *ammcom* and *localita* files respectively, and 35 municipalities were duplicated in the *ammcom* file. We created the missing locations and eliminated the duplicates. At the end of the importing we identified the objects reported in Table 1.

KIND OF OBJECT	OBJECTS IMPORTED
macro-classes	44
locations	20,162
part-of relations	20,161
alternative names	7,929

Table 1. Objects imported in the temporary database

4 Building the faceted ontology

As mentioned above, the macro-classes provided by the PAT are very generic. This is mainly due to the criteria used by PAT during categorization that were based not only on type but also on importance and population criteria. With the two-fold goal of refining them and determining the missing semantic relations between them, we analyzed the class names and created a multi-lingual faceted ontology.

Our final goal was to create an ontology that both reflects the specificity of the PAT and respects the canons of the analytico-synthetic approach [Ranganathan, 1967] for the

[8] S-Match uses WordNet by default but it is configurable

[9] https://www1.nga.mil/ProductsServices/GeodesyGeophysics/WorldGeodeticSystem/

generation of a faceted ontology. A faceted (lightweight) ontology [Giunchiglia *et al*., 2009] is an ontology divided into sub-trees, called *facets*, each encoding a different dimension or aspect of the domain knowledge. As a result, it can be seen as a collection of lightweight ontologies [Giunchiglia and Zaihrayeu, 2009].

From macro-classes to atomic concepts

We started from the 45 macro-classes, which are not accompanied by any description. Therefore, by analysing the locations contained in the macro-classes, each class was manually disambiguated and refined (split, merged or renamed) and as a result new classes had to be created. This was done through a statistical analysis. Given a macro-class, corresponding locations were searched in GeoWordNet. We looked at all the locations in the part-of hierarchy rooted in the Province of Trento having same name and collected their classes. Only a little portion of the locations were found, but they were used to understand the classes corresponding to each macro-class. The identified classes were manually refined. Some of them required a deeper analysis.

At the end of the process we generated 39 refined classes, including the class *province*, *municipality*, *ward* and *populated place* previously created. Each of these classes is what we call an atomic concept.

Arrange atomic concepts into hierarchies

By identifying semantic relations between atomic concepts and following the analytico-synthetic approach we finally created the faceted ontology of the PAT with five distinct facets: *antiquity*, *geological formation* (further divided into *natural elevation* and *natural depression*), *body of water*, *facility* and *administrative division*. As an example, below we provide the *body of water* facet (in English and Italian).

Body of water (Idrografia)
Lake (Lago)
Group of lakes (Gruppo di laghi)
Stream (Corso d'acqua)
    River (Fiume)
    Rivulet (Torrente)
Spring (Sorgente)
Waterfall (Cascata)
    Cascade (Cascatina)
Canal (Canale)

5 Populating the faceted ontology

Each location in the temporary database was associated a macro-class. The faceted ontology was instead built using the atomic concepts generated from their refinements. In order to populate the faceted ontology, we assigned each location in the temporary database to the corresponding atomic concept by applying some heuristics based on the entity names. As first step, each macro-class was associated to a facet. Macro-classes associated to the same facet constitute what we call a block of classes. For instance, the macro-classes from P110 to P142 (11 classes) correspond to the *natural elevation* block, including *inter-alia* mountains, peaks, passes and glaciers. Facet specific heuristics were applied to each block.

For instance, entities with name starting with *Monte* were considered as instances of the class *montagna* in Italian (*mountain* in English), while entities with name starting with *Passo* were mapped to the class *passo* in Italian (*pass* in English). The general criterion we used is that if we can successfully apply a heuristic then we classify the entity in the corresponding (more specific) class otherwise we select a more generic class, that is the root of a facet (same as the block name) in the worst case. For some macro-classes we reached a success rate of 98%. On average, nearly 50% of the locations were put in a leaf class thanks to the heuristics.

Finally, we applied the heuristics beyond the boundary of the blocks for further refinement of the instantiation of the entities. The idea was to understand whether, by mistake, entities were classified in the wrong macro-class. For instance, in the *natural depression* block (the 5 macro-classes from P320 to P350), 6 entities have name starting with *Monte* and therefore they are supposed to be mountains instead. The right place for them is therefore the *natural elevation* facet. In total we found 48 potentially bad placed entities, which were checked manually. In 41.67% of the cases it revealed that the heuristics were valid, in only 8.33% of the cases the heuristics were invalid and the rest were unknown because of the lack of information available on the web about the entities. We moved those considered valid in the right classes.

6 Integration with GeoWordNet

With the previous step the locations in the temporary database were associated to an atomic concept in the faceted ontology. The next step consisted in integrating the faceted ontology and corresponding locations with GeoWordNet.

Concept integration

This step consisted in mapping atomic concepts from the faceted ontology to GeoWordNet concepts. We automated the disambiguation process with a little amount of manual intervention. Basically, we first manually identified the concept corresponding to the root of each facet - that we call the *facet concept* - and then we restricted the matching of the atomic concepts in the facet to the sub-tree rooted in the facet concept in GeoWordNet. For instance, we restricted the matching of *mountain* to only those concepts more specific than *natural elevation*. If a candidate was found the corresponding concept was selected, otherwise a more general concept, i.e. a suitable parent, was searched. If neither the concept nor the parent was identified, we went for manual intervention.

Entity matching and integration

Two partially overlapped entity repositories, the temporary database built from the PAT dataset (i.e. the populated fac-

eted ontology) and GeoWordNet, were integrated. The PAT dataset overall contains 20,162 locations. GeoWordNet contains nearly 7 million locations from all over the world, including some locations of the PAT. We imported all but the overlapping entities from the temporary database to GeoWordNet. We also automatically generated an Italian and English gloss for each entity. We used several rules, according to the language. In order to detect the duplicates we experimented different approaches. We found that in order to maximize accuracy two entities must match only if they have same name, coordinates, class, parent entities, children entities and alternative names. We allowed a tolerance in matching the coordinates of +/- 0.05, corresponding to +/- 5.5 Km. Note that while matching classes, we took into account the subsumption hierarchy of their concepts. For instance, Trento as *municipality* in the PAT dataset is matched with Trento as *administrative division* in GeoWordNet because the former is more specific than the latter.

7 Evaluation

In order to improve the results associated to a user query, S-Match is used to match terms in the query with the faceted ontology. The background knowledge used for the matching is WordNet, but the tool is developed in such a way to allow substituting it with any other ontology. The matching terms are used to enrich those in the query thus obtaining a semantic query expansion. It is expected that such richer queries, given in input to GeoNetwork, would return a higher number of results. To prove the effectiveness of the approach followed, in Table 2 we provide some examples of queries and the terms in their extension.

Query	Terms identified by S-Match
Watercourse	Rivulet, Stream, River
Falls	Cascade, Waterfall
Elevation	Natural elevation, Mountain, Highland, Glacier, Mountain range, Peak, Hill
Mount	Mountain pass, Mountain, Mountain range
Installation	Milestone, Hut, Farm, Highway, Railway, Road, Street, Transportation system, Provincial Road, Facility, Shelter
Water	Rivulet, Waterfall, Cascade, River, Body of water, Stream, Spring, Canal, Group of lakes, Lake
Transportation facility	Transportation system, Road, Street, Provincial Road, Milestone, Railway, Highway
Reef	

Table 2. Some query expansion results

The last example shows how typing the query *reef* would not produce any result. This depends on the fact that the faceted ontology strictly codifies the local specificities of the Province of Trento that does not present any marine environment (it is a mountain region far from the sea). In all the other cases it is evident how S-Match identifies semantically related terms (synonyms, less or more specific terms).

Table 3 shows real results in terms of documents found. The portal actually interacts with the users in Italian. The Italian query is translated in English, matched with the faceted ontology and, once processed, results are given back in Italian. Only terms matching with at least one document are returned. For instance, the query for *tracking* returns only *pista* (track) and *ponte* (bridge), since no documents are available in the repository for the term *tracking*.

Query	Expansion (with number of documents)
foresta	foresta (119), bosco (14)
fiume	fiume (18), alveo (16)
lago	lago (4), laghi (20)
strada	strada (14), strada provinciale (5)
connessione	connessione (3), ponte (6)
paese	località (15), provincia (348), città (4), comune (952), frazione (2), centri abitati (16)
tracking	pista (5), ponte (6)

Table 3. Some query expansion results

Note that by populating the ontology with locations and taking into account the part-of relations between them, also location names can be expanded. For instance, by providing the information that *Povo* is an administrative division in *Trento* it is possible to expand the term *Trento* with *Povo*. However, providing this support was out of the scope of the SGC project.

8 Extending the faceted ontology

The work done with the PAT for the construction of the faceted ontology can be generalized to cover the whole world. We recently worked on a methodology - mainly inspired by the faceted approach - and a minimal set of guiding principles aimed at modeling the spatial domain (and in general any domain) and at building the corresponding background knowledge taking into account the *classes*, the *entities*, their *attributes* and *relations* [Dutta *et al*., 2011]. We consider classes, relations, and attributes as the three fundamental components, or categories, of any domain. In this approach, the analysis of the domain allows the identification of the basic classes of real world objects. They are arranged, per *genus et differentia* (i.e. by looking at their commonalities and their differences), to construct the *facets*, each of them codifying a different aspect of the domain at hand. This allows being much more rigorous in the definition of the domain and its parts, in its maintenance and use [Giunchiglia *et al*., 2009].

We selected the classes from GeoWordNet and arranged them into 8 facets, each of them further divided into sub-facets: region, administrative division, populated place, facility, abandoned facility, land, landform and body of water.

The spatial relations we propose extend those in [Pullar and Egenhofer, 1988]. In addition to the standard direction, topological, ordinal, distance and fuzzy relations, we extend them by including relative level (e.g. above, below), longitudinal (e.g. in front, behind), side-wise (e.g. right, left), position in relation to border or frontier (e.g. adjacent, overlap) and other similar relations. We also consider functional relations. For example, in the context of lakes, primary inflow and primary outflow are two important relations.

An attribute is an abstraction belonging to or a characteristic of an object. This is a construct through which objects or individuals can be distinguished. Attributes are primarily *qualitative* and *quantitative* in nature. For example, we may mention depth (of a river), surface area (of a lake), length (of a highway) and altitude (of a hill). For each of these attributes, we may have both qualitative and quantitative values. We store the possible qualitative values in the background knowledge. This provides a controlled vocabulary for them. They are mostly *adjectives*. For example, for depth (of a river) the possible values are {wide, narrow}. Similarly, for altitude (of a hill) the possible values are {high, low}. We also make use of *descriptive* attributes. They are used to describe, usually with a short natural language sentence, a specific aspect of an entity. Typical examples are the history (of a monument) or the architectural style (of a building) or any user defined tag.

Our space domain overall includes 845 classes, 70 relations and 35 attributes. In comparing it with existing geo-spatial ontologies, like GeoNames and TGN, our space domain is much richer in all its aspects. Further details can be found in [Dutta *et al*., 2011].

9 Conclusions

We briefly reported our experience in providing a semantic extension to the geo-catalogue of the PAT. S-Match, once integrated with GeoNetwork, performs a semantic expansion of the query using a faceted ontology codifying the domain knowledge about geography of the PAT. This allows identifying information that would be more difficult to find using traditional information retrieval approaches.

To mutually increase their coverage, we have also integrated the faceted ontology with GeoWordNet. At this purpose we had to match their concepts and entities. The matching of the concepts was done by focusing on one facet at a time. The entity matching criteria needed to be tuned to maximize accuracy. We also briefly reported the methodology that we use to build domains and how we applied it to the space domain on top of GeoWordNet.

Acknowledgments

This work has been supported by the TasLab network project funded by the European Social Fund under the act n° 1637 (30.06.2008) of the PAT, by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 231126 LivingKnowledge: Living-Knowledge - Facts, Opinions and Bias in Time and by "Live Memories - Active Digital Memories of Collective Life" funded by the PAT. We are thankful to Pavel Shvaiko, Aliaksandr Autayeu, Veronica Rizzi, Daniela Ferrari, Giuliana Ucelli, Monica Laudadio, Lydia Foess and Lorenzo Vaccari for their kind support.

References

[Dutta *et al*., 2011] B. Dutta, F. Giunchiglia, V. Maltese. A facet-based methodology for geo-spatial modelling. In *Proceedings of the GEOS, Vol. 6631*, 2011.

[EU Commission, 2009] European Commission. COMMISSION REGULATION (EC) No 976/2009 implementing Directive 2007/2/EC as regards the Network Services, 2009.

[EU Parliament, 2009] European Parliament. Directive 2007/2/EC establishing an Infrastructure for Spatial Information in the European Community (INSPIRE), 2009.

[Giunchiglia *et al*., 2008] F. Giunchiglia, F. McNeill, M. Yatskevich, J. Pane, P. Besana, and P. Shvaiko. Approximate structure-preserving semantic matching. In *Proceedings of ODBASE*, 2008.

[Giunchiglia and Zaihrayeu, 2009] F. Giunchiglia, I. Zaihrayeu. Lightweight Ontologies. *The Encyclopedia of Database Systems*, 2007

[Giunchiglia *et al*., 2009] F. Giunchiglia, B. Dutta, V. Maltese. Faceted Lightweight Ontologies. In "Conceptual Modeling: Foundations and Applications", A. Borgida, V. Chaudhri, P. Giorgini, Eric Yu (Eds.) LNCS 5600 Springer, 2009.

[Giunchiglia *et al*., 2010a] F. Giunchiglia, A. Autayeu, J. Pane. S-Match: an open source framework for matching lightweight ontologies. *The Semantic Web journal*, 2010.

[Giunchiglia *et al*., 2010b] F. Giunchiglia, V. Maltese, F. Farazi, B. Dutta. GeoWordNet: a resource for geo-spatial applications. In *Proceedings of ESWC*, 2010.

[Ivanyukovich *et al*., 2009] A. Ivanyukovich, F. Giunchiglia, V. Rizzi, V. Maltese. SGC: Architettura del sistema. *TCG/INFOTN/2009/3/D0002R5 report*, 2009.

[Pullar and Egenhofer, 1988] D. Pullar, M. J. Egenhofer. Toward formal definitions of topological relations among spatial objects. In *Proceedings of the 3rd International Symposium on Spatial Data Handling, Sydney, Australia, pp. 165–176*, 1988.

[Ranganathan, 1967] S. R. Ranganathan. Prolegomena to library classification. *Asia Publishing House*, 1967.

[Shvaiko *et al*., 2010] P. Shvaiko, A. Ivanyukovich, L. Vaccari, V. Maltese, F. Farazi. A semantic geo-catalogue implementation for a regional SDI. In *Proceedings of the INPSIRE Conference*, 2010.

Sense Induction in Folksonomies *

Pierre Andrews and **Juan Pane** and **Ilya Zaihrayeu**
DISI - University of Trento, Italy
{andrews,pane,ilya}@disi.unitn.it

Abstract

Folksonomies, often known as tagging systems, such as the ones used on the popular Delicious or flickr websites, use a very simple knowledge organisation system. Users are thus quick to adopt this system and create extensive knowledge annotations on the Web. However, because of the simplicity of the folksonomy model, the semantics of the tags used is not explicit and can only be inferred from the context of use of the tags. This is a barrier for the automatic use of such knowledge organisation systems by computers and new techniques have to be developed to extract the semantic of the tags used. In this paper we discuss an algorithm to detect new senses of terms in a folksonomy; we also propose a formal evaluation methodology that will enable to compare results between different approaches in the field.

1 Introduction

Folksonomies are uncontrolled knowledge organisation systems where users can use free-text tags to annotate resources. They create a network of user-tag-resource triplets that encodes the knowledge of users [Mika, 2007]. 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) [Golder and Huberman, 2006]. This lack of explicit semantics makes it difficult for computer algorithms to leverage the whole knowledge provided by the folksonomy model.

While there are existing Word Sense Disambiguation (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 [Andrews *et al.*, 2011], a standard a structured vocabulary such as WordNet [Miller, 1998] covers less that 50% of the terms used by the users of the folksonomy. This happens because of the dynamic nature of folksonomies where new concepts and terms appear quickly. To tackle this issue, *sense induction* algorithm are being developed [García-Silva *et al.*, 2010] to detect new concepts and extend the existing vocabularies.

While the computer does not "know" the actual meaning of the free-text tag used, the users always know the meaning they wanted to use when they tagged a resource. So if they tagged a bookmark with "java", in their mind, at the time of tagging, they knew exactly if they meant the "indonesian island" or the "programming language". This principle has already been widely illustrated in the automatic ontology building field where social network analysis methods were introduced [García-Silva *et al.*, 2010] to extract the so-called "emergent semantics" [Aberer *et al.*, 2004].

In this article, we discuss our approach to the detection of new concepts in folksonomies and how it differs from the state of the art by enabling the detection of homographs (see Sections 2 and 3). Because the state of the art also lacks a formalised evaluation methodology, we discuss in Section 4 a possible approach for a comparable and reproducible evaluation. While we are currently applying this evaluation methodology to the algorithm we introduce, we are not reporting results in this paper as they are not yet available.

2 Sense Induction

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 [Xu and Wunsch, 2005]. 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 [García-Silva *et al.*, 2010].

*This work has been partially supported by INSEMTIVES project (FP7-231181, see http://www.insemtives.eu).

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, [Zhang *et al.*, 2006] show that tags are often ambiguous in folksonomies (their study is also based on Delicious[1]) 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 (see Section 3).

3 Algorithms

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 [García-Silva *et al.*, 2010]). 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 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. [García-Silva *et al.*, 2010]). 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 [Ester *et al.*, 1996] 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: ϵ, the minimum distance between two items to put them in the same cluster and m the minimum number of items in a cluster.

  ϵ 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 as discussed in Section 4.

- 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 [Xu and Wunsch, 2005].

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 [Alfonseca and Manandhar, 2002a] or [Jamoussi, 2009]), 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, [Alfonseca and Manandhar, 2002a] 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 [Jamoussi, 2009] 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 [Lin, 1998]. The authors of [Weinberger *et al.*, 2008] 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.

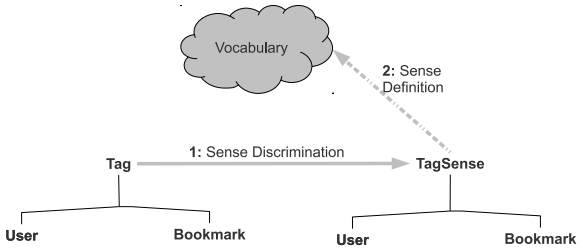


Figure 1: Sense-User-Bookmark Tripartite graph

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 [Au *et al.*, 2007], we discover the different meanings of the tag. By considering each cluster to be a tag

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

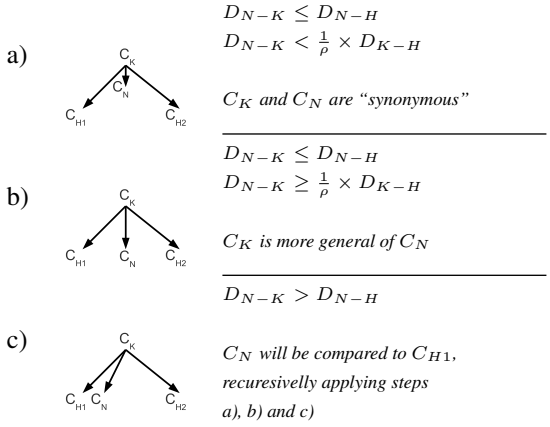


Figure 2: Decisions to Extend the Concept Taxonomy

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 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 [García-Silva *et al.*, 2010] 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 by applying a hierarchical classification approach similar to the one proposed in [Alfonseca and Manandhar, 2002a]. 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 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 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 2a)[2]);
   - 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 its hyponyms, then it is most likely that C_n is more specific than C_k (Figure 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 2c));

   We apply this search procedure on our structured vocabulary (in our case 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.

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

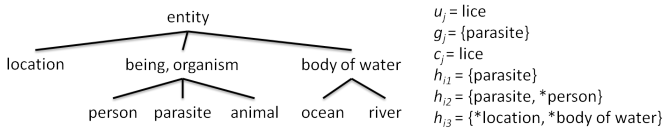


Figure 3: Example of taxonomy, an unknown relevant concept u_j, its correct generalisations g_j and the generalisations proposed by three hypothetical algorithms h_{ik}

We are currently running evaluations to show the behaviour of these algorithms with different values of the ϵ, m and ρ parameters and will report on these in future publications.

4 Evaluation Methodology

While there is existing research on the automatic construction of ontologies from folksonomies, [García-Silva *et al.*, 2010] points out that there is not yet any agreed evaluation dataset. In fact, from our knowledge of the state of the art approaches, there is not yet an appropriate evaluation methodology in the field. This is mostly due to the lack of a gold standard evaluation dataset and thus the evaluation of the existing methods were often only evaluated "*subjectively*" [Lin *et al.*, 2009] by checking manually some extracted clusters, thus only providing anachronyc results that cannot be compared or reproduced [García-Silva *et al.*, 2009; Van Damme *et al.*, 2007; Specia and Motta, 2007].

However, as pointed out earlier, the NLP field has already tackled the issue of concepts extraction from text and has considered different evaluation measures for this task. [Alfonseca and Manandhar, 2002b] 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 3, adapted from [Alfonseca and Manandhar, 2002b])

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 [Hahn and Schnattinger, 1998]).

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). Alfoncesca and Manandhar [Alfonseca and Manandhar, 2002b] 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 [Andrews *et al.*, 2011] (the tags2con dataset) as it provides a manually validated disambiguation for each tag in a subset of the Delicious folksonomy. The tags2con dataset is a collection of bookmarks from the Delicious website for which each free-text tag associated to the bookmarks has been manually disambiguated to its corresponding concept in a structured vocabulary, in this case WordNet.

The measures listed above can be computed on this dataset by applying a leave one out 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 ϵ and m parameters of the clustering steps, 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 metrics used and compare different approaches. For this, we need a clustering gold standard, that provides the "true cluster" (class) of each user-resource 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.

When building the gold standard (GS^j) we want to automatically generate the set of unknown concepts (U and C_i) to

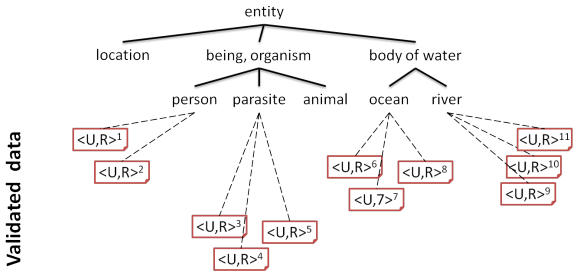


Figure 4: Validated Data

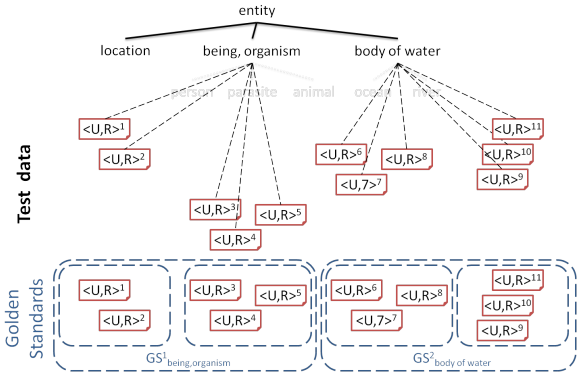


Figure 5: One Possible Test Set

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 Structured Vocabulary (SV) for which there is more than one hyponym that has more than one manually validated associated term in the annotation. In the example in Figure 4, 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 two hyponyms ("ocean" and "river") that have more than one annotation attached to it. Each of

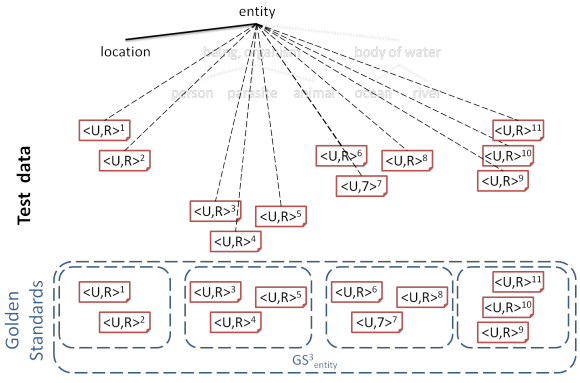


Figure 6: Another Possible Test Set Generated from a Higher Concept in the CV

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 1: Resulting gold standards GS^k for the evaluation of the sense induction algorithm.

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 5, 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 the higher we go in the hierarchy, the more classes will be created in GS^k. In our example in Figure 6, we further "forget" the concepts "being, organism" and "body of water" and create another gold standard GS^3 for "entity", creating four clusters.

If we apply the above mentioned process on the dataset depicted in Figure 4 we obtain three GS datasets as shown in Table 1. When using the tags2con dataset [Andrews *et al.*, 2011], we have 4 427 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. We are currently running the evaluation of different distances on this gold standard.

The purpose of each gold standard GS^k is twofold:

1. Evaluate Step one of the sense induction algorithm presented in the previous Section 3, where the input is a set of free-text tags C and the output is a set of clusters of similar tags that represent a new concept. In our example in Figure 4, we would be calling the clustering algorithm with each of the gold standard 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 [Amigó *et al.*, 2009].
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 SV as more specific than an existing concept in the SV. The generalisations in the SV discovered by the algorithm (H_i) is be compared to the one given in the gold standard (G_j). In our example in Figure 4, if we pass GS^1 to the algorithm, it should create concepts for "person" and "parasite", and put them as hyponyms of "being, organism".

5 Results

Using the methodology described in the previous section, we have run a set of preliminary evaluation for the first step of the algorithm using different clustering distances found in the state of the art.

To compute the minimum baseline, we perform random runs where a random number of clusters between one and the number of instances is selected and each instance is assigned randomly to one of these cluster. The mean F-measure on one thousand runs is of 25.8%[3].

In the state of the art, the number of collocated tags between bookmarks, and the number of users using a tag are most often used to compare bookmarks or tags. We have thus started by evaluating these distance measures to establish the state of the art baseline. When using only tag collocation, the first step clustering algorithm can achieve a maximum F-measure of 59.7%[4]. The user collocation measure achieves a very similar result with a maximum F-measure of 59.1%[5], we can however see that the distribution between precision and recall of these two approaches is quite different.

We are currently running evaluations for the other steps of the algorithm and with more complex distance measures that should improve on the naive tag collocation approaches.

6 Conclusion and Future work

We have presented a novel approach to detect concepts in a folksonomy. Our approach is an extension to the state of the art that adds a method to detect polysemous/homograph tags and assign them to different senses. Because of the – acknowledged – lack of standard evaluation methodology in the state of the art, we also propose a new methodology for evaluating sense induction in a folksonomy and building datasets to run such evaluation.

[3] SD = 27.9%; Precision=42.2%, Recall=29.2%.

[4] Precision=59.4%, Recall=63.4%.

[5] Precision=64.8%, Recall=40.1%.

We are currently running evaluations of different distance metrics and parameters to our algorithms by applying the proposed evaluation methodology described here and will report on results of this new approach in upcoming publications[6].

References

[Aberer *et al.*, 2004] Karl Aberer, Philippe C. Mauroux, Aris M. Ouksel, Tiziana Catarci, Mohand S. Hacid, Arantza Illarramendi, Vipul Kashyap, Massimo Mecella, Eduardo Mena, Erich J. Neuhold, and Et. Emergent Semantics Principles and Issues. In *Database Systems for Advances Applications (DASFAA 2004), Proceedings*, Lecture Notes in Computer Science, pages 25–38. Springer, March 2004.

[Alfonseca and Manandhar, 2002a] Enrique Alfonseca and Suresh Manandhar. Extending a lexical ontology by a combination of distributional semantics signatures. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, EKAW '02, pages 1–7, London, UK, 2002. Springer-Verlag.

[Alfonseca and Manandhar, 2002b] Enrique Alfonseca and Suresh Manandhar. Proposal for evaluating ontology refinement methods. *Language Resources and Evaluation*, 2002.

[Amigó *et al.*, 2009] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12:461–486, 2009. 10.1007/s10791-008-9066-8.

[Andrews *et al.*, 2011] Pierre Andrews, Juan Pane, and Ilya Zaihrayeu. Semantic disambiguation in folksonomy: a case study. In *Advanced Language Technologies for Digital Libraries*, Lecture Notes on Computer Science (LNCS) Hot Topic subline. to Springer-Verlag, 2011.

[Au *et al.*, 2007] Au, N. S. Gibbins, and N. Hadbolt. Understanding the Semantics of Ambiguous Tags in Folksonomies. In *The International Workshop on Emergent Semantics and Ontology Evolution (ESOE2007) at ISWC/ASWC 2007*, November 2007.

[Ester *et al.*, 1996] Martin Ester, Hans peter Kriegel, Jrg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. AAAI Press, 1996.

[García-Silva *et al.*, 2009] A. García-Silva, M. Szomszor, H. Alani, and O. Corcho. Preliminary results in tag disambiguation using dbpedia. In *Proc. of KCAP*, USA, 2009.

[García-Silva *et al.*, 2010] 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.

[6] while the results are not ready at the time of submitting this paper for review, they will be available at the time of the workshop.

[Golder and Huberman, 2006] Scott Golder and Bernardo A. Huberman. The structure of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, April 2006.

[Hahn and Schnattinger, 1998] Udo Hahn and Klemens Schnattinger. Towards text knowledge engineering. In *IN AAAI/IAAI*, pages 524–531, 1998.

[Jamoussi, 2009] Salma Jamoussi. Une nouvelle reprsentation vectorielle pour la classification smantique. In *Apprentissage automatique por le TAL*, volume 50, pages 23–57. TAL, 2009.

[Lin *et al.*, 2009] Huairen Lin, Joseph Davis, and Ying Zhou. An integrated approach to extracting ontological structures from folksonomies. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvnen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, editors, *The Semantic Web: Research and Applications*, volume 5554 of *Lecture Notes in Computer Science*, pages 654–668. Springer, Berlin / Heidelberg, 2009.

[Lin, 1998] Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 768–774, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[Mika, 2007] Peter Mika. Ontologies are us: A unified model of social networks and semantics. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):5–15, March 2007.

[Miller, 1998] G. Miller. *WordNet: An electronic Lexical Database*. MIT Press, 1998.

[Specia and Motta, 2007] L. Specia and E. Motta. Integrating folksonomies with the semantic web. In *Proc. of the European Semantic Web Conference (ESWC2007)*, volume 4519 of *LNCS*, pages 624–639, Berlin Heidelberg, Germany, July 2007. Springer-Verlag.

[Van Damme *et al.*, 2007] Caline Van Damme, Martin Hepp, and Katharina Siorpaes. Folksontology: An integrated approach for turning folksonomies into ontologies. In *Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007)*, pages 57–70, 2007.

[Weinberger *et al.*, 2008] Kilian Quirin Weinberger, Malcolm Slaney, and Roelof Van Zwol. Resolving tag ambiguity. In *Proceeding of the 16th ACM international conference on Multimedia*, MM '08, pages 111–120, New York, NY, USA, 2008. ACM.

[Xu and Wunsch, 2005] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks / a Publication of the IEEE Neural Networks Council*, 16(3):645–678, May 2005. PMID: 15940994.

[Zhang *et al.*, 2006] Lei Zhang, Xian Wu, and Yong Yu. Emergent Semantics from Folksonomies: A Quantitative Study. In *Journal on Data Semantics VI*, Lecture Notes in Computer Science, chapter 8, pages 168–186. Springer, 2006.

Towards a Theory of Diagnosis of Faulty Ontologies *

Alan Bundy
School of Informatics, University of Edinburgh
Edinburgh, Scotland, UK
A.Bundy@ed.ac.uk

Abstract

We initiate research into a generic theory of diagnosis of faulty ontologies. The proposals are based on, but generalise, our experience with the GALILEO and ORS systems. We make some initial simplifying assumptions, which we hope will not restrict the application of the proposals to new areas. In particular, we look at repairing faulty ontologies where the fault is revealed by an inference failure and the repair is implemented by a signature and/or theory morphism. More concretely, we focus on situations where a false conjecture has been proved. Diagnosis consists of constructing a morphism by analysis of failed inference. It is assumed that an oracle is available that can answer (and sometimes ask) questions, but whose own ontology is otherwise inscrutable.

1 Introduction

This paper builds on our experience of two systems for ontology evolution: ORS and GALILEO. Both systems start with a potentially faulty logical theory representing some knowledge. We will use the word *ontology* to describe such logical theories. As a result of inference failure, these ontologies are shown to be faulty. The fault is then automatically diagnosed and the faulty ontology is then repaired. We will try to generalise from the experience of these two systems to construct a generic theory of diagnosis of faulty ontologies.

ORS (Ontology Repair System) evolves planning ontologies [McNeill and Bundy, 2007]. In its domain a planning agent (PA) constructs plans from the services provided by service providing agents (SPAs). The PA represents these services with STRIPS-like operators. If the plan fails on execution then this indicates that the PA's representation of the SPA's services is faulty. ORS diagnoses the fault and repairs the PA's ontology. Repairs can be either to the theory or to the language. An example of a theory repair is inserting a missing precondition in a STRIPS operator e.g., an SPA issuing travel visas might require the PA to possess some additional documentation that it wasn't initially aware that it needed. An example of a language repair is adding an additional argument to a predicate, e.g., refining a proposition asserting that the PA must provide a photograph by specifying its size with the additional argument. The PA then replans with the repaired ontology. Several repairs may be required to form a plan that will execute successfully.

GALILEO (Guided Analysis of Logical Inconsistencies Leads to Evolved Ontologies) evolves ontologies representing physics theories [Bundy and Chan, 2008]. In its domain the predicted value of a physical function may be inferred to differ from its observed value. GALILEO represents the world with multiple ontologies, e.g., one for the theory and another for an experiment. These ontologies are locally consistent, but can be globally inconsistent. Different patterns of divergence and repair are represented in *ontology repair plans* (ORP)s. If an ORP's trigger pattern is matched then an appropriate repair is executed. Repairs include: splitting one concept into many, merging several concepts into one, adding a dependence on a hidden variable, or making a concept independent of a variable. For instance, an anomaly in the orbital velocity of spiral galaxies might suggest splitting the concept of matter into regular visible matter, dark matter and total matter.

An *ontology* is a pair $\langle \Sigma, A \rangle$, where:

- Σ is the ontology's *signature*, which defines its language. It consists of a set of type declarations for the concepts in the ontology.
- A is the ontology's *axioms*, which define its theory. It consists of a set of formulae asserted to be true.

The type declarations and formulae are expressed in a logic $\mathcal{L}$. We will represent the repair of a source ontology O as the target ontology $\nu(O)$, defined as:

$$\nu(O) \quad ::= \quad \langle \nu_\sigma(\Sigma), \nu_\alpha(A) \rangle$$

where ν_σ is a signature morphism and ν_α is a theory morphism. These morphisms are meta-level functions that describe how to repair the source signature/axioms to form the target ones.

The logic of ORS is KIF, an extended, typed, classical first-order logic. Its signature consists of type declarations for its

*Thanks to Liwei Deng and an anonymous referee for feedback on an earlier draft and to my research group for feedback during a seminar.

predicates and constants. Its axioms are (a) the STRIPS operators and (b) propositions describing its beliefs about its environment.

The logic of GALILEO is simply-typed lambda calculus, a form of higher-order logic. Its ontology's signature's consists of type declarations of functions describing the physical world and its attributes. Note that higher-order functions are required to represent force fields, calculus operations, astronomical orbits, etc. Its ontology's axioms include mathematical theorems, physical laws, experimental observations, etc.

2 Types of Ontological Fault

ORS and GALILEO have the following common operations:

- identification of a problem with an ontology;
- diagnosis of this problem;
- repair of the problem.

Although each system incorporates mechanisms for diagnosis and repair, these are domain specific. We lack a theory to underpin them. In particular, we lack *a theory of diagnosis*. In this paper we will initiate the development of such a diagnostic theory.

To aid formalisation, we will make the following simplifying assumptions.

1. *An attempt to prove a conjecture can one of the following outcomes:*
   - The conjecture may be *proved*;
   - The search space may be exhausted without a proof being found, i.e., the conjecture is *unprovable*; or
   - Resource limits may be encountered before a proof is found or the search space is exhausted, i.e., the conjecture is *undetermined*.
2. *By a* fault *in an ontology we will mean some kind of reasoning failure.* Examples of such failures include:
   (a) A conjecture is true but unprovable.
   (b) A conjecture is false but *is* proved.
   (c) Inference is too inefficient, e.g., the search space is infeasibly large. The evidence may be that too many conjectures are left undetermined.
3. *By an ontology* repair *we will mean the application of a morphism (signature, theory or some combination) to the source ontology to produce a new target one.* The fault being repaired should not hold in the target ontology.
4. *By a fault* diagnosis *we will mean constructing a morphism that can be used to repair the fault.*
5. *The* diagnostic process *is a procedure that, given a faulty source ontology, can produce a diagnosis. For both type 2a and type 2b faults the diagnosis process can be represented as the analysis of a failed or successful proof attempt.*
6. *Proof attempts can be represented as a search space containing a partial or complete proof. Without loss of generality, we can represent the search space as an* OR*-tree and a proof as an* AND*-tree.*

3 Justification of these Simplifying Assumptions

In both GALILEO and ORS, ontology evolution is driven by inference failure, in particular, type 2b inference failure. The derivation of false theorems can arise in two ways: (i) because the ontology is inconsistent, so all formulae are provable; (ii) because a particular theorem is false in the preferred model. Below we will focus on type 2b (ii) inference failure.

- In ORS, a plan is derived that then fails on execution. The derivation of the plan is also a formal verification that the plan will achieve the goals of the PA. The failure of the plan shows that the plan will *not* achieve these goals, so the verification has proved a false conjecture. Note that this is a type 2b (ii) inference failure, where the preferred model is the real world, and truth and falsity in this world are revealed by the success or failure of plan execution.
- In GALILEO, a contradiction is derived from the combination of two ontologies: a theoretical one and an experimental one. So, by merging these two ontologies into a single inconsistent one, we could regard GALILEO's type 2b inference failure as of the (i) kind. However, it will be more convenient to regard the experimental ontology as describing experiments performed in the real world. Under this interpretation, GALILEO is similar to ORS, in that the preferred model is the real world and truth and falsity in that world is revealed by experimental results that are described in the experimental ontology merely for implementational convenience.

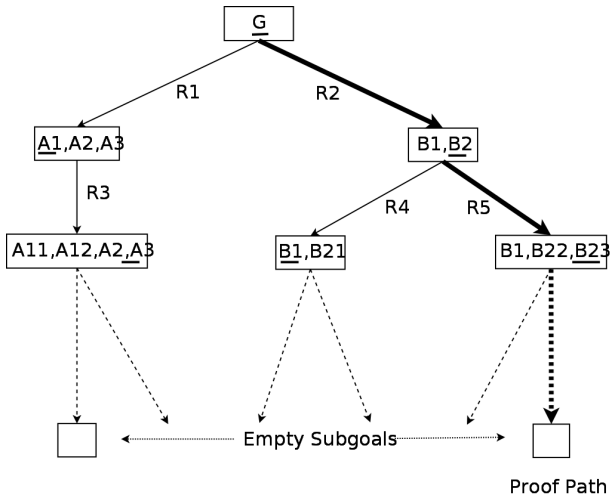
We will, therefore, concentrate on the diagnosis of type 2b (ii) faults. We leave type 2a faults for future work, but expect the diagnostic process for these faults to be essentially dual to the type 2b ones. Type 2c faults are inherently rather different and we expect the diagnostic process also to be very different. Early work on this problem includes McCarthy's mutilated checkers board problem [McCarthy, 1964] and Amarel's work on the evolution of representations of the missionaries and cannibals problem [Amarel, 1968].

In previous work it has been shown how various kinds of ontology repair operations can be represented as signature and/or theory morphisms, adapting ideas from Category Theory[1] For the purposes of the current paper, it is enough to envisage these morphisms as meta-level functions applied to the two components of ontologies. Typically, both theory and signature morphisms will be required in the overall repair of an ontology. Thus, *diagnosis* can be regard as finding these morphisms and *repair* as applying them. Since repair consists only of morphism application, the more interesting problem, and the focus of this paper, is diagnosis.

It will be convenient to represent a search space as an OR-tree (see Figure 1). Each node of the tree will consist of a set of sub-goals, where the root is the singleton set of the initial conjecture. The children of a node are the result of applying a rule of inference to one of the sub-goals and replacing it with the resulting sub-sub-goals. OR-branching occurs when there are multiple possible rule applications to a node. The

[1] Alan Smaill, personal communication (BBN) 1682.

sub-goals in a leaf node are unproven conjectures. A proof will be a path of the tree from the root to a leaf, in which the leaf subset is empty. illustrates these two kinds of tree.



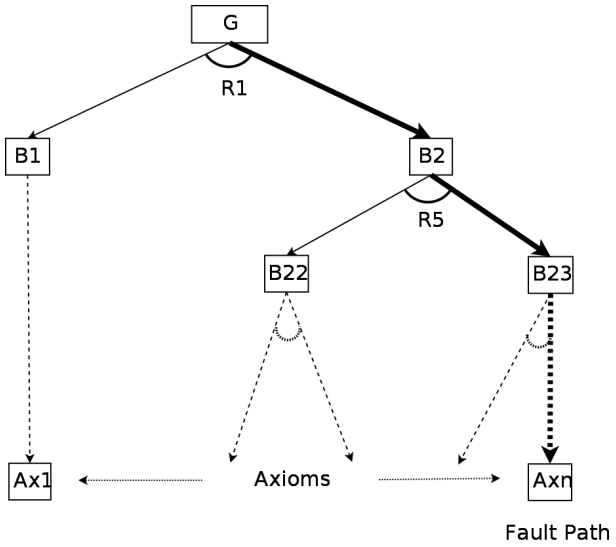
Each rectangle represents a set of AND *sub-goals, each of which must be proved. The arrows represent rules applied to the underlined sub-goal. Each path is a potential proof attempt, which completes when its leaf contains no sub-goals. The bold face path is a completed proof.*

Figure 1: A Search Space as an OR Tree

If we want to consider a proof alone, it will be convenient to consider it as an AND tree, in which each node consists of only one sub-goal (see Figure 2). The children of a node will be the sub-sub-goals arising from an application of a rule of inference to its sub-goal. The leaves will be instances of axioms.

An earlier attempt at diagnosis via proof analysis is Shapiro's work on logic programming debugging [Shapiro, 1983]. Suppose a Prolog program has unexpectedly failed. This is akin to a type 2a fault. Shapiro shows how to step through the failed proof. For each sub-goal, an oracle (the user) is asked whether the sub-goal is true or false. At each node of the OR-tree search space there will be at least one true sub-goal whose proof has failed, otherwise, the search space will contain a proof. A path of true/failed subgoals are traced through the search space until a leaf is reached. At this point the true/failed leaf sub-goal represents a missing axiom. Adding this sub-goal as a new axiom will debug the logic program.

A Shapiro-like technique is used in ORS but for type 2b faults, i.e., it is used to identify false 'facts' in the ontology, where the oracle is derivation in the ontology. This suggests that this kind of analysis can be adapted for faults of both type 2a and type 2b.



Each rectangle contains only one sub-goal. Each rule labels a family of arrows: one for each sub-goals arising from the rule's application. A proof is complete when all the leaves of the tree are instances of axioms. The bold face path represents a fault path, i.e., one in which all the sub-goals are false, including the final axiom instance.

Figure 2: A Proof as an AND Tree

4 How to Block Illegitimate Success

By 'illegitimate success' we mean that something false can be inferred in the ontology, i.e., the ontology has a type 2b (ii) fault. We now discuss the use of Shapiro-like, proof analysis to partially diagnose the fault by locating the source of the problem. Note that, in the case of type 2b faults, we don't have to explore the whole search space, as we have a proof of something false, so can restrict ourselves just to this proof.

Consider the proof as an AND-tree. The root of the tree is the false 'theorem'. The children of each node are the sub-goals that collectively imply it. We will assume that the derivation of a sub-goal by its children is via a logical rule of inference that is beyond reproach, i.e., the fault lies in the axioms of the ontology and not its underlying logic[2]. The leaves of the tree are instances of axioms.

Note that free variables in the sub-goals will arise from dual skolemisation of existentially quantified variables, and their instantiation during the proof will correspond to witnesses being discovered for these variables. Usually, all such free variables will be instantiated at some point during the proof. If any free variables are left uninstantiated then this indicates that the theorem has been proved for all their possible instantiations. Any instantiations of free variables made during the proof can be inherited back up the tree, so proof trees will usually be totally ground.

[2]If required, this assumption could be relaxed by treating the rule of inference as an additional child node.

For each false node, one of its children is also false, otherwise the node would be true. Therefore, there exists at least one path from root to leaf in which all the nodes are false. We call such a path a *fault path*.

We assume that an oracle exists that can identify false nodes, provided they are in its ontology's language. In §5 we discuss where these oracles might come from. For each false node, the oracle is asked for the truth of each of the child nodes. By recursing this process, all the fault paths can be identified. The leaves of these fault paths are all false instances of axioms. Therefore, the axioms are also false and must be repaired.

Suppose that, during this process, a sub-goal is found that is not in the oracle's language. The source ontology's signature should be repaired so that the sub-goal *is* in the oracle's ontology. If this newly repaired sub-goal is false, then the search for fault paths should resume at this sub-goal.

5 Where do the Oracles come from?

The proposals in §4 depend crucially on the existence of an oracle to identify proven, but false, subgoals. Is it realistic to assume the existence of such oracles? The answer depends on the application. We illustrate this with the two applications of ORS and GALILEO.

5.1 ORS Oracle

In ORS the oracle is provided by the service-providing agents. The assumptions about these SPAs in ORS are:

- An SPA does not have the functionality, nor would its owner usually consider it desirable, to reveal its ontology.
- An SPA is regarded as the ultimate authority of the conditions under which it is prepared to perform a service. As such, its ontology is not subject to change[3].
- An SPA will, however, answer specific questions. These are posed as KIF formulae, which the SPA will try to prove and to which it will return an answer.
- These answers can take the form of $true$, if the formula can be proven without instantiation, $false$, if the formula cannot be proven, or a substitution, indicating the instantiations necessary to prove the formula.
- An SPA may also ask questions of the planning agent (PA) during the plan formation process. These follow the same process as when the PA asks a question of the SPA.

Note that ORS oracles are able to classify non-ground subgoals and can sometimes prove them without any instantiation, i.e., for all possible values of the free variables. When instantiation is necessary to prove the subgoals, then this may prove useful in the next phase of axiom modification (see §6).

[3]However, in a current UG4 project by Agnieszka Bomersbach, we are relaxing this assumption.

5.2 GALILEO Oracle

In GALILEO the real world can be regarded as the oracle. This enforces rather different assumptions about the oracle than in ORS.

- The real world does not have an inherent ontology, but only one imposed by those trying to understand it. Consequently, its ontology is also subject to evolution as such understanding deepens.
- We can use it only to make measurements and observations of particular phenomena. These can be modelled either as ground atoms, which will be classified as $true$ or $false$, or ground function calls, for which the output will be returned. For instance, a ground predicate might be $At(Ball_1, Posn_1, Time_1)$ and a ground function call might be $Vel(Ball_1, Time_1)$, to which the answer might be $10\ metres/sec.$
- The real world cannot itself ask questions.

6 Modifying Axioms

If an instance of a faulty axiom is found, then we need to delete or modify this axiom. The simplest thing is to delete it, but that may not be optimal. In particular, the source axiom might be successfully used in proofs of true formulae. An unwanted side effect of deleting it would be that these true formulae would cease to be theorems — at least, by any proofs using the source axiom. The alternative is a modification that makes it inapplicable in the faulty proof, but retains its applicability elsewhere. In ORS, for instance, a precondition was often added to the STRIPS operators that differentiated the good uses from the bad ones.

The problem can be seen as a classification task, of the kind to which the HR system is suited [Colton *et al.*, 1999]. We form two sets of applications of the source axiom: Good and Bad. Good applications are those where it is used to prove true formulae and Bad ones where it is used to prove false formulae. In particular, its application in the faulty proof is in the Bad set. We now learn a classification that differentiates these two sets. It might, for instance, be an extra condition on the axiom or it might be an instantiation of the axiom to restrict it to Good applications. We can test our modified axiom by asking the oracle whether it is true. If not, then further or different modifications are needed.

7 Modifying Signatures

If fault is detected in the source ontology's signature, then we need to figure out what signature morphism, ν_σ, to construct. ν_σ must map the false sub-goal to a target sub-goal that *is* in the oracle's signature. There are many ways to do this: the target sub-goal can be any formulae in the oracle's signature. Ideally, however, we would like the source and target sub-goals to be similar, e.g., by minimising the edit difference between them. Another approach, that has been explored by Theodosia Togia in a recent MSc project, is the use of computational linguistics tools, such as Wordnet, to identify synonymous and similar connections between ontology signatures [Togia *et al.*, 2010]. In practice, ν_σ will typically be of the type listed in §3 or its dual, namely: naming apart/merging

of functions, permuting arguments and adding/removing arguments.

As in ORS we will make the following assumptions:

- We have no direct access to the oracle's ontology, i.e., we can't just browse its ontology for likely targets.
- We can, however, ask the oracle questions. So, we can construct a candidate target, send it to the oracle and expect an answer of one of the three forms: (i) in ontology's language and true, (ii) in ontology's language and false, (iii) not in ontology's language . If it is in the ontology's language then we can proceed, and if it is also false then we can use it to resume the search for a fault path.

As in ORS, we may already have some prior communication with the oracle, e.g., when discovering that the original 'theorem' is false. ORS has the notion of a *surprising question*, i.e., a question asked by the oracle that was not expected. Suppose that this question was not in the original ontology. Then it (or some instance of it) might be a candidate for the target. For instance, the PA might expect to be asked $Pay(PA, £100)$, since this instantiates a preconditions of one of its STRIPS operator. The SPA might, however, ask it $Pay(PA, £100, CreditCard)$ instead. This surprising question is a clue that it should modify the type of its Pay predicate from binary to ternary, where the extra argument defines the method of payment.

In the worst case, arbitrary modifications could be applied to the source, e.g., permuting and adding arguments, replacing names with synonyms, and then sent to the oracle for testing until one is in the oracle's signature. These might be explored breadth-first, e.g., first trying all one-step modifications, then all two-step, etc. This effectively uses minimisation of edit distance as a heuristic.

8 Conclusion

We have initiated a generic theory of diagnosis of faulty ontologies. The proposals here are based on, but generalise, our experience with the GALILEO and ORS systems. We have made some initial simplifying assumptions, which we hope will not restrict the application of the proposals to new areas. In particular, we are looking at repairing faulty ontologies where the fault is revealed by an inference failure and the repair is implemented by a signature and/or theory morphism. More concretely, in this note, we focus on situations where a false conjecture has been proved. Diagnosis consists of constructing a morphism by analysis of failed inference. It is assumed that an oracle is available that can answer (and sometimes ask) questions, but whose own ontology is otherwise inscrutable.

References

[Amarel, 1968] S. Amarel. On representations of problems of reasoning about actions. In D. Michie, editor, *Machine Intelligence 3*, pages 131–171. Edinburgh University Press, 1968.

[Bundy and Chan, 2008] Alan Bundy and Michael Chan. Towards ontology evolution in physics. In Wilfrid Hodges and Ruy de Queiroz, editors, *Logic, Language, Information and Computation*, volume 5110 of *Lecture Notes in Computer Science*, pages 98–110. Springer Berlin / Heidelberg, July 2008.

[Colton *et al.*, 1999] S Colton, A Bundy, and T Walsh. HR: Automatic concept formation in pure mathematics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence, Stockholm, Sweden*, pages 786–791, 1999.

[McCarthy, 1964] J. McCarthy. A tough nut for proof procedures. Stanford Artificial Intelligence Project Memo 16, Stanford University, 1964.

[McNeill and Bundy, 2007] F. McNeill and A. Bundy. Dynamic, automatic, first-order ontology repair by diagnosis of failed plan execution. *International Journal On Semantic Web and Information Systems*, 3(3):1–35, 2007. Special issue on ontology matching.

[Shapiro, 1983] E.Y. Shapiro. *Algorithmic Program Debugging*. MIT Press, 1983.

[Togia *et al.*, 2010] Theodosia Togia, Fiona McNeill, and Alan Bundy. Harnessing the power of folksonomies for formal ontology matching on the fly. In *Proceedings of the ISWC workshop on Ontology Matching*, November 2010.

Enriching Wikipedia Vandalism Taxonomy via Subclass Discovery

Si-Chi Chin
Interdisciplinary Graduate Program in Informatics (IGPI)
The University of Iowa
Iowa City, U.S.A.
si-chi-chin@uiowa.edu

W. Nick Street
Management Sciences Dept.
IGPI
The University of Iowa
Iowa City, U.S.A.
nick-street@uiowa.edu

Abstract

This paper adopts an unsupervised subclass discovery approach to automatically improve the taxonomy of Wikipedia vandalism. Wikipedia vandalism, defined as malicious editing intended to compromise the integrity of the content of articles, exhibits heterogeneous characteristics, making it hard to detect automatically. The categorization of vandalism provides insights on the detection of vandalism instances. Experimental results demonstrate the potential of using supervised and unsupervised learning to reproduce the manual annotation and enrich the predefined knowledge representation.

1 Introduction

Wikipedia, among the largest collaborative spaces open to the public, is also vulnerable to malicious editing – vandalism. Wikipedia defines vandalism as "any addition, removal, or change of content made in a deliberate attempt to compromise the integrity of Wikipedia[1]." The characteristics of Wikipedia vandalism are heterogeneous. It can a be large-scale editing, such as deleting the entire article or replacing the entire article with irrelevant content. It can be some irrelevant, random, or unintelligible text (e.g. *dfdfefefd #$%&@@#, John Smith loves Jane Doe.*) It can be a small change of facts (e.g. *This is true → This is not true*.) It can also be an unregulated formatting of text, such as converting all text to the font size of titles. Figure 1 illustrates a taxonomy of Wikipedia actions, highlighting the diverse vandalism instances. The reasons to structure the knowledge of Wikipedia vandalism include:

- sharing common understanding of Wikipedia vandalism,
- making knowledge of Wikipedia vandalism explicit and enabling its reuse,
- providing insights on how vandalism instances are different from legitimate edits, and
- improving the accuracy of Wikipedia vandalism detection.

The detection of Wikipedia vandalism is an emerging research area of the Wikipedia corpus. Prior research emphasized methods to separate the malicious edits from the well-intentioned edits [West *et al.*, 2010; Chin *et al.*, 2010; Smets *et al.*, 2008; Potthast *et al.*, 2008]. Research has also identified common types of vandalism[Vigas *et al.*, 2004; Priedhorsky *et al.*, 2007; Potthast *et al.*, 2008]. However, categorizing vandalism instances relies on laborious manual efforts. The heterogeneous nature of vandalism creates challenges for the annotation process. For example, a "misinformation" vandalism instance can be quite similar to a "nonsense" or a "graffiti" instance [Priedhorsky *et al.*, 2007; Chin *et al.*, 2010]. Current research has yet to establish a standardized or commonly accepted approach to construct the knowledge representation of vandalism instances. In this paper, we introduce an unsupervised learning approach to automatically categorize Wikipedia vandalism. The approach uses statistical features to discover subclasses in both the positive and negative spaces, identifying the partitions that perform the best in multi-class classification. The proposed approach aims to:

- enrich the Wikipedia vandalism taxonomy and knowledge representation automatically,
- improve vandalism detection performance,
- identify potential multi-label instances, and
- identify potential annotation errors.

The paper is structured as follows. In Section 2, we describe the data sets used for our experiments, and detail the implementation of the system. In Section 3 we present our experimental results. In Section 4, we review previous academic research on knowledge representation of Wikipedia vandalism and subclass discovery. In Section 5, we conclude the paper and discuss the opportunities for future work.

2 Experimental Setup

The experiments used the annotated Microsoft vandalism data set provided by Chin et al. [Chin *et al.*, 2010] [2] The dataset has 474 instances with 268 vandalism instances, comprising 21 features extracted from the Statistical Language Model [Clarkson and Rosenfeld, 1997] and Unix *diff* procedure. It also has annotations of 7 types of vandalism : *blanking, large-scale editing, graffiti, misinformation, link spam,*

[1] http://en.wikipedia.org/wiki/Wikipedia:Vandalism

[2] http://code.google.com/p/wikivandalismdata/downloads/list

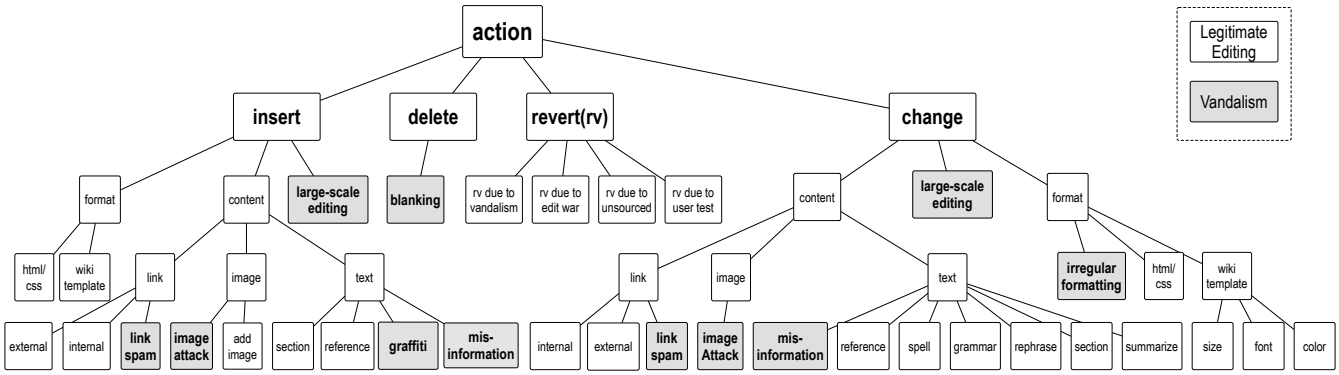


Figure 1: Wikipedia Action Taxonomy. The taxonomy groups Wikipedia editing by the four primary actions (change, insert, delete, and revert) and types of change (format and content), considering also the scale of editing. The shaded boxes are types of Wikipedia vandalism.

irregular formatting, and image attack. The distribution of the 7 types is shown in Figure 3.

Our approach combines unsupervised and supervised learning. Broadly, we use a clustering method to segment both the positive and negative spaces, allowing a better representation for the disjunctive nature of both vandalism and legitimate edits. The cluster memberships are then used as labels in a multi-label classification scheme. Our evaluation, however, is based on the original two labels.

The data was first shuffled into 10 randomized sets. For each shuffle, we clustered the data using k-means clustering. Classification was performed using a support vector machine (SVM) with RBF kernel, using a grid search to find the optimal C and γ parameters. For each shuffle, we used 9/10 of the data as the training set, using the parameters learned from the grid search, to learn a multi-class SVM classifier. The RBF kernel produces a highly nonlinear decision boundary for the disjunctive concept, allowing more accurate results compared to a linear boundary. To evaluate the results, we performed 10-fold cross-validation for each shuffle and averaged the ranked results. The optimal partition was selected based on the average precision (AP) [3] and the Area Under ROC Curve (AUC) metrics. We compared the unsupervised experiment results with the manually annotated results. Figure 2 shows a flowchart of the proposed approach and the design of the experiments. We used Weka [Hall *et al.*, 2009] to implement all experiments.

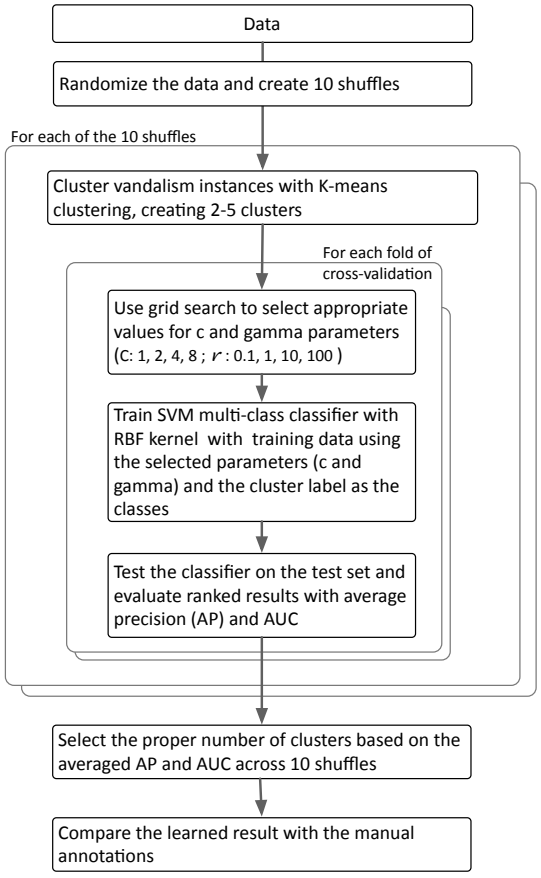


Figure 2: Experiment flowchart

3 Experiment Results

3.1 Unsupervised Clustering vs. Manual Labeling

The experiments used unsupervised clustering to determine the optimal partitions of data that performed the best in the multi-class classification. The clusters were then compared to

[3] We used the following definitions to compute the average precision (AP):

$AP = \frac{\sum_{r=1}^{N}(P(r)\times \mathrm{rel}(r))}{\textit{number of relevant documents}}$

$P(r) = \frac{\textit{relevant retrieved documents of rank r or less}}{r}$

the manual annotations to explore the opportunity of enriching the predefined knowledge representation of Wikipedia vandalism.

Tables 1 and 2 show the multi-class classification performance for 20 different combinations of positive and negative classes. Both metrics indicate the ideal number of clusters are three for the positive space and four for the negative space. The multi-class classification, compared to the binary classification, increase the AP from 0.425 to 0.443 and the AUC from 0.711 to 0.737. The increases are significant compared to the baseline binary classification.

	P.2	P.3	P.4	P.5
N.1	0.42832	0.43634	0.43874	0.44211
N.2	0.42522	0.43097	0.43720	0.43573
N.3	0.42789	0.43884	0.43538	0.43259
N.4	0.43197	**0.44374**	0.43675	0.43707
N.5	0.42999	0.43878	0.43242	0.43064
Baseline (binary class): **0.42752**				

Table 1: Average Precision (AP) scores of 20 combinations of positive and negative subclasses.

	P.2	P.3	P.4	P.5
N.1	0.71676	0.72800	0.72431	0.72592
N.2	0.71377	0.71648	0.72447	0.72264
N.3	0.71936	0.73366	0.72505	0.72298
N.4	0.72358	**0.73723**	0.72912	0.72640
N.5	0.72434	0.73021	0.72192	0.71693
Baseline (binary class): **0.71144**				

Table 2: Area under curve (AUC) scores of 20 combinations of positive and negative subclasses.

3.2 Enhanced Taxonomy Recommendation

We manually examined the content of vandalism instances in each cluster in order to answer the following questions:

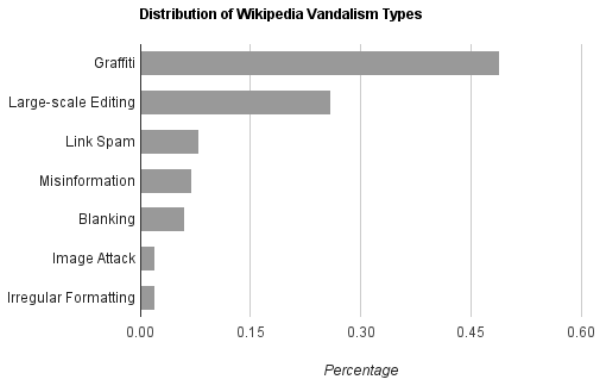


Figure 3: Distribution of Wikipedia Vandalism Types

- How are the instances of large-scale editing and graffiti different from each other in the three clusters?
- Can we identify annotation errors?

Table 3 presents the comparison between the results of clustering and the manual annotation. It is observed that about two-thirds of the graffiti instances are assigned to Cluster 2 with the remaining third assigned to Cluster 3. It is also noted that the large-scale editing instances appeared in all three clusters. The content analysis of the clusters provides insights to enhance the predefined taxonomy, and to discover multi-label instances and annotation errors.

Three Types of Large-scale Editing

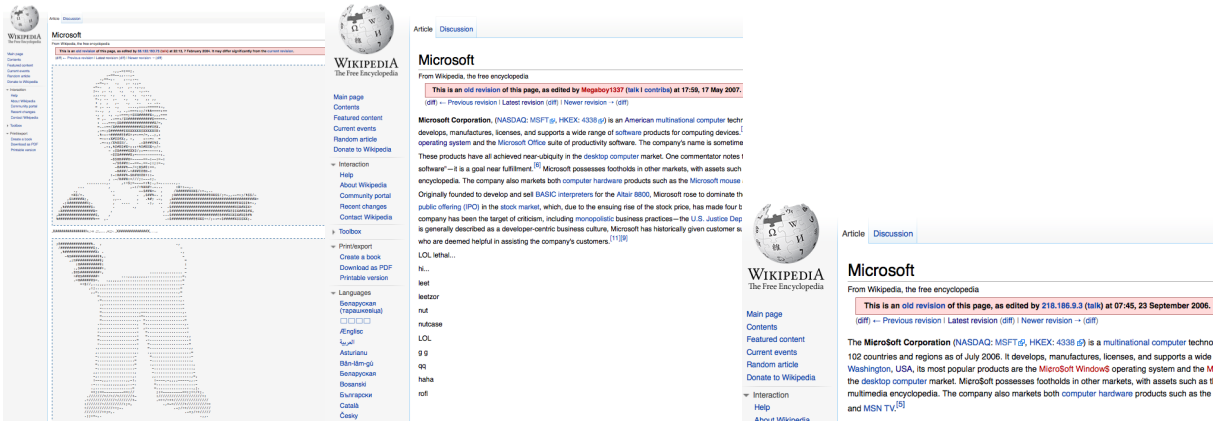
We observed, from Table 3, three different types of large-scale editing. Figure 4 exemplifies three typical instances of large-scale editing from each of the three clusters. The feature space contains three clusters of the large-scale editing instances. We manually examined the data in each cluster to characterize the three types of large-scale editing.

We observed that Cluster 1 contains large insertions of text with diverse vocabulary, usually co-occurring with massive deletion of existing text. For example, we found an ASCII art of the cartoon figure Homer Simpson[4], a complete gibberish text[5], replacing the article with the Apple Computer, Inc article[6], and massive replacement of spelling [7]. Cluster 2 contains the large-scale editing instances that have massive insertion of text with a substantial amount of deletion. [8] [9] Cluster 3 contains instances with numerous spelling changes and named entity replacements, for example, changing "Microsoft" to "Nintendo" ; "Bill Gates" to "George Bush"[10];

[4]http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=2330007
[5]http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=9122754
[6]http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=81420090
[7]http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=9923514
[8]http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=24305432
[9]http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=131585774
[10]http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=62013580

Cluster	Types	Count	Recall
1	**Large-scale Editing**	28	96 %
	Blanking	1	
2	**Graffiti**	84	83 %
	Misinformation	18	
	Link Spam	15	
	Large-scale Editing	10	
	Blanking	6	
	Irregular Formatting	2	
	Image Attack	2	
3	**Graffiti**	46	56 %
	Large-scale Editing	32	
	Blanking	9	
	Link Spam	7	
	Irregular Formatting	4	
	Image Attack	4	
Total number of vandalism instances:		268	74 %

Table 3: Cluster analysis of vandalism types



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3

Figure 4: Typical large-scale editing in the three clusters. Edits in Cluster 1 involved large insertion of rich and diverse text. Edits in Cluster 2 involved mass insertion with substantial deletion. Edits in Cluster 3 involved the replacement of named entities and spellings.

"Microsoft" to "Micro$oft."[11]

Two Types of Graffiti: Large vs. Minor Scale

Graffiti is an insertion of unproductive, irrelevant, random, or unintelligible text. We examined the two types of graffiti in Cluster 2 and Cluster 3. We found that graffiti in the Cluster 2 involved insertion of short irrelevant text, such as "LOOK AT ME I CAN FLY!!!![12]" or "I like eggs... [13]". Graffiti in the Cluster 3 involves inserting short unintelligible text, such as "blurrrrrgj[14]," "dihjhkjk, [15]," and "asfasfasf[16]."

Multi-label Instances and Annotation Errors

Although the predefined taxonomy (see Figure 1) considered both the amount of edit (i.e. How much has been changed compared to the previous edits?) and the content characteristics of edits (i.e. What are the edits?), categories that overlap two dimensions are absent in the taxonomy. However, the content analysis indicates numerous instances of copy-and-paste of irrelevant text that has both characteristics of large-scale editing and graffiti [17] [18] [19] [20] [21], as well as massive deletion mixed with graffiti[22] [23].

The results confirm the diverse nature of Wikipedia vandalism, indicating the possibility of improvement for the predefined taxonomy. For example, to include multi-label instances, creating new categories such as "Repeating graffiti (see Figure 5)" to describe the large amount of repeating insertion of irrelevant text, or "Erasure by graffiti" to describe the replacement of majority of content with nonsensical words would enrich the knowledge representation of Wikipedia vandalism.

We searched for the irregular distribution patterns from Table 3 to investigate potential annotation errors. The single blanking instance in the Cluster 1 should actually be a large-scale editing[24]. This finding shows the potential of our approach to amend annotation errors.

4 Related Work

Previous research has identified many common types of vandalism. Viégas et al. [Vigas *et al.*, 2004] identified five

[11] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=77323423

[12] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=28384195

[13] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=13233361

[14] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=86731761

[15] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=78923750

[16] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=69519551

[17] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=45456321

[18] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=41754476

[19] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=27056109

[20] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=12899659

[21] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=24631945

[22] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=76785744

[23] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=63662542

[24] http://en.wikipedia.org/w/index.php?title=Microsoft&oldid=89513491

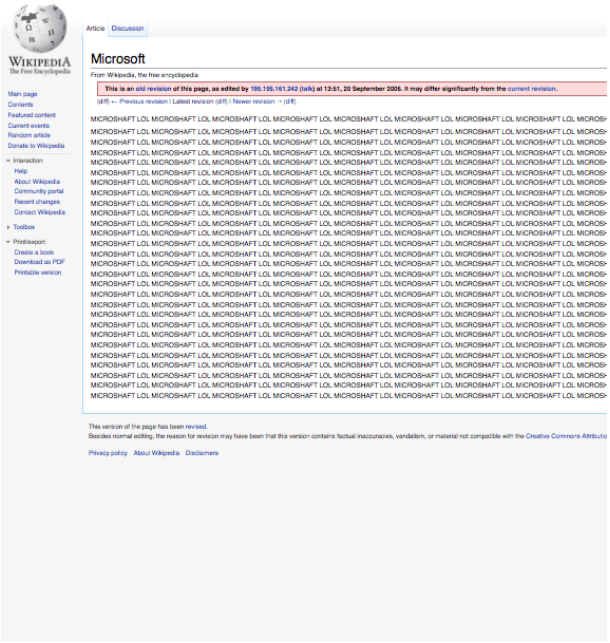


Figure 5: An example of mixed-type graffiti. This instance involves the replacement of entire Microsoft article with repeating nonsensical text.

common types of vandalism: mass deletion, offensive copy, phony copy, phony redirection, and idiosyncratic copy. Priedhorsky et al. [Priedhorsky *et al.*, 2007] categorized Wikipedia damaged edits[25] into seven types: misinformation, mass delete, partial delete, offensive, spam, nonsense, and other. Potthast et al. [Potthast *et al.*, 2008] organized vandalism edits according to the "Edit content" (text, structure, link, and media) and the "Editing category" (insertion, replacement, and deletion). Chin et al. [Chin *et al.*, 2010] constructed a taxonomy of Wikipedia editing actions based on the four primary actions (change, insert, delete, and revert) and types of change (format and content). They identified 7 types of vandalism : *blanking, large-scale editing, graffiti, misinformation, link spam, irregular formatting, and image attack.* The categories proposed in prior works were primarily based on empirical observations of researchers, and can be made more comprehensive or systematically. In our work, we propose using unsupervised clustering and supervised multi-class classification to discover and enrich the knowledge representation of Wikipedia vandalism.

Classification problems involve assigning data to observed categories. In the setting of binary classification, the data has only two classes: positive and negative. However, binary classification becomes difficult in the presence of a heterogeneous positive space. An increasing number of papers have discussed motivations and methods of multi-class classification. [Li and Vogel, 2010a; Lorena *et al.*, 2008; Garca-Pedrajas and Ortiz-Boyer, 2011; Tsoumakas *et al.*, 2010; Zhou *et al.*, 2008]. Subclass classification is subset of multi-class classification, where the multiple class labels belong to a hierarchical structure, and has been shown to enhance classification accuracy. Li and Vogel [Li and Vogel, 2010a; 2010b] utilized sub-class partitions to achieve better performance than the traditional binary classification on the 20 newsgroups dataset. Assent et al. [Assent *et al.*, 2008] incorporated class label information to provide appropriate groupings for classification.

Our work recognizes the heterogeneous nature of Wikipedia Vandalism, discovering clusters that achieved the best performance in the subclass classification. We use the information of discovered subclasses to evaluate and enrich the predefined Wikipedia vandalism categories.

[25] Although damage edits were not referred to as vandalism in their work, they were in fact in line with the definition of Wikipedia vandalism.

5 Conclusion and Future Directions

This paper addresses the problem of detecting diverse Wikipedia vandalism categories, and the problem of recommending appropriate knowledge representation of Wikipedia vandalism instances. We used k-means clustering to map learned categories to a predefined taxonomy, and used supervised classification and content analysis to assist the discovery of novel categories, multi-label instances, and annotation errors.

Wikipedia vandalism detection has previously been regarded as a binary classification problem: ill-intended edits vs. well-intended edits. However, the characteristics of Wikipedia vandalism are in fact heterogeneous. Therefore, our work approached it as a multi-class classification problem, and used unsupervised learning to enhance the manual annotations. Our experimental results showed enhanced performance from the use of multi-class classification method. The results also demonstrated the ability to automate the process of discovering and enriching the Wikipedia vandalism knowledge representations using unsupervised learning.

Future work may include more annotated datasets and comparing the knowledge representation schema between different articles. It is also valuable to investigate how the learned knowledge could be transferred from one articles to the others. Future work may also explore the temporal aspect of the knowledge representation, describing the dynamic evolution of Wikipedia vandalism categories.

References

[Assent *et al.*, 2008] I. Assent, R. Krieger, P. Welter, J. Herbers, and T. Seidl. SubClass: Classification of multidimensional noisy data using subspace clusters. *Advances in Knowledge Discovery and Data Mining*, page 4052, 2008.

[Chin *et al.*, 2010] Si-Chi Chin, W. Nick Street, Padmini Srinivasan, and David Eichmann. Detecting Wikipedia vandalism with active learning and statistical language models. In *Proceedings of the 4th Workshop on Information Credibility*, WICOW '10, pages 3–10, New York, NY, USA, 2010. ACM. ACM ID: 1772942.

[Clarkson and Rosenfeld, 1997] Philip Clarkson and Ronald Rosenfeld. Statistical language modeling using the CMU-Cambridge toolkit. pages 2707—2710, 1997.

[Garca-Pedrajas and Ortiz-Boyer, 2011] Nicols Garca-Pedrajas and Domingo Ortiz-Boyer. An empirical study of binary classifier fusion methods for multiclass classification. *Information Fusion*, 12:111130, April 2011. ACM ID: 1920692.

[Hall *et al.*, 2009] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11:1018, November 2009. ACM ID: 1656278.

[Li and Vogel, 2010a] B. Li and C. Vogel. Improving multiclass text classification with error-correcting output coding and sub-class partitions. *Advances in Artificial Intelligence*, page 415, 2010.

[Li and Vogel, 2010b] B. Li and C. Vogel. Leveraging subclass partition information in binary classification and its application. *Research and Development in Intelligent Systems XXVI*, page 299304, 2010.

[Lorena *et al.*, 2008] Ana Carolina Lorena, Andr C Carvalho, and Jo\ ao M Gama. A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*, 30:1937, December 2008. ACM ID: 1670491.

[Potthast *et al.*, 2008] Martin Potthast, Benno Stein, and Robert Gerling. Automatic vandalism detection in Wikipedia. In Craig Macdonald, Iadh Ounis, Vassilis Plachouras, Ian Ruthven, and Ryen White, editors, *Advances in Information Retrieval*, volume 4956 of *Lecture Notes in Computer Science*, pages 663–668. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-78646-7_75.

[Priedhorsky *et al.*, 2007] Reid Priedhorsky, Jilin Chen, Shyong (Tony) K. Lam, Katherine Panciera, Loren Terveen, and John Riedl. Creating, destroying, and restoring value in Wikipedia. In *Proceedings of the 2007 International ACM Conference on Supporting Group Work*, pages 259–268, Sanibel Island, Florida, USA, 2007. ACM.

[Smets *et al.*, 2008] K. Smets, B. Goethals, and B. Verdonk. Automatic vandalism detection in Wikipedia: Towards a machine learning approach. In *AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy*, page 4348, 2008.

[Tsoumakas *et al.*, 2010] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer US, 2010. 10.1007/978-0-387-09823-4_34.

[Vigas *et al.*, 2004] Fernanda B. Vigas, Martin Wattenberg, and Kushal Dave. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 575–582, Vienna, Austria, 2004. ACM.

[West *et al.*, 2010] Andrew G West, Sampath Kannan, and Insup Lee. Detecting Wikipedia vandalism via spatio-temporal analysis of revision metadata. In *Proceedings of the Third European Workshop on System Security*, EUROSEC '10, page 2228, New York, NY, USA, 2010. ACM. ACM ID: 1752050.

[Zhou *et al.*, 2008] Jie Zhou, Hanchuan Peng, and Ching Y Suen. Data-driven decomposition for multi-class classification. *Pattern Recognition*, 41:6776, January 2008. ACM ID: 1285197.

When owl:sameAs isn't the Same Redux: A preliminary theory of identity and inference on the Semantic Web

Abstract

The Web changes knowledge representation in a number of surprising ways, and decentralized knowledge representation systems such as the Semantic Web will require a theory of identity for the Web beyond current use of *sameAs* links between various data-sets. We propose that one problem, as posed by the semantics of *sameAs* in OWL, is enshrining a metaphysical notion of individual identity into the formal semantics of the logic that prevents universal identity linking across semantic roles. However, as shown by languages such as RDF and KIF, a metaphysics-free semantics is possible that can handle both classes and properties as first-class citizens of the universe of discourse that can have identity statements and so produce valid inferences. We empirically analyze the behavior of identity and inference on the Semantic Web currently in order to analyze the size of the problem, and show the non-normal distribution of equivalence classes inference produces. We also show via a human experiment how inference leads to more difficulty in judgments about identity by individual experts, yet adding experts leads to fairly reasonable results, even over inferred triples.

1 Introduction

With the beginning of the deployment of the Semantic Web, the problem of identity in knowledge representation – sometimes assumed trivially solved – has returned with a vengeance. In traditional logic names are arbitrary strings, but by definition, one criterion that distinguishes a Web logic from traditional logic is that in Web logic names are URIs (Uniform Resource Identifiers, such as *http://www.example.org/)* (henceforth abbreviated as *ex:*). On the 'actually existing' deployed Semantic Web known as Linked Data, a URI not only can refer to things, but can be accessed, so that accessing *ex:Paris* in a Web browser results in receiving a bundle of statements in a Web logic such as RDF (Resource Description Format, where statements are composed of 'triples' of subject-property-object names). When users of Semantic Web find another URI about their item of interest, they connect their two different URIs with an *identity link*(usually a *owl:sameAs* property, henceforth just called *sameAs*), which means that the two things referred to by the URIs are identical, and thus the URIs are equal as names [Patel-Schneider *et al.*, 2004]. For example, if someone wanted to state that the city Paris in DBpedia (a Semantic Web version of Wikipedia) is the same as the city of Paris in my data-set, then they could state *dpedia.org:Paris owl:sameAs ex:Paris* in their data-set. However, there is what has been termed a 'semantic' elephant in the room that some refuse to see, namely that *sameAs* also has a formal semantics whose intended use is between individuals who share all the same properties. The results of inference using *sameAs* are often surprising, either 'smushing' up distinct individuals or failing to 'smush' individuals due to their semantic role. One solution would be to forget the semantics of the Semantic Web entirely, treating *sameAs* as just an English language mnemonic. Those that do not remember history are doomed to repeat it; for it was precisely the forgoing of logical semantics that led to semantic networks having their infamous crisis over divergence in meaning in *IS-A* links [Brachman, 1983]. One can only imagine that such problems will be increased given the global scale of the Semantic Web. We point out that a new formal semantics is needed for identity links if the Semantic Web is going to succeed in deployment. In Section 2 we explicate identity in formal semantics, and show two different ways that equality across different semantic roles can be implemented formally, one known as 'punning' and the other known as 'pollarding.' We empirically analyze identity links on the Semantic Web, with a focus on *sameAs*, in Section 3. An experimental investigation of whether real identity links and their concomitant inferences are accurate or not is presented in Section 4, and then the results and future prospects are analyzed.

2 Identity in Logic

2.1 Equality

Identity is logically defined as equality, $A = A$. The basic relation of identity holds only between a thing and itself. So, it is important not to think of an equality sentence such as $A = B$ as saying that there are two things, A and B, which are equal. This is a category of error of believing that two separate *names* A and B in the syntax can be confused with the *things* in the universe of discourse of the semantics. Properly,

$A = B$ says that there is one thing which has two names, 'A' and 'B' respectively. To say A equals B, $A = B$, is to say that A is identical with B, so that if $A = B$ is true then the set $\{A, B\}$ is identical to the set $\{A\}$ in the semantics with $A = B$ =[some individual in the semantics]. This means that all notions such as 'approximately equal' or 'equal for some purposes' or 'equal with respect to certain facets,' etc., are not the same as logical *equal*. All of these other kinds of relations require two entities to be at least logically distinguishable in order to be not equal for all purposes, such as being not equal with regards facets and the like. All such 'nearly equal' notions are relations between two things, and so are fundamentally different from logical equality.

A hallmark of logical equality is that anything said about a thing using one of its names should be just as true when said using the other name. If some sentence with a syntax ...A...A...A... containing A is true, and $A = B$ is true, then the same sentence replacing 'A' with 'B' syntactically one or more times must also be true, i.e. ...B...A...B... should be true. This is the basic inference rule, substitution, associated with equality. Notice that this rule makes no mention of how the name 'A' is used in the sentence, of its semantic role. For true equality, substitutivity should apply to names *regardless of their semantic role* (i.e. what it is that A denotes, be it a function, a class, a proposition, an individual), since the intuitive argument for the validity of the rule is semantic: the meaning of a sentence is determined by the things referred to by the names in the sentence rather than the names themselves. Since logical sentences are *de re* - 'about the thing' it follows that the choice of which name to use to denote a thing must be irrelevant to the truth of the sentence.

2.2 Punning

However, there are essentially two kinds of semantics so far proposed for as logics for the Web: logics such as OWL DL (Description Logic)[Patel-Schneider *et al.*, 2004] and OWL2[Motik *et al.*, 2009] that only allow substitivity in the same semantic role and logics such as RDF[Hayes, 2004] and ISO Common Logic[Delugach, 2007] that allow substitivity regardless of semantic role. One can think of logics as either partitioning the universe of discourse into disjoint sets of things or having *all* things in a single universe of discourse. Languages that partition the universe of discourse must have separate equality operators for each kind of thing in the universe of discourse, so OWL has *owl:sameAs* for individuals, *owl:equivalentClass* for classes, and *owl:equivalentProperty* for properties [Patel-Schneider *et al.*, 2004]. It is incorrect to use *sameAs* across a class and an individual in OWL DL, and if one does so, one immediately falls into OWL Full. This semantic partitioning is reflected then in the partitioning of names in the language itself into lexical roles. So a name in OWL DL had always to refer to either properties, classes, or individuals, but identity can never be made between different semantic roles.

This inability to have the same name used across different semantic roles was corrected in OWL2 by the introduction of 'punning.'[Motik *et al.*, 2009] This refers to a technique for reducing the number of lexical categories used in a formalism by allowing a single name to be used in a variety of semantic roles so allowing a property like *ex:Latitude* to serve as the subject of a statement, an important use-case for metamodelling. For example, the language may allow punning between class and property names only if the syntax unambiguously assigns a class or property role to every occurrence of a name. This ensures that any language which uses either of these techniques can - in principle - be replaced by an equivalent language which does not use it by replacing each occurrence of a punned name by an alternative name of a recognizable type which is reserved for use in that particular semantic role. We will call such a language *segregated*. Conventional textbook accounts of logic usually define segregated languages. For example, a conventional syntax for FOL distinguishes 1+2ω categories of names, with the roles being individual, function of arity n and relation of arity n, which map respectively to elements of the universe, n-ary functions over the universe and n-ary relations over the universe. Only the first kind of name can be bound by a quantifier, since FOL allows quantification only over elements of the semantic universe, which is required to not contain relations and functions.

Punning works by allowing the name categories to intersect (or simply to be identified) while retaining the conventional interpretation mappings. This amounts to the use of multiple interpretation mappings between names and the semantics. Each name is given several denotations in an interpretation, and the immediate syntactic context is used to 'select' which one of these to use in the semantic truth recursion. Thus equality statements across all different kinds of lexical roles can be made, and the name will be given an interpretation that fits into the necessary semantic role. If a name is used in multiple semantic roles, then the name will denote a different thing in each semantic role.

2.3 Pollarding

'Pollarding' refers to a different, but closely related, technique which is used, in various forms, in the semantics of RDF and RDFS [Hayes, 2004], OWL Full [Patel-Schneider *et al.*, 2004], and ISO Common Logic [Delugach, 2007]. Pollarding retains the single denotation mapping of a conventional interpretation, but treats all names as denoting individuals, and the other semantic constructions are related to individuals – not names! – by extension mappings, which are treated as part of the interpretation. Just as with punning, the immediate syntactic context of a name is used to determine whether it is intended to be interpreted as the immediate denotation (that is, an individual) or one of the 'extensions' associated with that individual. Therefore any name that is syntactically declared equal to another can be substituted across all sentences while maintaining the truth value of the sentences, and unlike punning, in any interpretation both names denote the same individual, albeit the same individual may have multiple extension mappings.

The two schemes are illustrated in Figure 1. Figure 2 illustrates how the two schemes reduce to a classical model theory when the language is, in fact, segregated. The case of punning is trivial: the extra mappings are simply ignored; and given any classical interpretation, one can create a punning interpretation simply by adding the extra mappings in some arbitrary way. Pollarding is a bit more complicated.

The classical mappings for non-individual names are created by composing the interpretation and extension mappings. In the other direction, given a classical interpretation, one has to select an individual in the universe to be the 'representative' of each non-individual semantic entity, map the name to it, and assign the individual to be its appropriate extension (One way to do this is to use a Herbrand-style construction: put the name itself into the universe, use the classical interpretation mapping as the extension function, and treat the name as denoting itself.)

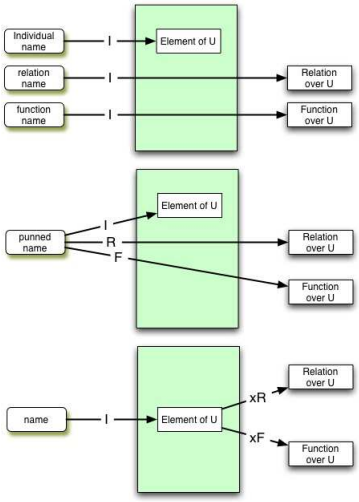


Figure 1: For an individual: lexical and semantic segregation (top), punning (middle), and pollarding (bottom).

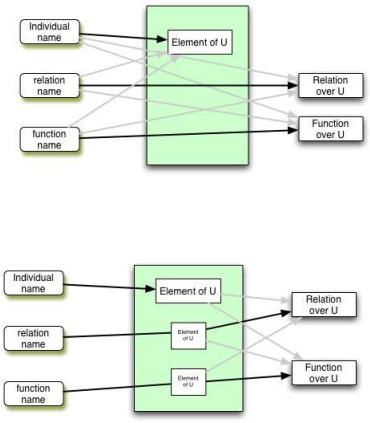


Figure 2: For an individual, relation, and function: punning (top) and pollarding (bottom). Grey arrows are interpretation functions.

As this shows, the pollarding technique does impose one requirement on the classical interpretation, viz. that its universe is large enough. If the cardinality of a classical universe is smaller than the number of non-individual (class, relation, property or function) semantic entities required to interpret the language, then that universe cannot be used as the basis of a pollarded interpretation of the same language. An example illustrating this is $\forall x.y.(x = y)$, $P(a)$, and $\neg Q(a)$. This set of sentences is satisfiable, but cannot be satisfied in a pollarded interpretation since the two relations P and Q must be distinct, but there is only one entity in the universe, which is therefore too small to provide the distinct relational extension mappings required. This is only a concern in cases where the universe can be consistently made small. Languages such as RDF and Common Logic, whose universes are required to be at least denumerable, will always have a sufficient cardinality. One possible objection to pollarding is that it seems to make the logic 'higher-order.' Although pollarding is not used in textbook first-order logic [Andrews, 2002], pollarding does not make a logic higher-order. The difference between higher-order and first-order logic is that higher-order logics all impose comprehension principles upon the universe, which is not necessary in pollarding.

3 Empirical Analysis of Inference

However, *sameAs* is already being used 'in the wild' in Linked Data, and an inspection of its behavior is in order before considering improving its semantics. First, can one actually infer anything from these *sameAs* links on Linked Data? To test this, we used as our data-set a crawl of the accessible Linked Data Web, so that our results would be an accurate 'snapshot' of the use of *sameAs* on the Web. This resulted in 10,850,606 *sameAs* statements being found (forming 7,229,140 equivalence classes), with the average number of URIs in an equivalence class being 2.00 *sameAs*. No blank nodes and 9 literals were found in the set of equivalence classes (the literals from the treatment of a URI as a string in RDF), and the total number of distinct URIs was 14,461,722. The largest equivalence class explicitly declared was of size 22, the particular case being an equivalence class for Semantic Web researcher Denny Vrandenic. *sameAs* statements substantially outweigh *owl:equivalentProperty* (1,451 occurrences) and *owl:equivalentClass* statements (106,305) on the Semantic Web. Do any of these statements violate the segregation of individuals, classes, and properties found in OWL? Without any inference, our sample showed that only a small number (3,636) of them did.

Then, using an Amazon Cloud instance with four reducers, we ran *sameAs* reasoning over the set of statements containing at least one *sameAs* statement. These produced, excluding symmetric statements (which would trivially create a 'double' of every statement), 60,045,705 additional inferred *sameAs* statements with 79,276,264 distinct URIs. The 40 largest equivalence classes of (or closures over) *sameAs* connections (via their connection to a single 'pivot' URI) are visualized in Figure 3. This shows the range of sizes varies dramatically, with the largest closure containing 4,177 URIs, seemingly consisting of all sorts of biomedical data, and originating at *UniProt*, a massive protein knowledge-base. As the number of RDF-enabled biomedial databases surely does not measure in the thousands, one wonders if at some point the *sameAs* chain got out of control and distinct drugs are being declared logically equal. Upon closer inspection, one finds that in this massive equivalence class very different things -

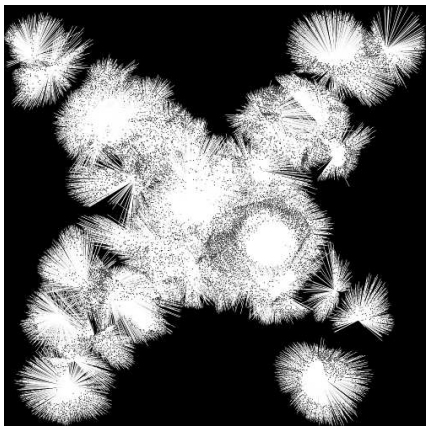


Figure 3: Visualization of top 40 *sameAs* equivalence classes.

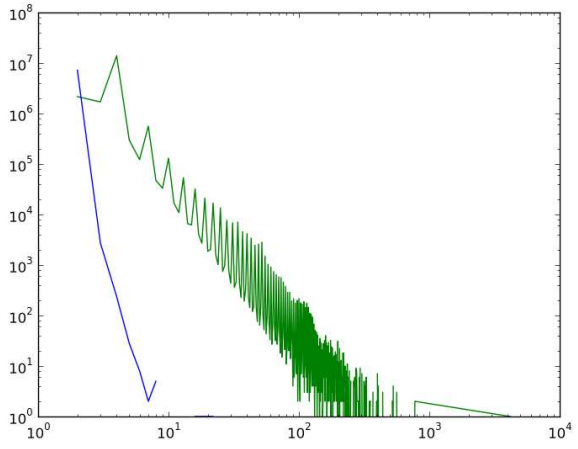


Figure 4: Size of Equivalence class vs. frequency: No inference (blue) and inferred classes (green) on a log-log axis.

which break OWL-style segregation semantics - have been connected by *sameAs*, such as 'class of drugs' and an 'individual instance of that drug applied' in open Linked Drug Data. And *uniprot* is not the only one; the *bio2rdf* Semantic Web triple-store produces a *sameAs* equivalence class of size 1,367. Even the ubiquitous DBPedia, a Semantic Web-export of Wikipedia, produces a closure of size 367 around a French verb conjugation, mixing up the class of French verbs in WordNet and the class of actions those verbs are about in Cyc. Interestingly enough, there are also occurrences of 'semantic' spam, with organizations such as the New York Times whole-scale copying *Freebase*-produced RDF to their own URIs (probably an innocent move), and the possibly less innocent copying of RDF data from diverse sources by *rdfabout.com*. 'Semantic' spam is simply the use of *sameAs* links and copying of RDF data to other data-sources in order to increase the visibility of one's own URI.

In total, the *sameAs* reasoning produced a total of 19,230,559 equivalence classes with an average content of 4.16 URIs. This is an approximate double in size over the average size of explicitly declared *sameAs* equivalence classes. However, this interpretation assumes the size of equivalence classes is normally distributed, which it is not, as illustrated in Figure 4 (log-log scale). The inferred *sameAs* distribution is clearly skewed towards smaller equivalence classes, but the mass of the distribution continues out towards the hundreds. Again, view Figure 3 for their visualisation. Due to this, after inference the size of the *sameAs* statements that violated the segregation semantics increased to 893,370, a small but statistically significant part of the Semantic Web. Upon closer inspection, the nature of this increase seems to be due to having some *sameAs* statements across segregated semantic roles being caught in a few of the larger equivalence classes.

4 Experiment

Another question would be, how many of these inferred *sameAs* are actually correct and obey the semantics of identity? A set of 250 *sameAs* triples (an identity link between two URIs) were chosen at random (with the chances of being chosen at random had been scaled down by the logarithm of the frequency of their domain name, in order to prevent a few major 'hubs' from dominating the entire data-set) from the previously described sample of the Semantic Web that violated the segregation semantics of *sameAs* (i.e. had an individual being *sameAs* with a class, for example). The closures of these 250 triples were generated, and from each of these about half (132) had at least two inferred identity links via transitivity. After removing 2 triples for training purposes, the remaining 130 triples were used for our experiment. Five judges (each Semantic Web experts and computer scientists) were chosen. A standard sample of properties and object (URI and literals) were retrieved from each URI and displayed to the user in a table. The judge was given a three-way forced choice, categorizing the statement as *Same*: clearly intended to identify the same single thing, *Related*: descriptions of fundamentally different things which none-the-less have have some important properties in common, or *Unrelated/Can't tell*: neither of the above, or you can't tell. We merged the 'unrelated' and 'can't tell' cases based on earlier work [Halpin *et al.*, 2010], which suggested that trying to make a distinction here was unreliable: being based mainly of personality, with some rushing to judge something different while others wanting more information.

There was also a checkbox next to each property, as well as one for each URI itself, and the judges were instructed to check the properties that were useful in forming their judgement. Overall, the judges had only slight agreement, reaching a Fleiss's κ of 0.06, which shows that the scheme is 'slightly' reliable. Looking at the distribution of judgements suggests more consistency than that, however. Figure 5 shows that more than half the time a 'consistent'judgement (unanimous, or only one disagreement) was made, and that consistent judgements of *same* and *different* are the most common. Using voting, we can determine that 52 (22%) are the same, 36 (28%) are related, 22 (17%) are different or can't tell, and 20 are ties (15%). Inspecting the useful properties triples judged as related, on an average a human judge chose

5.08 (S.D. 2.23) properties as important. This was primarily because some judges choose almost all properties to be relevant, while others would choose only a few.

n	j1	j2	j3	j4	j5
23	**2**	**3**	**3**	**3**	**3**
17	**3**	**3**	**3**	**3**	**3**
14	**1**	**1**	**1**	**1**	**1**
11	1	2	2	2	3
9	2	2	2	3	3
8	1	2	2	3	3
7	1	1	2	2	3
6	**2**	**2**	**2**	**2**	**2**
5	**1**	**2**	**2**	**2**	**2**
5	1	1	2	3	3
4	2	2	3	3	3
4	1	2	3	3	3
4	**1**	**1**	**1**	**1**	**2**
3	1	3	3	3	3
3	1	1	2	2	2
2	1	1	1	2	2
2	**1**	**1**	**1**	**1**	**3**
1	1	1	3	3	3
1	1	1	1	3	3
1	1	1	1	2	3

Figure 5: Distribution of Human Ratings per Judgment: 3 = same, 2 = related, 1 = not related or can't tell; 'Consistent' judgements are bold

5 Discussion

Surprisingly enough, the results with inference showed that the task itself with inference is significantly *hard* for individual experts in aggregate. In fact, in comparison with the 'moderate' κ values from previous studies done by others [Halpin *et al.*, 2010], inference seems to make experts less reliable. Simply, inference forces individual experts to confront edge-cases produced by the inference chains. However, groups of experts, despite the low aggregate reliability, could on individual cases discriminate via voting, and so could determine cases of mistaken identity via an 'identity leak' caused by a misplaced *sameAs*. In fact, compared to randomly selected triples without inference [Halpin *et al.*, 2010], the inferred triples have a slightly less chance of being correct (40% compared to 53% after inference), slightly more related triples (21% compared to 28% after inference), and a decrease in the amount of triples thought to be incorrect (29% compared to 17%). This decrease in the amount of incorrect is likely due to the triples being tied between two categories, which was 20%. When this is taken into account as likely "can't tell" it appears that the number of incorrect triples may go up as high as 37%. So in general, it seems that 'identity leaks' of misued *sameAs* statements cause the reliability of some identity statements to lower somewhat, but still remain broadly comparable with the kinds of human judgments not having inference produced.

What is also interesting is precisely when an approach based on the 'wisdom of crowds' can help disambiguate difficult results even when individual judges have difficulty. For example, a voting method could be deployed to solve *sameAs* ambiguities. Not all results are difficult: for geographical entities and individual people such as 'Lower Austria' and 'Johnny Bristol' agreement that they represented the same entity in the experimental data was generally high. Also, the results of incorrect inferences could also be easy to determine, such as an equivalence between a computer science researcher Tom Heath and a 'word-paving machine' or Le Flore County and the Marburg Virus in the experiment. Some things that were even related were relatively straightforward, such as the mereological relationship between the town of New Canaan and the South School therein or the historical relationship between Confucius and Confucianism. However, where there was more likely to be confusion was very closely related things such as the relationship between the concept of 'scoring an ace' in Cyc and the verb 'acing' in WordNet that denotes that concept, or the relationship between the Maranthi language and international code for that language. Then there were topics that were difficult due to background knowledge, such as whether Francisco Franco was related to Spain or not.

What should be done about this weaker notion of 'being related' that *sameAs* is erroneously used for? Indeed, one would be tempted to say that such a practice is simply wrong, and a virtually inference-free alternative such as *rdfs:seeAlso* or a *SKOS* construct should be used. However, a closer look at the experimental data shows that what is happening is that the notion of *related* is somewhat stronger: 'being related' means that two distinct things have a set of properties that show they have *some* relationship while nonetheless being distinct. While *identity* requires all properties to be shared (as there is only one thing being referred to), being related requires that either a core of essential properties be shared (but not all) amongst distinct entities or some other structured relationship be established. Yet it seems because some of these ontological properties needed by users are missing, users resort to using *sameAs* to connect two things simply because they are related and they are aware of the *sameAs* link. Note that our approach is distinct from any theory of ontological identity criteria, as the objects sharing the core constructs can remain distinct logically regardless of the core ontology [Guarino and Welty, 2000]. One could imagine inverse functional properties as given in OWL would allow to formalize this 'core,' but that would be a mistake as it declares two objects to be identical. Instead, another approach would be a class that provides property chains, so that a simple predicate could declare a dense structured network of properties that would state exact kind of relationships between distinct things. To address the latter, these class-specific property chains can then provide a way of stating these two items share enough properties to be substituted for each other in some (but not all!) cases as given by their class defintion, but nonetheless are distinct things. Again, the key is proper treatment of contexualized relatedness, which requires serious domain-specific treatment in the ontology. By virtue of being domain-specific the notion of 'being related' in all its degrees should be kept out of the semantics of identity.

6 Related Literature

The issue of identity has a long history in philosophy and ontology [Guarino and Welty, 2000], and more recently in knowledge representation and databases [Bhattacharya and Getoor, 2004]. To narrow the focus to the Semantic Web, this work directly follows the studies of Ding et al. [Ding *et al.*, 2010] and Halpin et al. [Halpin *et al.*, 2010]. However, there are important differences. The analysis of Halpin et al. presents some informal alternatives to the use of *sameAs*, but the only alternative tested was a more fine-grained identity scheme with no logical semantics, which could not be replicated even by experts using directly declared *sameAs* statements [Halpin *et al.*, 2010]. In contrast, we present a preliminary articulation of a logical alternative to *owl:sameAs*, focussing our studies on the reliability of identity in Linked Data on inferred triples. Our empirical analysis in general confirms some of the results of the more detailed empirical analysis of Ding et al. [Ding *et al.*, 2010], and variance between our results is explained by the fact that while Ding et al. used the Billion Triple Challenge, our analysis was performed over a 'live' copy of the Linked Data Cloud.

7 Conclusions

In order for a logic to be universal on the Web, it should be able to describe anything in any possible way, and still enable inference. By giving the users more freedom in modeling rather than less, one simultaneously encourages good modeling constraints, so that the language provides the necessary machinery to distinguish the world as it exists to the ontological engineer while not requiring particular metaphysical distinctions to be committed to *a priori*. An 'individual' in a metaphysical sense is one notion, whose merits can be debated; but 'individual' in the logical sense is quite another. The latter means simply 'a member of the universe of discourse' or 'within the scope of quantification'. Traditional 'good practices' typically get these two distinct notions confused, and use syntactic constraints arising from the latter to model the former. It's possible if someone is using *sameAs* to mean 'related to' (and our experiments show people do this sometimes), then they are actually in error. Yet then our experiment *also* showed for a significant minority they reasonably meant to declare equality across semantic roles. As *sameAs* as currently being used as a generic equality statement is in violation of the segregated lexical roles of OWL, it would make sense to create a different identity link that is based on a pollarding-based semantics.

Once logical identity is no longer mired in domain-specific ontology, good ontological practice also should require that ontological distinctions for the myriad of ways things can be 'related' be tested on humans to see if they can be reliably repeated. Previous studies by others have shown that fine-grained identity schemes make this difficult [Halpin *et al.*, 2010], and this study showed that using inference makes it *even* more difficult. Current experiments should be extended to take into account the reliability based on the size of the equivalence class, type-based reasoning over identity that utilizes all of Linked Data, and whether or not a suitable and more specific description (or replacement) of the 'related' category can be found that leads to higher agreement, and if an error-correction could help judge agreement. There seems to be hope: even though reliability on the results of inference is difficult, the wisdom of crowds seems to be able to discover and correct errors, and future experiments should be done over larger amounts of experts with more kinds of and quantity of identity links. The combination of consistent and unrestricted logical identity and richer use of domain-specific ontological constructs outside of *sameAs* could simultaneously put the links back into Linked Data and takes advantage of the semantics in the Semantic Web.

References

[Andrews, 2002] Peter Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Kluwer Academic Publishers, 2002.

[Bhattacharya and Getoor, 2004] Indrajit Bhattacharya and Lise Getoor. Iterative record linkage for cleaning and integration. In *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, DMKD '04, pages 11–18, New York, NY, USA, 2004. ACM.

[Brachman, 1983] Ronald Brachman. What IS-A is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10):30–36, 1983.

[Delugach, 2007] Harry Delugach. ISO Common Logic. Standard, ISO, 2007. http://cl.tamu.edu/ (Last accessed on March 8th 2008).

[Ding *et al.*, 2010] Li Ding, Joshua Shinavier, Zhenning Shangguan, and Deborah L. McGuinness. Sameas networks and beyond: Analyzing deployment status and implications of owl:sameAs in Linked Data. In *Proceedings of International Conference on the Semantic Web (ISWC)*, pages 145–160, Shanghai, China, 2010. Springer.

[Guarino and Welty, 2000] Nicola Guarino and Christopher Welty. Ontological analysis of taxonomic relationships. *Data and Knowledge Engineering*, 39:51–74, 2000.

[Halpin *et al.*, 2010] Harry Halpin, Patrick Hayes, Deborah L. McGuinness, and Henry S. Thompson. When owl:sameAs isn't the same: An analysis of identity in Linked Data. In *Proceedings of International Conference on the Semantic Web (ISWC)*, pages 166–181, Shanghai, China, 2010. Springer.

[Hayes, 2004] Patrick Hayes. RDF Semantics. Recommendation, W3C, 2004. http://www.w3.org/TR/rdf-mt/ (Last accessed Sept. 21st 2008).

[Motik *et al.*, 2009] B. Motik, P. F. Patel-Schneider, and B. Cuenca Grau. OWL 2 Web Ontology Language: Direct Semantics, 2009.

[Patel-Schneider *et al.*, 2004] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language: Semantics and Abstract Syntax , 2004.

It makes sense ...and reference: A bag-of-words model for meaning grounding and sharing in multi-agent systems

Theodosia Togia and Fiona McNeill

School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom
{ttogia, fmcneill}@inf.ed.ac.uk

Abstract

In a Semantic-Web-like multi-agent environment, ontology mismatch is inevitable: we can't realistically expect agents created at different times and places by different people to commit to one unchanging universal ontology. But what happens when agents attempt to interact? Can we benefit from ontology matching techniques to prevent communication breakdown? In this paper we present the Semantic Matcher, a system created to handle semantic heterogeneity (i.e. the use of different words for similar meanings), which is one of the most common mismatches across existing web-based ontologies. We claim that independently created agents with different ontologies are bound to miscommunicate because word meanings are private and not grounded in the real world. We propose the solution of achieving reference by means of sense, a common argument within Philosophy of Language. In our work, senses are simulated by bags of words, constructed for each lexical item known to a service-requesting agent, and for selected lexical items known to a service-providing agent in a given interaction. In the end we show that adopting this notion of 'sense' for words in formal ontologies allows for successful matching and creates the groundwork for a theory in which the vast amount of data available on the web can be used to facilitate agent communication by providing meaning for mismatched terms.

1 Introduction

In open multi-agent systems agents can request and provide services in order to automatically carry out complex tasks. These agents, called *requesters* and *providers* respectively[1], have ontologies which are quite often based on a variant of the BDI model (Beliefs, Desires, Intentions) (see [Wooldridge, 2009] for an introduction): they have *beliefs* (i.e. facts in their knowledge base), *desires* (i.e. goals; facts that they would like to make true, that is to add among their beliefs) and *intentions* (i.e. actions whose effects amount to them fulfilling their desires). In order to bring about a desired state of affairs, requesters need to follow a course of action, which is decided by planning.

A requester will typically have frequent and fleeting interactions with a potentially large number of providers that it has never talked to before. Therefore, interaction will be unpredictable before run-time and communication will usually be unsuccessful given the expected ontological mismatches. A solution to this problem was proposed by McNeill (see [McNeill and Bundy, 2007]), who built the Ontology Repair System (ORS), the first known system that attempts *ontology matching* on the fly; that is, during agent interaction. Similarly, the Semantic Matcher, described in the current paper, is - to our knowledge - the first example of on-line *semantic matching*. Since the Semantic Matcher is embedded into ORS, it is important to briefly describe the overall system before we move on to later sections.

The following section provides a brief overview of ORS. Section 3 explains the problem of *semantic mismatch in agent communication situations* and places it in the context of previous work. Section 4 discusses the philosophical and practical challenges associated with deriving the meaning of terms in ontologies. Section 5 explains our method of using a bag-of-words model for matching terms in formal ontologies on the fly and describes the architecture of our Semantic Matcher. Section 6 presents our initial results, Section 7 describes ways in which this work can be extended and Section 8 concludes the paper.

2 Ontology Repair System

ORS has been designed as a plug-in to a service requesting agent that can consult the system in order to make its ontology conform to the provider's view of the world. In a multi-agent environment ontology matching has to be fast, therefore ORS only 'fixes' parts of the requester's ontology that are relevant to a given encounter. In fact, dealing with incomplete information will be very common as providers might not be willing to reveal their full ontology to a service requesting agent or to any ORS-like system.

The lifecycle of a service requesting operation is as follows:

[1] In previous work (e.g. [McNeill and Bundy, 2007; Togia, 2010]) we have referred to them as Planning Agents and Service Providing Agents respectively.

1. The requester uses its ontology to form plans, which consist of actions that usually require interaction with other agents;
2. The requester begins to execute the plan by contacting a suitable service provider to perform the first action;
3. If this action succeeds, the requester continues with the rest of the plan. If not, it calls ORS to diagnose and fix the problem:
   - ORS analyses the communication so far to identify possible mismatches. This may lead to a precise diagnosis or a 'best guess' diagnosis, or it may prompt the requester to further question the provider in an attempt to locate the source of the problem[2];
   - Once a diagnosis has been made (in the worst case, this may just be marking some part of the requester's ontology as unusable in conjuction with the provider), ORS repairs the requester's ontology on this basis[3];
4. The requester replans with its improved ontology and starts again.

One principle behind the design of ORS is that representations should be expressive enough to allow for planning. But this also means that ORS has to deal with mismatches of many kinds (e.g. differences in predicate arity, differences in pre-conditions for actions etc). The original ORS can successfully handle some of these mismatches (see [McNeill and Bundy, 2007] for an overview) but always taking for granted that agents share the same words and understand them in the same way, thus ignoring semantic heterogeneity, unless it is hardcoded in the ontology. In this paper we demonstrate how this problem was handled with the creation of the Semantic Matcher, an ORS module that initiates negotiation of meaning between agents that have different vocabularies.

3 Semantic matching in multi-agent systems

Two agents trying to communicate on the web are very likely to have ontologies that differ both *structurally* and *semantically*[4]. Structural heterogeneity is dealt with by the original ORS under the simplifying assumption that words and their meanings are shared between the two agents. Once this assumption is removed, the system needs to align words before it performs any structural repairs.

[2]We assume these must be directed yes/no/don't know or instantiation questions rather than request for large parts of the provider's ontology to be revealed

[3]The current implementation assumes that the provider is always right and disposes of the requester's original ontology. We are currently investigating more sophisticated ways of using this information.

[4]Examples of structural mismatch are different number of arguments for equivalent predicates, different number of preconditions or effects for equivalent axioms, differences in quantifiers, implication and many others (see [Togia, 2010] for a detailed list). Semantic mismatches are differences in wording: for instance, the requester might represent UK currency as 'GB_pounds' while to the provider the same notion might be known as 'UKCurrencySterling'.

Semantic matching is necessary in situations where communication is hindered as a result of agents' lexicalising similar concepts in a different way. For example, imagine a requester that contacts a provider in order to buy a book. The provider is willing to perform this action on some conditions and starts by asking the requester if its 'credit_card_balance' is at least £5. The requester has never heard of this word before but has the same concept under the label 'moneyAvailable'. Our system has to perform semantic matching and suggest that 'moneyAvailable' should be changed into 'credit_card_balance' in order for the transaction to proceed.

The design of our system has benefited from previous work done in the area of Ontology Matching but, as we shall see, had to diverge significantly. One notable piece of reasearch is presented in Giunchiglia and Shvaiko's article *Semantic Matching* [Giunchiglia and Shvaiko, 2003], where the authors describe a process in which a 'match' operator takes as input two graph-like constructs (e.g. ontologies, web directory structures etc.) and produces as output a mapping between semantically equivalent nodes by looking at their 'parents', 'sisters' and 'children'. This enables nodes like 'Europe' and 'pictures' to be matched as equivalent (synonymous) if their ancestors are 'Images' and 'Europe' respectively, given that both nodes mean 'pictures of Europe'. Another work worth mentioning is a system built by Qu and his colleagues [Qu *et al.*, 2006], which provided the inspiration for the design of our on-line Semantic Matcher. This system computes correspondences between nodes of RDF graphs, which are Uniform Resource Identifiers (URIs), by constructing 'virtual documents' for each one of them. The term 'document' comes from Information Retrieval and means a multiset of words that represents a web page. In the system described, 'virtual documents' representing nodes are compared for similarity using the vector space model [Salton *et al.*, 1975]. The bags are generated from the tokenised URI (i.e. name of node) but also from 'neighbouring information'; that is, names of other nodes connected to it in the graph. Examples of many other node matching techniques can be found in [Euzenat and Shvaiko, 2007].

For our system, previous ontology matching methods could not be followed to the letter. Some reasons are: 1) they all deal with simple taxonomies or at best Description Logics. However, in our case ontologies are based on First-Order Logic, whose semi-decidability prevented us from attempting to traverse their structure and performing complex computations; 2) to our knowledge, no previous work exists that performs semantic matching on the fly, is which case computational time is a serious consideration; 3) our system has to deal with incomplete information because it is reasonable to assume that service providers will not be willing to reveal their full ontology to service requesting agents for privacy reasons. ORS only has access to the ontology of the requester, so it cannot align two ontologies but rather map unknown terms from the provider's ontology (revealed during interaction) to the best-matching terms from the requester's ontology. In fact, partial matching is practical given the time constraints of agent interaction.

Apart from the above implementation challenges, our sys-

tem also has to meet an important theoretical challenge: it has to tackle the root of the semantic mismatch problem. In the next section we claim that what inhibits meaning sharing among agents is the fact that words in agents' ontologies are not grounded to the objective world without human interpretation. We will demonstrate how our system enables ontology terms to make *reference* to real-world entities by creating *sense*, that is a 'mental representation' for each word in the agent's 'mind'.

4 Sense and reference in agent interaction

Humans use language in order to describe reality. Words, phrases and other grammatical constructions are only useful to the extent that they can make statements about the world around us. In fact, any definition that attempts to assign meaning ignoring the actual world is bound to cause infinite regress because words (and whatever is made up of words) are symbols and if they rely solely on other symbols for their meaning, at least some primitive symbols[5] have to be grounded. In current Philosophy of Language research linguistic meaning is usually accommodated in the framework of Truth-Conditional Semantics, where meaning is *reference* to the world: the meaning of a proper name is the individual named [Kripke, 1972], the meaning of a noun is a set of entities that it designates, the meaning of a verb is either a set of entities or a set of tuples of entities and the meaning of a sentence is a set of situations that make it true. When words are combined, the meaning of the construction is a combination (usually intersection) of the sets they point to.

Human communication is possible if the speakers involved have a shared understanding of the linguistic symbols used, that is if they all associate the symbols with the same entities or sets in the world. So, *reference* is essential for communication. However, restricting semantic assignment to reference only is not immune to problems because 1) the referrent might not exist[6], 2) words might have some extra meaning apart from their referrent[7], 3) reference can only be achieved by means of a mental representation, that is a meaning 'in the head'. The third reason is very important for our work and can be easily understood if we think about the meaning of nouns. For example, the meaning of 'cat' is the set of cats in the world. This is perfectly acceptable semantics but humans are unlikely to see or have seen all the cats in the world. So, when they see a cat that they have never seen before, they need some rules that help them decide if this entity belongs to the set of cats or not. These rules can be of many kinds, for example, individually necessary and jointly sufficient conditions encoded as a 'concept', that is mental representation. There are many theories that try to accommodate the structure of concepts. For an extensive overview the reader can be referred to [Laurence and Margolis, 1999].

The distinction between the 'mental' and the 'physical' facet of meaning was first made by Gotlob Frege [Frege, 1892], who called them *sense* and *reference* respectively. A few decades later, Carnap [Carnap, 1947] introduced the terms *intension* and *extension*, which are widely used in Philosophy of Language. A common way of visualising the fact that sense determines reference is Ogden and Richards' [Ogden and Richards, 1923] 'meaning triangle'. In Figure 4 the linguistic symbol "cat" on the left stands for the set of entities that the word extends to (i.e. all the cats in the world), but this relation is only indirect – hence the dotted line – because what mediates is the 'thought', that is the sense CAT of the word.

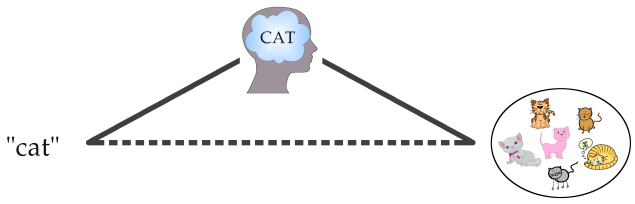


Figure 1: Meaning triangle for humans

The bottom line is that to achieve communication we need reference and to achieve reference we need sense. But what happens in the case of agents?

Agents use 'words' (predicates or constants) in order to form 'sentences' (formulas[8]). Their ontologies are written in languages that have clearly defined semantics with symbols, combinatorial rules and a universe of discourse which represents the real world. Sets of entities[9] from this universe give meaning to the predicates and ultimately to formulas. However, it is important to mention that the grounding of the ontology symbols to the actual world is performed by means of an interpretation function, which is where subjectivity enters the scene: ontology words have the meaning that humans give to them. In other words, it is the sense in the human head that determines reference to the real world. This is acceptable from a formal semantics point of view, but it does nothing to prevent communication breakdown between agents. If we want agents to communicate as if they were human, we need to make them fix the reference *themselves*. In other words, they need to be the ones that understand the meaning of their terms in order to share it with other agents. Therefore, we will need to create mental representations (senses) for every word in the agent's 'head' as shown in Figure 4.

As we will see in the next section, senses are represented as bags of words which are created automatically by aggregating information from various sources inside the ontology. Each word in the service requesting agent's ontology will have a bag associated with it, which will represent the agent's understanding of the word; that is, a sense which will help the agent determine the word's reference in the world. Compar-

[5] To be more precise, linguistic 'symbols' are actually signs, that is they are *conventionally* associated with their referrent.

[6] e.g. 'Santa Claus'

[7] e.g. If the person named could exhaust the meaning of a proper name then a sentence like 'Superman is Clark Kent' would be a tautology.

[8] i.e. predicates instantiated with constants and perhaps combined with other formulas in order to encode something that the agent believes to be true

[9] or sets of tuples of entities

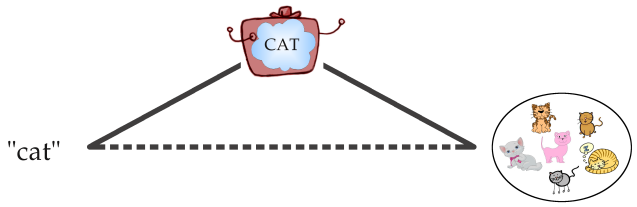


Figure 2: Meaning triangle for agents

ing senses from the requester's and the provider's ontology is a way to simulate negotiation of meaning and help the agents communicate.

The system has a theoretical justification, but it also solves previously unsolvable practical problems, namely online semantic matching with limited access to the service provider's ontology. The following section breaks it down into its components and explains the design desicions behind them.

5 Implementation

Our system is a search engine whose 'queries' and 'documents' are senses. As a thought experiment, imagine a 'Google of meanings', where we can input a mental representation and get as output a ranked list of similar mental representations.

Our model of semantic matching involves incorporating bags of words in formal ontologies, in order to achieve some *semantic grounding* - i.e. relations between the words in the ontology and the objective world. The Semantic Matcher in ORS is a search engine that tries to find the 'best match' for the provider's lexeme among the candidate lexemes in the requester's ontology. The provider's lexeme is expanded into a *bag of words* that make up its intensional meaning (i.e. sense). This bag serves as a *query* to the search engine[10]. The requester's candidate lexemes are also bags or words, acting as a collection of *documents* which will be ranked from the most to the least relevant after the search has been performed.

The architecture of the Semantic Matcher is illustrated in Figure 5. The Semantic Matcher as a search engine can be broken down into the following three components:

1. **Training the text acquisition model**
   The Text Acquisition Model is a set of databases created by our WordNet and SUMO parsers (see Figure 5), taking as input the lexical resource *Wordnet* [Gross and Miller, 2005], a database of *SUMO-WordNet mappings*[11] [Niles and Pease, 2003] and a collection of 645 *ontology files* (different versions of 38 ontologies that extend SUMO). The text acquisition model only makes use of Wordnet synonyms, ignoring hyponyms and instances, since this tends to disorient us from the lexeme's

[10]something like what we would submit as input to Google; not to be confused with a formal query written in languages such as SQL, providerRQL etc.

[11]http://sigmakee.cvs.sourceforge.net/viewvc/sigmakee/KBs/WordNetMappings/

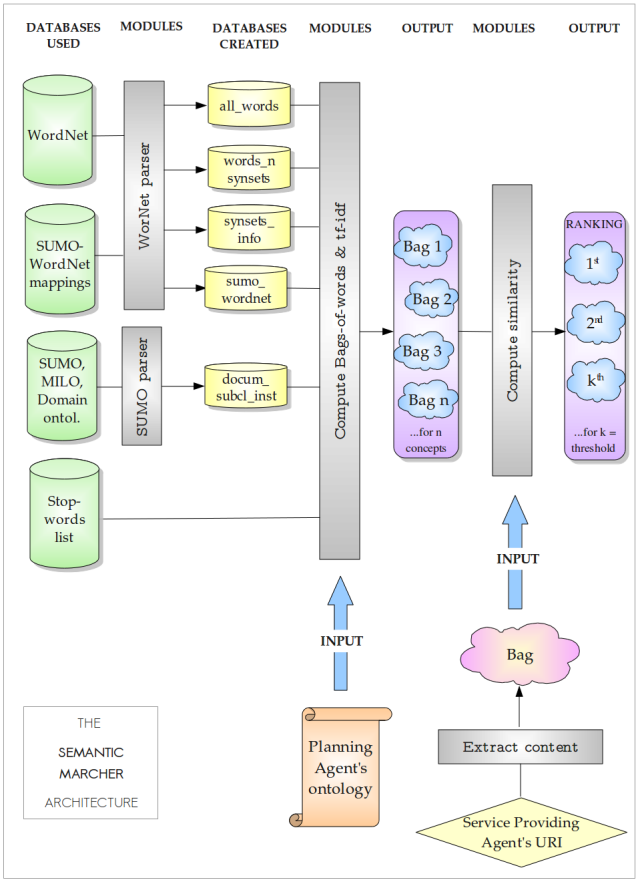


Figure 3: Architecture of the Semantic Matcher

meaning. The 645 ontology files provide us with natural language documentation and *superclass* or *type* information (i.e. what a lexeme is a subclass or an instance of) for some SUMO lexemes, which is necessary since SUMO is a modular ontology.

The above databases were parsed to create a set of new easily readable databases. WordNet and SUMO-WordNet mappings were scanned with regular expressions using the *WordNet parser* and the ontology files were parsed with the *SUMO parser* (which is able to deal with SUO-KIF [12] and is robust to errors such as unbalanced parentheses or quotes).

The text acquisition model that these databases form is a collection of resources (later read as lookup tables; dictionaries) that will determine what information can enter the bag for each lexeme in the requester's ontology. These files have been created once and will only have to be re-computed if the ontologies or WordNet version have to be updated. Their format is very easily processable so as to decrease the computational time of the next stage.

2. **Sense Creation and Term Weighting**
   This module takes as input the requester's ontology and

[12]the language that SUMO is written in

the databases created in the previous stage and returns a collection of bags of words, each one of which represents the intensional meaning ('sense') of a candidate lexeme. Each bag contains weighted words; that is, words with a co-efficient of importance.

We use the term **sense creation** to cover both *text acquisition*, where some text relevant to the lexeme's meaning is extracted from the databases, and *text transformation*, where this text is converted into words: the two subprocesses of creating the bags of words. Acquisition and transformation take place many times for every lexeme, since bags are filled incrementally.

The text aquired from the databases is put through a process of **tokenisation** (essentially, turned into words, e.g., camel case words separated into individual words), **stopping** (removal of semantically vacuous words) and **stemming** (reducing words to their stem so that it is easier to identify repetition of words, and therefore gauge their significance). Next, every word in every bag is assigned a weight, indicating how well that word represents the bag. This is done using the *tf-idf* (term frequency - inverse document frequency) weighting scheme [Robertson and Jones, 1976].

3. **Query Processing** Whilst the previous two stages can occur off-line, before interaction commences, this stage must be done online, during interaction, because it is only during interaction that the unknown lexemes are revealed. Once interaction failure occurs, ORS performs a diagnosis and, in the case that an unknown lexeme is identified, calls the semantic matcher to identify which of the candidate lexemes is the best match for this unknown lexeme.

6 Evaluation

The system was evaluated using different versions of the SUMO ontology and its sub-ontologies from the Sigmakee repository[13]. When terms are changed between SUMO versions (e.g. 'Corn' becomes 'Maize'), we have an objective way of measuring the performance of the matcher because we can safely regard terms and their renamings as synonyms and compare these pairings with our system's prediction. Initial results are encouraging, with 57% of correct matches chosen as the best by the system, 19% as the second-best. The outcome of this implementation cannot be compared against previous work because, to our knowledge, it is the first attempt to incorporate semantic matching in an agent communication system that has minimal access to one of the ontologies, but given the goals that ORS wants to achieve, we believe that these results are very encouraging. However, because of the sparse data available in the repository, it would be too early to jump to conclusions. We are currently evaluating this work more fully. In particular, we are investigating the possibility of performing large-scale evaluation using pairs of terms from manually matched ontologies, which will be compared to the pairs predicted by the system. We will not need to use matched first-order ontologies, which are after all hard to find. Only the pairs of terms will suffice. We hope that our full evaluation will confirm our initial positive results.

[13] http://sigmakee.cvs.sourceforge.net/viewvc/sigmakee/KBs/

7 Future goals

In the above sections we showed that a bag-of-words model can simulate word meaning in ontologies and facilitate communication between agents with semantically heterogeneous ontologies. However, we believe that perhaps the most important contribution of this work is that it paves the way for a new approach in ontology matching where senses are not just *created* from within the ontology but also *discovered* on the web and therefore take advantage of the vast amount of data available.

A bag-of-words model is structurally similar to a *broad folksonomy* [Vander-Wal, 2005]. Folksonomy is a bottom-up, not centrally controlled classification system in which structure emerges out of the practice of users labelling digital resources ('objects') with keywords ('tags'). Vander Wal distinguishes between broad folksonomies and narrow folksonomies [Vander-Wal, 2005]. The former are created when a particular object can be tagged by different people so the same tag can be assigned more than once. For example a Delicious[14] bookmark about, say, chocolate can have the word 'recipes' assigned to it 600 times, the word 'chocolate' 578 times, the word 'food' 423 times and so on. This pattern reveals some trends as to what vocabularies are generally considered appropriate to describe this resource. Narrow folksonomies, on the other hand, are formed in systems where one object can be labelled only by its author with distinct tags. For example, a Flickr[15] user can submit a photograph and annotate it with keywords such as 'surfing', 'waves', 'beach' and 'summer'. If it is made publicly available, it can be found by other users who search for photos about 'surfing' or 'waves' and so on.

Folksonomy is a way to annotate digital resources on the web for ease of retrieval and categorisation. Bringing folksonomy into formal ontologies is an attempt to annotate physical resources, that is things in the actual world. It should be noted that in our work so far folksonomies have been constructed from sources (e.g. comments in the ontology) that might reveal the ontology engineer's intended meaning. Therefore, they are not products of collaborative tagging. One of our future goals is to use tag data from online sources in order to help create senses in the agent's 'head'[16].

The idea of statistically approximating senses has been extensively discussed by Halpin (e.g. [Halpin, 2009]) in the context of semantic web architecture. In all these works the author argues that "the meaning of any expression, including URIs [equivalent to our 'lexemes' in ontologies], are grounded out not just in their formal truth values or referents, but in their linguistically-constructed 'sense'". Halpin's 'sense' is built on late-Wittgenstein's [Wittgenstein, 1953] idea of socially constructed meaning: the meaning a URI (or an ontology word in our case) is determined by users.

[14] http://www.delicious.com/

[15] http://www.flickr.com/

[16] For instance, tags that often co-occur with the tag 'cat' on Delicious might be used to simulate the meaning of the word 'cat'.

This is obvious in natural language, where words can change meanings through the years because they are used differently. To approximate this social aspect of meaning, Halpin and Lavrenko [Halpin and Lavrenko, 2010] propose relevance feedback as a way to collect data that indicates what meanings people assign to URIs. For example, in a semantic search engine, a query term (e.g. 'dessicated coconut') typed by a user can help construct the sense of the URI which the user found relevant.

This line of research is worth mentioning because it can point to further work within the area of formal ontologies and agent interaction. In particular, our work can be extended to take social meaning into account and throw more words into the 'bags' for ontology words. Since semantic search engines only work with URIs for now, we can use tags co-occuring with these words[17] on sites like Delicious, where the power of collaborative tagging can give us an intuition as to how people assign meaning to words.

8 Conclusions

This paper briefly introduced our work on statistically approximating the meaning of words in formal ontologies in order to perform matching on the fly, whenever the need becomes apparent. We believe these ideas could be a major step forward in the problem of ontology matching in an agent communication environment, and in providing symbol grounding for ontology terms. Furthermore, they can provide a framework for the design of matchers which exploit the large amount of tag data available on the web. For instance, one of our future goals is to extend the Semantic Matcher so that it takes *social meaning* (i.e. how users conceptualise words) into account. To achieve this, large databases of tags such as Delicious or perhaps users' input to semantic search engines will prove to be of central importance.

The Semantic Matcher was created on the basis of Information Retrieval principles, treating senses for words in the ontology as 'bags of words'. This was not just an engineering decision but also a proposal for incorporating unordered sets of words into the semantics of formal ontologies in order to achieve symbol grounding.

Full details of the theory on which this work is based, together with full descriptions of the implementation and evaluation of our original work can be found in [Togia, 2010].

References

[Carnap, 1947] R. Carnap. *Meaning and Necessity*. University of Chicago Press, Chicago, 1947.

[Euzenat and Shvaiko, 2007] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, Berlin, 2007.

[Frege, 1892] G. Frege. On sense and reference, 1892. In P. Ludlow, editor, *Readings in the Philosophy of Language*, pages 563-583, Cambridge, Mass, 1997.

[Giunchiglia and Shvaiko, 2003] F. Giunchiglia and P. Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18:265–280, 2003.

[Gross and Miller, 2005] G.A. Miller R. Beckwith C. Fellbaum D. Gross and K. Miller. Introduction to wordnet: An on-line lexical database, 2005. Online at http://courses.media.mit.edu/2002fall/mas962/MAS962/miller.pdf.

[Halpin and Lavrenko, 2010] H. Halpin and V. Lavrenko. A generative model of tagging, search and social networks. In *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line, April 26-27th, 2010*, Raleigh, NC, US, 2010.

[Halpin, 2009] H. Halpin. Social meaning on the web: From wittgenstein to search engines. In *Proceedings of the WebSci'09: Society On-Line*, 2009.

[Kripke, 1972] S. Kripke. *Naming and Necessity*. Harvard University Press, Cambridge, Mass, 1972.

[Laurence and Margolis, 1999] S. Laurence and E. Margolis. Concepts and cognitive science. In E. Margolis and S. Laurence, editors, *Concepts: Core Readings*, pages 2–81. MIT Press, Cambridge, Mass, 1999.

[McNeill and Bundy, 2007] F. McNeill and A. Bundy. Dynamic, automatic, first-order ontology repair by diagnosis of failed plan execution. *IJSWIS (International Journal on Semantic Web and Information Systems) special issue on Ontology Matching*, 3:1–35, 2007.

[Niles and Pease, 2003] I. Niles and A. Pease. Linking lexicons and ontologies: Mapping wordnet to the suggested upper merged ontology. In *Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE 03)*, Las Vegas, 2003.

[Ogden and Richards, 1923] C. K. Ogden and I. A. Richards. *The Meaning of Meaning*. Harcourt, Brace & World Inc., New York, 1923.

[Qu *et al.*, 2006] W. Qu, W. Hu, and G. Chen. Constructing virtual documents for ontology matching. In *Proceedings of the 15th International World Wide Web Conference*, pages 23–31, Edinburgh, 2006.

[Robertson and Jones, 1976] S. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[Salton *et al.*, 1975] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18:613–620, 1975.

[Togia, 2010] T. Togia. Automated ontology evolution: Semantic matching, 2010. MSc thesis, unpublished. Available on line at http://dream.inf.ed.ac.uk/projects/dor/.

[Vander-Wal, 2005] T. Vander-Wal. Explaining and showing broad and norrow folksonomies, 2005. Blog post, online at http://www.vanderwal.net/random/entrysel.php?blog=1635.

[Wittgenstein, 1953] L. Wittgenstein. Philosophical investigations, 1953. 2001.

[Wooldridge, 2009] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, New York, 2009.

[17] or with *part* of these words; e.g. 'cheesecake' instead of 'LemonCheesecakeDessert'

Towards "On the Go" Matching of Linked Open Data Ontologies*

Isabel F. Cruz
ADVIS Lab
Department of Computer Science
University of Illinois at Chicago
ifc@cs.uic.edu

Matteo Palmonari†
DISCo
University of Milan-Bicocca
palmonari@disco.unimib.it

Federico Caimi
ADVIS Lab
Department of Computer Science
University of Illinois at Chicago
fcaimi@cs.uic.edu

Cosmin Stroe
ADVIS Lab
Department of Computer Science
University of Illinois at Chicago
cstroe1@cs.uic.edu

Abstract

Creating links between the schemas of published datasets is a key part of the Linked Open Data (LOD) paradigm. The ability to discover these links "on the go" requires for ontology matching techniques to achieve sufficient precision and recall within acceptable execution times. In this paper we add two methods to the AgreementMaker ontology matching system (but that could be used together with other systems). A first method consists of discovering a set of high-quality equivalence mappings based on concept similarity and of inferring subclass mappings from that set. A second method is based on the adoption of background terminology that serves as a mediator. Sets of synonyms and a hierarchy of concepts are taken into account, as well as subclass relationships involving external concepts that are imported into the ontologies being matched. Experimental results show that our approach has good performance and improves precision on most matching tasks when compared with a leading LOD approach.

1 Introduction

Linked Open Data (LOD) extends the linked data paradigm, which identifies a set of best practices to publish and share data on the web [Bizer *et al.*, 2009], to open licensed data. In order to integrate information from different datasets, the capability of establishing "correct" links among data is crucial. Linked data together with their schemas are usually represented by web ontologies that are defined using semantic web languages such as RDFS and OWL. The problem of establishing links between datasets [Volz *et al.*, 2009; Bizer *et al.*, 2009] is therefore closely related to the problem of ontology matching that has been investigated in the semantic web and in the database communities [Rahm and Bernstein, 2001; Euzenat and Shvaiko, 2007].

Recently, the need to perform matching "on the go" has been addressed for dynamic applications [Besana and Robertson, 2005; Togia *et al.*, 2010]. For example, the need to match terms issued by an agent (sender) to terms in another agent (receiver) where such communication may require only a transitory agreement between parts of the agents' ontologies is one such scenario [Besana and Robertson, 2005]. To address this scenario, possible research directions that are mentioned include the modification of "static" ontology matching approaches to include filters that emphasize some mappings over others, thus increasing efficiency. However, none of these dynamic applications are in the LOD domain.

Matching a set of input data and several LOD ontologies "on the go" requires new techniques: (1) for ontology matching that achieve a good trade-off between quality of the mappings and efficiency, (2) for classifying sentences against the knowledge base built from the LOD cloud integration, and (3) for evolving individual ontologies or the knowledge base to adapt to a stream of incoming statements. In this paper we focus on the first aspect.

Ontology matching in the linked data context faces new challenges for it has been shown that several ontology matching systems perform poorly when it comes to matching LOD ontologies [Jain *et al.*, 2010]. One of the reasons is that many ontology matching systems are better tailored to discovering equivalence relations. This is clearly a drawback in matching LOD ontologies because only a few equivalence relations can be found among concepts in different ontologies. Therefore the capability to discover subclass relations becomes crucial when the number of links among LOD sources increases.

Prior work in matching ontologies in LOD has been performed by the BLOOMS system [Jain *et al.*, 2010]. This work has introduced a new matching approach based on searching Wikipedia pages related to ontology terms: the categories extracted from these pages are then organized into graphs and used to match the terms in the ontology. BLOOMS performs better than other systems that were not designed with the goal of matching LOD ontologies, but were instead designed to work in "classic" ontology matching settings based on equivalence mappings, such as those in the

*Research partially supported by NSF IIS Awards 0513553 and 0812258 and by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7061. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

†Work partially performed while visiting the ADVIS Lab at UIC.

Ontology Alignment Evaluation Initiative (OAEI) competition [Euzenat *et al.*, 2010]. However, both the accuracy and the efficiency obtained by BLOOMS in LOD settings are far lower than those obtained by "classic" systems when performing tasks for which they were designed. BLOOMS is also not a top performer in "classic" ontology matching.

In this paper we extend AgreementMaker[1] [Cruz *et al.*, 2009a; 2009b], an ontology matching system for ontologies expressed in a wide variety of languages (including XML, RDF, and OWL), which has obtained some of the best results in the OAEI competition [Cruz *et al.*, 2010], with the objective of testing its viability in the LOD domain. Therefore, in this paper we address the following two questions. How can a system like AgreementMaker be extended to handle mappings other than equivalence mappings? Can AgreementMaker achieve good accuracy and efficiency in the LOD domain?

To address the first question, we present two methods. The first method evaluates the lexical similarity between two ontology terms to discover equivalence relations. Afterwards we exploit the equivalence relations to infer a first set of subclass relations between the ontology terms. The second method discovers a second set of subclass relations by comparing the two ontologies with a third-party ontology representing background knowledge.

As for the second question, we show that our approach achieves better results in matching LOD ontologies than any other ontology matching system in terms of average precision (over a set of tasks). In terms of average recall our approach is the second best after the BLOOMS system. In addition, our approach is more efficient than BLOOMS and has the advantage of consisting of methods that can be integrated with an existing ontology matching system. Because of these results, AgreementMaker is currently the only system that achieves top performance both in the "classic" and LOD domains.

The paper is organized as follows. Related work is discussed in Section 2. The proposed methods to improve ontology matching in the LOD domain are described in Section 3. The experimental evaluation of the proposed approach, based on previously proposed reference alignments [Jain *et al.*, 2010] is discussed in Section 4. Concluding remarks end the paper in Section 5.

2 Related Work

We discuss related work whose main focus is on schema-level mappings (as opposed to instance-level mappings [Volz *et al.*, 2009]). We also mention an approach that makes use of background information and finally we describe three approaches that use the "on the go" paradigm.

The data fusion tool KnowFuss uses schema-level mappings to improve instance co-reference [Nikolov *et al.*, 2009]. It does not, however, address the discovery of schema-level mappings. An approach for ontology matching that uses schema-level (as well as instance-level) mappings has been proposed in the context of geospatial linked datasets [Parundekar *et al.*, 2010]. This approach infers mappings between ontology classes by analyzing qualitative spatial relations between instances in two datasets. It is therefore specific to the geospatial domain.

The BLOOMS system features a new approach that performs schema-level matching for LOD. It consists of searching Wikipedia pages related to ontology concepts: the categories extracted from these pages (using a Web service) are organized into trees and are compared to support matching between ontology concepts [Jain *et al.*, 2010]. To evaluate ontology matching for LOD, BLOOMS uses seven matching tasks and defines the *gold standard* or *reference alignment* for those tasks. Their tasks consider pairs of popular datasets (e.g., DBpedia, FOAF, GeoNames). They compare BLOOMS with well-known ontology matching systems such as RiMOM [Li *et al.*, 2009], S-Match [Giunchiglia *et al.*, 2007], and AROMA [David *et al.*, 2006] that have participated in the Ontology Alignment Evaluation Initiative (OAEI) [Euzenat *et al.*, 2010]. They show that BLOOMS easily outperforms those systems in the LOD domain. However, in the OAEI tasks, when compared with those systems, BLOOMS produces worse results when discovering equivalence mappings even if achieving better results when discovering subclass mappings [Euzenat *et al.*, 2010].

The SCARLET system introduces the idea of looking for clues in background ontologies to determine the meaning of concepts [Sabou *et al.*, 2008]. It uses logic-based rules to infer mappings whereas in this paper we determine the overlap between sets of concept descriptors. SCARLET has not been evaluated in the LOD domain.

There are a few ontology matching approaches that specifically address the "on the go" matching paradigm (but have not been tested in the LOD domain). In one of them, an interesting matching process takes place where mappings between terms are dynamically discovered during the interaction between autonomous agents [Besana and Robertson, 2005]. Because only relevant portions of the ontologies are mapped at a time, this approach is quite relevant to the problem of matching sentences against a set of available ontologies. Another approach introduces an "on-the-fly" method to match RDF triples to support semantic interoperability in smart spaces [Smirnov *et al.*, 2010]. A third approach proposes a framework where folksonomies are used as mediators in the ontology matching process [Togia *et al.*, 2010].

3 Matching LOD ontologies

Given a *source ontology* S and a *target ontology* T, a *mapping* is a triple $\langle c_S, c_T, r\rangle$ where c_S and c_T are concepts in S and T, respectively, and r is a semantic relation that holds between c_S and c_T.

A set of mappings is called an *alignment*. A *reference alignment* is an alignment found by experts, against which the accuracy of other alignments, as measured in terms of precision and recall, can be determined. In *ontology matching* one attempts to find as many accurate mappings as possible using *matching algorithms*, which we call *matchers*.

We consider three types of semantic relations: *subclass of*, *superclass of*, and *equivalence*, or, for short, *sub*, *sup*, and *equiv*, respectively. Given these relations we can define three types of mappings: $\langle c_S, c_T, sub\rangle$, meaning that c_S is a *subclass* of c_T, $\langle c_S, c_T, sup\rangle$ meaning that c_S is a *superclass*

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

of c_T, and $\langle c_S, c_T, equiv\rangle$ if and only if $\langle c_S, c_T, sub\rangle$ and $\langle c_S, c_T, sup\rangle$. In this case, c_S and c_T are *equivalent* classes.

Our approach to matching LOD ontologies is based on two methods:

Similarity-based mapping discovery. This method uses a *similarity metric* to compare the source and target ontologies: (1) to discover a set EQ_{sim} of equivalence mappings, and (2) to infer two sets of subclass and superclass mappings, respectively SUB_{sim} and SUP_{sim} from the equivalence mappings.

Mediator-based mapping discovery. This method compares the source and target ontologies to a third-party ontology, the *mediator ontology*, to discover two sets of subclass and superclass mappings, respectively SUB_{med} and SUP_{med}.

The alignment between S and T is defined as the union of the sets of mappings determined by the two discovery methods:

$$EQ_{sim} \cup SUB_{sim} \cup SUP_{sim} \cup SUB_{med} \cup SUP_{med}$$

Next, we first describe the algorithms that are at the core of each method and then we explain how these algorithms have been modified to improve the accuracy of the alignment.

3.1 Similarity-based Mapping Discovery

Equivalence mappings are discovered by evaluating a similarity value in the interval [0,1] between every pair $\langle c_S, c_T\rangle$ of source and target concepts, denoted $sim(c_S, c_T)$. The similarity value signifies the confidence with which we believe that the two concepts are semantically equivalent. A *matcher* computes the similarity values between all the possible pairs of concepts and stores the results in a similarity matrix. We use the *Advanced Similarity Matcher (ASM)* to compute the similarity matrix, which evaluates the lexical similarity between the two strings that represent both concepts [Cruz *et al.*, 2010]. For each pair of concepts and a threshold th, such that $sim(c_S, c_T) \geq th$, the mapping $\langle c_S, c_T, equiv\rangle$ is included in the set of equivalence mappings EQ_{sim}.

Starting from EQ_{sim}, we build SUB_{sim} and SUP_{sim} by considering subclasses and superclasses of the concepts c_S and c_T that appear in the mappings $\langle c_S, c_T, equiv\rangle \in EQ_{sim}$. We add to the set SUB_{sim} (respectively, SUP_{sim}) all the triples $\langle x_S, c_T, sub\rangle$ (respectively, $\langle c_S, x_T, sup\rangle$) such that x_S is a subclass of c_S (respectively, c_T is a subclass of x_T).

In the context of LOD, we note two important differences from ontologies that have been used for other tasks, such as those used in the OAEI competition [Euzenat *et al.*, 2010]. The first being a consequence of the use of subclass and superclass mappings that are derived from equivalence mappings. This means that a wrong equivalence mapping can propagate an error to all the derived mappings. For this reason, in the LOD domain we set a very high threshold, e.g., 0.95, while in several other domains thresholds in the range [0.6, 0.8] are usually adopted [Cruz *et al.*, 2010].

The second difference is that when compared with the OAEI datasets, LOD ontologies often use several concepts (e.g., *foaf:Person* in the Semantic Web Conference ontology) imported from other ontologies that need to be considered in the matching process. The above method is therefore refined by introducing a *global matching technique* for the external concepts used in an ontology.

For example, several external concepts used in LOD ontologies, such as *wgs84_pos:SpatialThing*—a concept referenced in the GeoNames ontology—are used across different ontologies; they could provide useful information in discovering additional mappings. That is, one could arrive at a mapping between *dbpedia:Person* and *wgs84_pos:SpatialThing* by knowing that *foaf:Person* has been defined as subclass of *wgs84_pos:SpatialThing* elsewhere.

We introduce the following technique. For each concept c_S in S that has been imported from an external ontology E, we search across several LOD ontologies (currently a restricted pool of well-known ontologies, such as DBpedia or FOAF, under the assumption that their concepts are shared in practice by a large community) for all concepts that are defined as subclasses of c_S and we match these concepts with the concepts of the target ontology using ASM. We perform the same for each concept c_T in T. In particular, if there is in some external ontology E a concept x_E such that x_E has been defined as subclass of c_S (respectively, c_T) and for some concept c_T (respectively, c_S), we have that $sim(x_E, c_T) \geq th$ (respectively, $sim(c_S, x_E) \geq th$) then $\langle c_S, c_T, sup\rangle \in SUP_{sim}$ (respectively, $\langle c_S, c_T, sub\rangle \in SUB_{sim}$).

3.2 Mediator-based Mapping Discovery

In order to discover the sets SUB_{med} and SUP_{med} we use an algorithm that takes as input the mediator ontology in addition to the source and target ontologies. The mediator ontology provides *background terminology* organized in a class hierarchy. Each concept is associated with a set of labels, which are synonyms of the concept. Conversely, a label can be associated with one or more concepts. In this paper our mediator ontology is WordNet, with the class hierarchy being the hyperonym hierarchy and the set of labels being the synsets.

As in Subsection 3.1, we compare every concept of the source ontology with every concept in the target ontology. However, this time we perform comparisons involving also the sets of labels and the sets of hyperonyms in the mediator ontology.

Step 1: Each concept in the source (respectively, target) gets associated with a set of concepts in the mediator ontology. This association is made through the concept labels: every time a label matches exactly a concept in the source (respectively, target) ontology, then that mediator concept becomes associated with the source (respectively, target) concept. Given a concept c, the set of mediator concepts associated with it is denoted BST_c (for *Background Synonym Terminology*).

Step 2: Each concept in the source (respectively, target) gets associated with a set of hyperonyms from the mediator ontology. This association is made through the previously built sets of synonyms. Given a concept c, we consider each concept in BST_c and extract its hyperonyms in the mediator ontology. Finally, we union all such sets thus obtaining a set for each concept c denoted BHT_c (for *Background Hyperonym Terminology*).

Step 3: We use the sets obtained in the previous two steps to build the sets of subclass and superclass mappings denoted respectively by SUB_{med} and SUP_{med} as follows: if there is a number $n \geq 1$ of synonyms of a concept c_S (respectively, c_T) that appear as hyperonyms of another concept c_T (respectively, c_S), then c_S is more general than c_T (respectively, c_T is more general than c_S), that is, $\langle c_S, c_T, sup\rangle \in SUP_{med}$ (respectively, $\langle c_S, c_T, sub\rangle \in SUB_{med}$). We experimentally set n to 2, which provides a good trade-off between precision and recall.

This three-step algorithm has potentially two shortcomings. The first one is that we seek labels in the mediator ontology that are similar to a concept in the source (or target) ontology. In certain cases a mediator label (e.g., bank) can be similar to an ontology concept (e.g., bank, meaning financial institution), but the mediator concept (e.g., mound) be semantically unrelated, thus decreasing precision. The second one is that it may not be possible to find a match among the mediator labels for ontology concepts that are noun phrases after tokenization (e.g., *Sports Event*), thus decreasing the algorithm's recall. Therefore, to improve precision and recall of the proposed algorithm, we refine Step 1 by performing "lightweight" semantic disambiguation that involves the superclasses and subclasses of the ontology concepts and the hyponyms and hyperonyms of the mediator concepts. We also add Step 4, which performs lexical analysis of noun phrases.

Step 1 (addition): When possible, we narrow the set BST_c into a subset $\overline{BST}_c$ by checking if some subclass (respectively, superclass) of c is related through synonyms with some hyponym (respectively, hyperonym) of some mediator concept x in BST_c, and, if this is the case, we include only that mediator concept x in $\overline{BST}_c$.

Step 4: In the case of noun phrases, we use a best effort approach that produces good results in practice. The concept denoted by a noun phrase is more specific than the concepts denoted by the individual names occurring in the noun phrases. Since in most cases the noun phrase narrows the scope of the *main noun* occurring in the phrase (e.g., *Sports Event* denotes a narrower concept than *Event*), and since in English the main noun is usually the last name occurring in the noun phrase, we use this knowledge to extract the main nouns and then attempt to find correspondences between them and the names of the concepts using ASM; based on these correspondences, we extrapolate subclass and superclass mappings. In particular, let $noun(c)$ be the main noun of a noun phrase denoting concept c. If $sim(noun(c_S), c_T)) \geq th$, then $\langle c_S, c_T, sub\rangle \in SUB_{med}$; if $sim(c_S, noun(c_T)) \geq th$, then $\langle c_S, c_T, sup\rangle \in SUP_{med}$.

4 Experimental Results

Table 1 lists the ontologies that we have used for our experiments, which are the same that were considered by the BLOOMS system[2] [Jain *et al.*, 2010], as no benchmark has been set otherwise for the LOD domain. The table shows the number of concepts in the ontologies and the number of external ontologies that they import.

[2]http://wiki.knoesis.org/index.php/BLOOMS.

Ontology	# Classes	# External ontologies
AKT Portal	169	1
BBC Program	100	2
DBpedia	257	0
FOAF	16	0
GeoNames	10	0
Music Ontology	123	8
Semantic Web Conference	172	0
SIOC	15	0

Table 1: Ontologies in the experimental dataset.

The evaluation settings consist of seven matching tasks, involving different types of comparisons. For example, Music Ontology and BBC Program are both related to entertainment, whereas some other comparisons involve general purpose ontologies, such as DBpedia.

Table 2 shows the comparison between the results obtained by AgreementMaker and the results previously obtained for the S-Match, AROMA, and BLOOMS ontology matching systems [Jain *et al.*, 2010]. We are omitting other results because they are not competitive [Jain *et al.*, 2010]. In addition, we have modified the reference alignment for the GeoNames–DBpedia matching task and marked such results clearly with "*", while reporting also on the results without this modification. We modified the reference alignment by including the subclass mapping between *dbpedia:Person* and *wgs84_pos:SpatialThing* and a set of mappings consistently derived from this one (namely mappings between all subclasses of *dbpedia:Person* and *wgs84_pos:SpatialThing*). This update makes sense in light of the fact that the subclass relation between *foaf:Person* and *wgs84_pos:SpatialThing* already appears in the reference alignment of several matching tasks. As can be seen in Table 2, our system achieves the best average precision (with or without the modification), while being the second best in average recall after BLOOMS. We comment next on the results obtained for each task.

Task 1. For the FOAF–DBpedia matching task, our system is the best one, both in precision and recall. This is because of our *global matching technique* described in Section 3.1, which finds correct mappings based on external ontologies and propagates those mappings through the subclasses of the involved concepts.

Task 2. For the GeoNames–DBpedia matching task, BLOOMS is not able to find mappings. This is because the GeoNames ontology has very little information in the ontology proper; instead, the actual categories are encoded in properties at the instance level. However, S-Match has perfect recall (100%), though precision is low (20%). Our global matching technique, which uses the external definition of concepts, is the reason why AgreementMaker outperforms all the other systems. With the modification to the reference alignment already mentioned, our results are extremely good, though comparison with the other systems was not possible in this case.

Task 3. For the Music Ontology–BBC Program matching task, the clear winner is BLOOMS, with AgreementMaker second. BLOOMS uses Wikipedia while we use WordNet, a generic background ontology. Wikipedia is very well suited

	S-Match		AROMA		BLOOMS		AgreementMaker	
Matching Task	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec
FOAF–DBpedia	0.11	0.40	0.33	0.04	0.67	0.73	0.72	0.80
GeoNames–DBpedia	0.23	1.00	0.00	0.00	0.00	0.00	0.26	0.68
GeoNames*–DBpedia*	-	-	-	-	-	-	0.88	0.88
Music Ontology–BBC Program	0.04	0.28	0.00	0.00	0.63	0.78	0.48	0.16
Music Ontology–DBpedia	0.08	0.30	0.45	0.01	0.39	0.62	0.62	0.40
Semantic Web Conference–AKT Portal	0.06	0.40	0.38	0.03	0.42	0.59	0.48	0.43
Semantic Web Conference–DBpedia	0.15	0.50	0.27	0.01	0.70	0.40	0.58	0.35
SIOC–FOAF	0.52	0.11	0.30	0.20	0.55	0.64	0.56	0.41
Average	0.17	0.43	0.25	0.04	0.48	0.54	0.53	0.46
Average*	-	-	-	-	-	-	0.62	0.49

Table 2: Comparison of AgreementMaker with other ontology matching systems.

Matching Task	Load	SB	MB	Total
FOAF–DBpedia	6.9	3.1	1.7	11.7
GeoNames–DBpedia	6.6	1.5	1.6	9.8
Music Ontology–BBC Program	16.0	3.7	4.7	24.4
Music Ontology–DBpedia	26.3	18.2	7.5	52.1
Semantic Web Conference–AKT Portal	3.5	2.1	2.8	8.3
Semantic Web Conference–DBpedia	7.9	8.1	2.4	18.5
SIOC–FOAF	0.1	0.2	1.7	2.0

Table 3: Execution times (in seconds) of the matching process (loading, similarity-based, mediator-based, and total).

for this kind of ontologies, because it covers their specific vocabulary.

Task 4. For the Music Ontology–DBpedia matching task, and in contrast with the previous task, our results are comparable with those of BLOOMS. We achieve higher precision, while BLOOMS achieves higher recall, in what constitutes a perfect swap between precision and recall, respectively 39% and 62% for BLOOMS and 62% and 40% for AgreementMaker. That is, our system presents only mappings for which it is very confident, thus favoring precision, while BLOOMS clearly favors recall. The next best system, S-Match, has reasonable recall (30%), albeit at the cost of very low precision (6%).

Task 5. For the Semantic Web Conference–AKT Portal matching task of scientific publications, BLOOMS favors recall again while AgreementMaker favors precision. S-Match favors recall at the cost of very low precision while Aroma favors precision at the cost of very low recall.

Task 6. For the Semantic Web Conference–DBpedia matching task, BLOOMS is the leader in both precision and recall, with AgreementMaker second. The conference domain is the same used in the OAEI competition. BLOOMS performs well because DBpedia is closely related to Wikipedia. S-Match has interesting recall (50%) but low precision (15%).

Task 7. For the SIOC–FOAF matching task, both general linguistic understanding and specific domain vocabulary are needed because SIOC is an ontology related to online communities. BLOOMS, S-Match, and AgreementMaker are close in precision (respectively, 55%, 52%, and 56%), but BLOOMS leads in recall because of its use of Wikipedia.

Table 3 shows the total execution times of AgreementMaker for the seven matching tasks as well as the times for the different subtasks, namely loading, mapping discovery using the similarity-based (SB) method and the mediator-based (MB) method. We note that the total time never exceeds one minute, even when large ontologies like the Music Ontology and DBpedia are being matched.

A complete comparison of all the systems in terms of execution time was not possible. However, we compared the performance of the Semantic Web Conference–AKT Portal matching task in BLOOMS and in AgreementMaker. While BLOOMS took 2 hours and 3 minutes, AgreementMaker performed the same task in only 8.3 seconds. We ran our experiments using an Intel Core2 Duo T7500 2.20GHz with 2GB RAM and Linux kernel 2.6.32-30 32 bits.

5 Conclusions

In this paper we described an efficient approach to schema-level ontology matching, which provides a step forward towards "on the go" ontology matching in the Linked Open Data (LOD) domain. We extended the AgreementMaker ontology matching system with two methods, one based on the evaluation of similarity between concepts and one based on the comparison of ontologies using background terminology.

The current leading system in the LOD domain is BLOOMS [Jain *et al.*, 2010], therefore this is the system with which to compare other systems. The work on BLOOMS has also led to the creation of seven tasks and respective reference alignments in the LOD domain, an important contribution to the field. While AgreementMaker outperforms BLOOMS in "classic" ontology matching settings [Euzenat *et al.*, 2010], in the LOD domain no system clearly outperforms the other. A general observation is that BLOOMS obtains better recall than AgreementMaker (in five of the seven tasks), while AgreementMaker obtains better precision than BLOOMS (in five of the seven tasks). However, given these results, AgreementMaker is the top performer in both "clas-

sic" and LOD ontology matching.

In the same way that in "classic" ontology matching no single strategy yields the best results in all cases, the same can be said for the LOD domain. For example, AgreementMaker needs to extend access to background information and in particular to Wikipedia for the matching of certain ontologies (e.g., Music Ontology and BBC Program). In spite of the good results obtained by BLOOMS and by AgreementMaker, precision and recall are lower than in the "classic" domains. The well-known trade-off between precision and recall appears to be more noticeable in the LOD domain.

The problem of ontology matching in the LOD domain is a necessary component of discovering meaning "on the go" in large heterogeneous data—this being the subject of this paper. Another important component consists of the ability to match sentences (for example, expressed in natural language or as RDF statements) against the large knowledge base that can be built from the LOD cloud—this will allow for answering queries, annotating text, and disambiguating terms.

References

[Besana and Robertson, 2005] Paolo Besana and David Robertson. Contexts in Dynamic Ontology Mapping. In *AAAI Workshop on Context and Ontology: Theory, Practice and Applications*, 2005.

[Bizer *et al.*, 2009] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data—The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22, 2009.

[Cruz *et al.*, 2009a] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. *PVLDB*, 2(2):1586–1589, 2009.

[Cruz *et al.*, 2009b] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods. In *ISWC International Workshop on Ontology Matching (OM)*, volume 551 of *CEUR Workshop Proceedings*, 2009.

[Cruz *et al.*, 2010] Isabel F. Cruz, Cosmin Stroe, Michele Caci, Federico Caimi, Matteo Palmonari, Flavio Palandri Antonelli, and Ulas C. Keles. Using AgreementMaker to Align Ontologies for OAEI 2010. In *ISWC International Workshop on Ontology Matching (OM)*, volume 689 of *CEUR Workshop Proceedings*, 2010.

[David *et al.*, 2006] Jérôme David, Fabrice Guillet, and Henri Briand. Matching directories and OWL ontologies with AROMA. In *International Conference on Information and Knowledge Management (CIKM)*, pages 830–831, 2006.

[Euzenat and Shvaiko, 2007] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag, Heidelberg (DE), 2007.

[Euzenat *et al.*, 2010] 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. In *ISWC International Workshop on Ontology Matching (OM)*, volume 689 of *CEUR Workshop Proceedings*, 2010.

[Giunchiglia *et al.*, 2007] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic Matching: Algorithms and Implementation. In *Journal on Data Semantics IX*, volume 4601 of *Lecture Notes in Computer Science*, pages 1–38. Springer, 2007.

[Jain *et al.*, 2010] Prateek Jain, Pascal Hitzler, Amit Sheth, Kunal Verma, and Peter Z. Yeh. Ontology Alignment for Linked Open Data. In *International Semantic Web Conference (ISWC)*, volume 6496 of *Lecture Notes in Computer Science*, pages 402–417. Springer, 2010.

[Li *et al.*, 2009] Juanzi Li, Jie Tang, Yi Li, and Qiong Luo. RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE Transactions on Data and Knowledge Engineering*, 21(8):1218–1232, 2009.

[Nikolov *et al.*, 2009] Andriy Nikolov, Victoria Uren, Enrico Motta, and Anne Roeck. Overcoming Schema Heterogeneity between Linked Semantic Repositories to Improve Coreference Resolution. In *Asian Semantic Web Conference (ASWC)*, volume 5926 of *Lecture Notes in Computer Science*, pages 332–346. Springer, 2009.

[Parundekar *et al.*, 2010] Rahul Parundekar, Craig Knoblock, and José Luis Ambite. Aligning Geospatial Ontologies on the Linked Data Web. In *Workshop On Linked Spatiotemporal Data in conjunction with the International Conference on Geographic Information Science*, 2010.

[Rahm and Bernstein, 2001] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[Sabou *et al.*, 2008] Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Exploring the Semantic Web as Background Knowledge for Ontology Matching. In *Journal on Data Semantics XI*, volume 5383 of *Lecture Notes in Computer Science*, pages 156–190. Springer, 2008.

[Smirnov *et al.*, 2010] Alexander V. Smirnov, Alexey Kashevnik, Nikolay Shilov, Sergey Balandin, Ian Oliver, and Sergey Boldyrev. On-the-Fly Ontology Matching in Smart Spaces: A Multi-model Approach. In *International Conference on Smart Spaces and Next Generation Wired/Wireless Networking (NEW2AN)*, volume 6294 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2010.

[Togia *et al.*, 2010] Theodosia Togia, Fiona McNeill, and Alan Bundy. Harnessing the Power of Folksonomies for Formal Ontology Matching On-the-Fly. In *ISWC International Workshop on Ontology Matching (OM)*, volume 689 of *CEUR Workshop Proceedings*, 2010.

[Volz *et al.*, 2009] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and Maintaining Links on the Web of Data. In *International Semantic Web Conference (ISWC)*, volume 5823 of *Lecture Notes in Computer Science*, pages 650–665. Springer, 2009.

Large Scale Semantic Matching: Agrovoc vs CABI

Aliaksandr Autayeu
University of Trento
Italy
autayeu@disi.unitn.it

Abstract

Achieving semantic interoperability is a difficult problem with a lot of challenges yet to address. Some of them include matching large-scale data sets, tackling the problem of missing background knowledge, evaluating large scale results, tuning the matching process and doing all of the above in a realistic setting with resource and time constraints. In this paper we report the results of a large-scale matching experiment performed on domain-specific resources: two agricultural thesauri. We share the experience concerning the above mentioned aspects of semantic matching, discuss the results, draw conclusions and outline perspective directions of future work.

1 Introduction

Semantic heterogeneity is a well-known problem and arises in many fields and applications. Many solutions have been proposed and many aspects of the problem are covered in papers (see surveys [Doan and Halevy, 2005; Choi *et al.*, 2006; Shvaiko and Euzenat, 2005]) and books [Euzenat and Shvaiko, 2007]. However, the problem is far from being considered solved. In fact, a definitive list of challenges yet to address in the field have been proposed [Shvaiko and Euzenat, 2008].

This paper considers an instance of semantic heterogeneity problem: matching two thesauri. The experiment described in this paper relates to several of the problems highlighted in [Shvaiko and Euzenat, 2008]. We test the feasibility of conducting a large scale mapping experiment. Large scale experiments were conducted before in [Giunchiglia *et al.*, 2009b] and [Giunchiglia *et al.*, 2009c], but this paper pushes the bar one order of magnitude higher: to the data sets with tens of thousand nodes. We test the importance of background knowledge with larger coverage for a domain-specific problem. Similarly to [Lauser *et al.*, 2008] we focus on agricultural domain, but on different aspects of the matching problem. This experiment can also serve as use case and as starting point for a creation of a new large-scale golden standard.

This paper describes an experiment in matching two thesauri: Agrovoc and CABI. This paper briefly describes the technique used for matching, the matched thesauri, the set up of the experiment, the experiment results and their evaluation. Finally, we make conclusions and propose future work.

2 Matching technique

In this experiment we use a semantic matching algorithm S-Match [Giunchiglia *et al.*, 2007] empowered with a minimal semantic matching algorithm [Giunchiglia *et al.*, 2009a]. The semantic matching algorithm implemented in the S-Match framework[1] [Giunchiglia *et al.*, 2010] consists of four steps. First, input sources, which consist of labels in natural language, are enriched with logical formulas using concepts drawn from a linguistic resource. Second, the formulas are contextualized to reflect the position of the concept in the initial data. During these two steps we use natural language processing techniques tuned to short text [Zaihrayeu *et al.*, 2007]. As a result we have transformed our input into lightweight ontologies [Giunchiglia and Zaihrayeu, 2009]. Third, all atomic concepts identified in the source and target thesaurus, are matched using background knowledge and other techniques, like string matching. Fourth, complex concepts from source and target thesaurus are matched using satisfiability solver and axioms collected on the third step.

The minimal semantic matching algorithm MinSMatch [Giunchiglia *et al.*, 2009a] exploits the fact that given the two source trees and the mapping, the mapping may contain some elements that may be derived from the others. Such elements are called redundant and can be excluded from the mapping, reducing it, often significantly. MinSMatch allows one to compute directly such a minimal mapping, saving computational efforts.

As a source of background knowledge for the first and the third steps we have used WordNet [Fellbaum, 1998]. In an attempt to alleviate the problem of lack of background knowledge [Giunchiglia *et al.*, 2006] we also use an extended version of WordNet, made available by the Stanford WordNet project[2]. WordNet provides a good coverage of the general part of the language, its slowly changing core. In turn, the extended version of WordNet contains about 4 times more concepts than the original WordNet 2.1. For example, we extracted 78 551 (WordNet: 19 075) multiwords and 1 271 588

[1] `http://semanticmatching.org/`
[2] `http://ai.stanford.edu/~rion/swn/`

(WordNet: 755 306) hypernym relations. The extended version is generated automatically and is 84% accurate [Snow *et al.*, 2006].

3 Thesauri

Agrovoc thesaurus and CABI thesaurus are two thesauri that contain terms from agricultural domain. They both cover the same domain of agriculture, which makes the matching task specific to agricultural domain. However, the knowledge base we use is generic and provides limited coverage of the agricultural domain. This factor alone is a serious challenge.

3.1 Agrovoc thesaurus

For our experiments we have used two versions of Agrovoc thesaurus. For the first two runs we have used hierarchical representation of the thesaurus from 10 August 2007[3] and for the second two runs — the database version of the thesaurus from 03 November 2009[4], which we converted into a compatible representation.

The version from 10 August 2007 was preprocessed and some terms[5] with a special mark-up were removed. The preprocessed version contains 35036 terms.

The version from 03 November 2009 is significantly improved in content and in structure as compared to the 2007 version. It was loaded directly from the database and converted into a hierarchical representation. Again, same terms with a special mark-up were removed. This resulted in 40 574 loaded terms. After several checks[6] on the thesauri structure and following terms removal, 32 884 terms remained. Finally, after taking into account multiple broader terms (BT) issue and decision to leave all BT links, including those leading to multiple inheritance, we have obtained a tree with 47 805 leaf-terms.

All the following figures describe the version of Agrovoc from 03 November 2009. Out of 32 884 terms 11 720 are single words, 21 161 are multiwords, that is, terms having more than one word. Most of them use space as a separator, although there are few terms using ",", "-" and "/" as a separator. Table 1 provides details on comparison of thesauri.

The final tree of 47 805 leaf-terms is a hierarchy containing a maximum of 14 levels. 76 BT relations were found to be redundant[7]. 1 207 terms have more than one BT relation. During this conversion we decided to leave these relations, effectively multiplying terms. Leaving a term in two places in a hierarchy increases the chances of it to be matched.

3.2 CABI thesaurus

For our experiments we used an "unversioned"[8] version of CABI thesaurus, which we converted into a compatible representation. The available version is not complete. As we found out during the conversion, there are many terms, referred to, but not present. We suppose that the chemical and taxonomical terms are missing.

After the same checks we did for Agrovoc, 18 200 terms were loaded. 57 BT relations were found to be redundant. 2 546 terms have more than one BT relation. During this conversion we decided to leave these relations, effectively multiplying terms. Leaving a term in two places in a hierarchy increases the chances of it being matched. The final tree of 29 172 leaf-terms is a hierarchy containing a maximum of 7 levels.

Out of 18 200 terms 6 842 are single words, 11 358 are multiwords. Most of them use space as a separator, although there are few terms using ",", "-" and "/" as a separator. Table 1 provides detailed thesauri comparison.

Fig. 1 shows an example CABI record in the original formatting.

Characteristic	CABI	Agrovoc
Tree leaves	29 172	47 805
Terms count	18 200	32 884
Single words	6 842	11 720
Multiwords	11 358	21 161
Hierarchy depth	7	14
multiple BT	2 546	1 207
redundant BT	57	76

Table 1: Thesauri comparison

```
cachexia
TNR: 18089
  BT: human diseases
  BT: nutritional disorders
  NT: wasting disease
  RT: anaemia
  RT: malnutrition
```

Figure 1: Sample CABI record

4 Experiments set up

We conducted a set of four experiments. Table 2 summarizes the parameters of our experiments. Notice for the fourth experiment we use S-Match because applying Min S-Match for flat structures brings no advantages. The following steps could be varied in the experiment.

First, a conversion from thesauri formats can be performed in a variety of ways. The most important parameters that influence the final result include: how to import relations, how to resolve ambiguities arising during conversion process and which knowledge base to use. We import only BT and NT (narrower term) relations for establishing a hierarchy of concepts. During the import we found a number of terms that have multiple broader terms. Such concepts could be placed in two (or more) places in the final hierarchy. Instead of removing BT relations until one remains, we left these terms under all their broader terms to increase matching chances.

[3] file ag_hierarchy_20070810_EN_UTF8.txt

[4] file agrovocnew.zip/agrovocnew 20091103 1627.sql

[5] like "Siluroidei (141XXXXXXX)"

[6] symmetric use of USE/UF, hierarchy redundancy, multiple broader terms

[7] if "A nt C" and "A nt B nt C" then relation "A nt C" is considered redundant

[8] file CABTHESNontaxNonchem.txt dated from 01 November 2009

Second, we can preserve the hierarchy of terms using BT and NT relations, or we can match term to term without considering the hierarchy (flat match).

Third, we can use different knowledge bases. We used two knowledge bases: WordNet version 2.1 and a 400.000 version of Stanford WordNet Project.

Fourth, we can choose between standard semantic matching and minimal semantic matching.

Fifth, the input sources were changed for the last two experiments mostly for technical reasons. According to experts, the structure and content of the 2009 version of Agrovoc has been improved a lot in comparison with the 2007 version. However, the 2009 version was not available during the first two experiments, but due to the amount of changes it was decided that it would be beneficial to proceed with a new version.

The matching consists of four steps: preprocessing (or concept at label computation), contextualization (or concept at node computation), element-level matching and structure-level matching. Below we will refer to some parameters and figures related to these stages of matching process.

Table 3 summarizes the quantitative results of the preprocessing stage. Using a general-purpose knowledge base such as WordNet on a domain-specific input results in a high amount of unrecognized words. For these words the matcher has to rely only on string-based matching techniques. Using extended WordNet from Stanford WordNet Project results in slightly improved coverage. Differences in coverage also depend on the differences in thesauri versions and on the conversion parameters.

5 Experimental results

Table 4 summarizes the results of the experiments. Using extended knowledge base on the element-level matching step increases mapping size. Relatively small amount of equivalence relations should be noted in the first experiments. In the fourth experiment, where BT/NT relations were not used for conversion and only plain terms were matched, the amount of discovered equivalence links is significantly larger.

In the latter case the algorithm was able to establish an equivalence (EQ) relation directly between two terms, while in the former cases it failed to establish the relation when intermediate terms were present in the hierarchy. We hypothesize that if the pairs of terms in question are the same, then this could be due to the lack of background knowledge. That is, in the former cases, a proper relation was not established between the intermediate terms, thus preventing the establishment of a relation between the end terms. Another option is that this is a consequence of using minimal match algorithm. Namely, the relation was established from one term to another, but either remained as a derived one and to be found in the maximized mapping (unlikely, because the amount of EQs in maximized mapping is roughly the same), or again, lack of background knowledge prevented the establishment of a relation between intermediate terms, in turn preventing the establishment of a relation between the end terms.

We report here both maximized and minimized mapping sizes due to their different purposes. The minimized mapping contains sort of "compressed information", leaving out many links, which could be derived. Therefore it is useful for exploration and validation as it minimizes the effort required [Maltese *et al.*, 2010]. If used with applications, however, the consuming application should be aware of semantics of the minimal mapping. The maximized mapping has traditional semantics and is ready for the immediate consumption by applications. The difference between minimized and maximized mapping sizes reaches 17 times.

Experiment	1	2	3	4
Memory used, Mb	2 082	1 718	2 982	1 104
Run time, hours	~10,5	~12	~27	~7,5

Table 5: Experiments run time and memory

For the fourth experiment, mapping maximization (or minimization) is not applicable due to the absence of hierarchy. For convenience only, we report the results together with minimized mappings.

Table 5 provides the runtime and the memory as reported by the Java virtual machine during the experiments. The experiments were executed on a laptop with Intel Core 2 Duo T9600 processor and 4G RAM under Windows 7 x64 using the 64-bit Java machine. The run times should be considered approximate, because, although S-Match currently runs single-threaded and there were two processors available with one available almost exclusively for JVM, the matching process was not the only process in the OS and other (lightweight) activities were conducted during the experiments.

The significantly larger run time of the 3rd experiment could be explained by the fact that JVM was using all available memory and parts of it were swapped, slowing down calculations.

6 Evaluation

In matching experiments, evaluation is not a simple task [Autayeu *et al.*, 2009]. For large matching tasks, such as this one, many of the more precise techniques based on a manual examination are not applicable due the size of the data. Other techniques for large data sets evaluation, such as reported in [Giunchiglia *et al.*, 2009c], exploit availability of instances to obtain the golden standard, and were not applicable in this case.

To evaluate the quality of links discovered by the matching algorithm, we need a golden standard to compare the mapping to. Such a mapping is usually created by an expert in the domain of the resources being matched and not only requires significant effort, but in many cases is impossible to create. Expert time is an extremely valuable resource and there is but a little of it available. This limits us in choosing an evaluation method.

We have chosen to evaluate a random sample of links from the mapping. We have used a sample of 200 randomly selected links. In the following we assume that the mapping being evaluated contains links with 4 relations: EQ (equivalence), DJ (disjointness), LG (less generality), MG (more

Parameter	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Agrovoc version	2007-08-10	2007-08-10	2009-11-03	2009-11-03
CABI version	2009-11-01	2009-11-01	2009-11-01	2009-11-01
Agrovoc terms-leaves	35 036	35 036	47 805	40 574
CABI terms-leaves	29 172	29 172	29 172	24 241
Conversion	hierarchy	hierarchy	hierarchy	terms only
Knowledge base	WordNet 2.1	SWN 400.000	SWN 400.000	SWN 400.000
Matching algorithm	Min S-Match	Min S-Match	Min S-Match	S-Match

Table 2: Experiments parameters

Parameter	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Knowledge base	WordNet 2.1	SWN 400.000	SWN 400.000	SWN 400.000
Unrecognized words in Agrovoc	16 080	14 934	15 725	17 301
Unrecognized words in CABI	18 235	16 890	17 537	18 890

Table 3: Preprocessing stage figures

	Experiment 1		Experiment 2		Experiment 3		Experiment 4	
Relations	min	max	min	max	min	max	min	max
EQ (equivalence)	3 698	3 564	3 603	3 468	3 009	3 407	28 457	N/A
DJ (disjointness)	125 439	3 811 923	124 648	3 777 493	184 564	8 304 315	199 836	N/A
MG (more general)	84 759	204 665	83 931	173 992	120 464	253 161	882 331	N/A
LG (less general)	218 579	1 262 700	223 140	123 6684	312 548	2 155 002	1 084 186	N/A
Total	432 475	5 282 852	435 322	5 191 637	620 585	10 715 885	2 194 810	N/A

Table 4: Experiments results

generality). The part of the mapping consisting of EQ, LG and MG links is called "positive" part. The rest, namely DJ links, is called "negative" part.

Traditionally, the most interesting part of the mapping is the positive part, with equivalences being the most desired links. However, one should consider the value of the mapping together with its intended use, keeping the target application in mind. For example, traditionally DJ relations are discarded as being of non interest. However, if the mapping is used for search purposes, DJ relations could be used to prune search space and therefore shorten search times. Similar reasoning could be done with less or more general links for narrowing or broadening search in a manner similar to how BT/NT relations work.

6.1 Methodology

Non-trivial nature of evaluating matching algorithm leads us to splitting the evaluation into two phases. The first phase is a "relaxed" evaluation of the matching results. This is a commonly used approach and it does not take into account the relation returned by the matcher. Only the presence of the relation is considered and it is treated like a similarity between two terms. This approach is applicable only to the positive part of the mapping.

The second phase of the evaluation is a "strict" evaluation of the results. It does take a link relation into account. The evaluation of the matching results in both cases was conducted by a single expert, actively involved in the development of the thesauri.

Let us illustrate the relaxed and strict evaluation with the example. Consider the link in Figure 2. This link is considered by an expert as valid under relaxed conditions. Definitely, there is a relation between Egypt and Suez Canal. However, this relation is not *less generality*. Therefore, the link is considered invalid under strict evaluation conditions.

We hypothesize that these kinds of approximations in link relations originate from various approximations in translation of semantics of input sources and semantics of knowledge bases, as well as approximations in source data sets and knowledge base data. For example, in the source data sets no explicit difference is made between partOf and isA relations. These two fundamental relations are often mixed in the same hierarchy, which leads to a less precise conversion to lightweight ontologies [Giunchiglia and Zaihrayeu, 2009].

6.2 Results

This section presents the results of the evaluation. Given the resources available, it was not possible to evaluate recall, therefore, we report only precision. Moreover, the sample we have been able to evaluate is extremely small compared to the mapping size, therefore these figures should be considered only as an approximation. How close they approximate exact figures (obtained by evaluating the complete mapping)

\Top\Africa\North Africa\Egypt < \Top\Middle East\Egypt\Suez Canal

Figure 2: Relaxed link example

and how big should be the sample to have a fair approximation is still a research issue [Autayeu *et al.*, 2009].

The following "facets" of the results are available. We differentiate between strict and relaxed evaluation, between overall, positive and negative parts of the mapping, between minimized and maximized mappings. Relaxed evaluation gives traditional precision measure and allows some degree of comparison with other matching systems. Strict evaluation gives a closer and more rigorous view of the results. Due to varying degree of interest of different mapping parts we provide overall, positive and negative parts precision separately. In addition to the reasons mentioned above, our recent report [Autayeu *et al.*, 2009] gives more details on why we differentiate minimized and maximized mappings precision.

Table 6 and Table 7 show precision figures for the relaxed and strict evaluation scenarios, respectively. The overall line contains precision for the complete set of links. Positive and negative lines show precision for the positive and negative parts of the mapping, respectively. "min" columns stand for minimized mapping precision and "max" columns stand for maximized mapping precision.

Experiment	**1**		**2**		**3**	**4**	
	min	max	min	max	N.A	min	max
Positive	**18.60**	14.08	10.49	14.61	N.A	06.98	N.A
Negative	97.18	52.15	94.74	99.13	N.A	100.00	N.A
Overall	25.81	31.45	21.74	21.74	N.A	34.28	N.A

Table 6: Relaxed precision for minimized and maximized mappings, %

Experiment	**1**		**2**		**3**	**4**	
	min	max	min	max	N.A	min	max
Positive	**05.43**	03.30	02.80	01.38	N.A	03.49	N.A
Negative	97.18	52.15	94.74	99.13	N.A	100.00	N.A
Overall	38.00	21.41	29.00	35.28	N.A	14.87	N.A

Table 7: Strict precision for minimized and maximized mappings, %

In the fourth experiment a significantly larger amount (28 457) of equivalences was discovered and we decided to evaluate them separately, again, using strict and relaxed scenarios. To make it more interesting, we removed trivial cases (8 654) of equal strings out of this set and evaluate only a sample out of the remaining 19 782 non-trivial equivalence links. Here we obtain 27% and 16.50% precision in relaxed and strict scenarios, respectively. The mapping produced in this experiment was selected for the future processing and use by the users.

7 Conclusions

The experiment allowed us to accomplish several goals. First, the matching algorithm was tested on a domain-specific data, showing that further research and improvements in acquiring and using domain specific background knowledge is needed. Second, it was stress-tested on large data sets. As far as we know, this is the largest matching task tried. Third, the experiment provides one more confirmation for a trend observed in other cases: many algorithms show high precision and recall values on small "toy" data sets, with a decrease in performance with the increasing data set size. It therefore further confirms the need for a large and diverse golden standards for better evaluations of matching algorithms, including their robustness aspects. Last, we created several versions of the mappings, some of which were used by the users.

Precision figures reported here allow one to use created mappings accordingly. High precision parts (such as negative parts) could already be used without reservations. Positive parts with sufficiently high precision, such as the equivalences from the fourth experiment, can be used as a basis for further manual validation.

8 Future work

This experiment confirmed and further outlined the following directions of the future work. The experiment results can be improved by using the natural language processing pipeline to improve translation into logics. Currently, there are some phenomena in thesauri that remain unaddressed by the current heuristics and are translated incorrectly. These include round brackets and their use for disambiguation as well as use of comma for term specification and qualification.

Another promising direction is more carefully importing knowledge from existing sources to augment current knowledge base with more relations and terms, achieving better domain coverage and domain specificity. This includes the knowledge from input sources (CABI and Agrovoc), as well as other thesauri covering the domain, such as the National Agricultural Library Thesaurus (NALT).

To analyse further the performance of the algorithm and ways to improve it, an extended analysis of current results would be beneficial. There are two ways to proceed with this task. First, for the first time we have a strict evaluation together with a relaxed one and we can analyse the reasons why the relation is not established correctly. Second, a set of false positives (links discovered, but judged to be incorrect) can be analysed to discover the errors in matching and ways to fix them. Third, a set of correct links (need to be established by an expert) would allow calculating recall and analysing the reason why the algorithm misses the links.

References

[Autayeu *et al.*, 2009] Aliaksandr Autayeu, Vincenzo Maltese, and Pierre Andrews. Recommendations for quali-

tative ontology matching evaluations. In Shvaiko et al. [2009].

[Choi *et al.*, 2006] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *SIGMOD Record*, 35(3):34–41, 2006.

[Doan and Halevy, 2005] Anhai Doan and Alon Y. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26:83–94, 2005.

[Euzenat and Shvaiko, 2007] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag New York, Inc., 2007.

[Fellbaum, 1998] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[Giunchiglia and Zaihrayeu, 2009] Fausto Giunchiglia and Ilya Zaihrayeu. Lightweight ontologies. In M. Tamer Ozsu and Ling Liu, editors, *EoDS*, pages 1613–1619. Springer, 2009.

[Giunchiglia *et al.*, 2006] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Discovering missing background knowledge in ontology matching. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *ECAI*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 382–386. IOS Press, 2006.

[Giunchiglia *et al.*, 2007] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: Algorithms and implementation. *J. Data Semantics*, 9:1–38, 2007.

[Giunchiglia *et al.*, 2009a] Fausto Giunchiglia, Vincenzo Maltese, and Aliaksandr Autayeu. Computing minimal mappings. In Shvaiko et al. [2009].

[Giunchiglia *et al.*, 2009b] Fausto Giunchiglia, Dagobert Soergel, Vincenzo Maltese, and Alessandro Bertacco. Mapping large-scale knowledge organization systems. In *ICSD*, 2009.

[Giunchiglia *et al.*, 2009c] Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, and Pavel Shvaiko. A large dataset for the evaluation of ontology matching systems. *The Knowledge Engineering Review Journal*, 24(2):137–157, 2009.

[Giunchiglia *et al.*, 2010] Fausto Giunchiglia, Aliaksandr Autayeu, and Juan Pane. S-Match: an open source framework for matching lightweight ontologies. *Semantic Web Journal*, 2010.

[Lauser *et al.*, 2008] Boris Lauser, Gudrun Johannsen, Caterina Caracciolo, Johannes Keizer, Willem Robert van Hage, and Philipp Mayr. Comparing human and automatic thesaurus mapping approaches in the agricultural domain. *CoRR*, abs/0808.2246, 2008.

[Maltese *et al.*, 2010] Vincenzo Maltese, Fausto Giunchiglia, and Aliaksandr Autayeu. Save up to 99% of your time in mapping validation. In Robert Meersman, Tharam S. Dillon, and Pilar Herrero, editors, *OTM Conferences (2)*, volume 6427 of *Lecture Notes in Computer Science*, pages 1044–1060. Springer, 2010.

[Shvaiko and Euzenat, 2005] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *JoDS*, 4:146–171, 2005.

[Shvaiko and Euzenat, 2008] Pavel Shvaiko and Jérôme Euzenat. Ten challenges for ontology matching. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (2)*, volume 5332 of *Lecture Notes in Computer Science*, pages 1164–1182. Springer, 2008.

[Shvaiko *et al.*, 2009] Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Natalya Fridman Noy, and Arnon Rosenthal, editors. *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*, volume 551 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[Snow *et al.*, 2006] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *ACL*. The Association for Computer Linguistics, 2006.

[Zaihrayeu *et al.*, 2007] Ilya Zaihrayeu, Lei Sun, Fausto Giunchiglia, Wei Pan, Qi Ju, Mingmin Chi, and Xuanjing Huang. From web directories to ontologies: Natural language processing challenges. In *ISWC/ASWC*, pages 623–636, 2007.

On the Investigation of Similarity Measures for Product Resolution

Krisztian Balog
Norwegian University of Science and Technology
Department of Computer and Information Science
Trondheim, Norway
krisztian.balog@idi.ntnu.no

Abstract

Entity resolution is an important information integration problem that has been looked at by a number of research communities. Resolution of a specific type of entity, products, however, is a largely unexplored area. This work sets out a series of first steps in that direction. We devise similarity measures for comparing products based on specific attributes. To estimate the discriminative power of these similarity functions we introduce a metric termed pairwise discriminative power. We report on experimental results using two purpose-built test sets, corresponding to two very different e-commerce segments: electronics and toys.

1 Introduction

E-commerce has become undeniably popular over the past decade. Almost any product imaginable can be purchased online: books, computers, clothing, furniture, flowers, and the list continues endlessly. The increase of e-commerce activity has had, and continues to have, a direct effect on the search industry. Product search systems are becoming indispensable tools for aiding users in making their selection decisions and finding the best offers in the ever-growing landscape of online web-shops. There are two main manifestations of product search systems: (i) verticals of major web search engines (such as Google product search[1] and Yahoo! shopping[2]) and (ii) price comparison websites (such as pricegrabber[3]). Both rely on information that retailers are required to make available: either as semantic markup on unstructured HTML documents (microdata, microformats, or RDFa) or as a data feed provided in some predefined structured format (e.g, XML or CSV) on a regular basis. Also of note that most price comparison sites use their own data schema; with the advent of microformats it is likely to change over time—but, even then, challenges remain.

Chief of these challenges is the resolution of products; webpages that represent the same product should be recognized. However, unique identifiers that could be used to join records (like EAN or SKU) are often absent or incompatible (as different sources use disparate means of identifying products). Further, the records representing the same product may have differing structure; one source might put the brand-name and the main product properties into a single name field (e.g,. *Apple iPod classic 160GB black*), while another organizes the same information into separate attributes: manufacturer, model, colour, and so on. The task of resolving entities has been looked at by a number of research communities, yet, we are not aware of any prior work focusing on the resolution of products in heterogeneous environments. This work sets out a series of first steps in that direction. Our main research objective is to devise measures for comparing products; first, on a per-attribute basis. These attribute-specific similarity functions can then be used as building blocks in more complex information integration scenarios.

Since no publicly available dataset exists for this purpose, we start off by constructing a test collection. As most prior research related to products focused on electronics (digital cameras, most prominently) we feel strongly about having test instances of another kind too. We build two test sets corresponding to two e-commerce segments that are of a very different nature: electronics and toys. The comparison of these two domains is woven throughout the whole paper. Product pages returned by a web search engine in response to an ad-hoc product query are used as candidate sets. Within each such set, products representing the same real-world entity are grouped manually into equivalence classes. Although there exist solutions for the automatic extraction of attributes from product pages, in the lack of training material, and, in the interest of avoiding any side-effects of using (potentially) erroneous data, we extracted product features manually.

To be able to measure how well similarity functions can can set apart entities within equivalence classes from other entities in the candidate set, we introduce a simple and intuitive metric termed *pairwise discriminative power* (PDP). This measure allows for the comparison of various similarity functions, even cross-domain, and also in cases where not all pairwise similarities can be computed, because of missing attributes. Finally, we introduce specific similarity functions for four main product features: name, price, manufacturer, and productID, and evaluate them using the proposed PDP measure. We find that the relative ordering of these four attributes is the same across the two domains.

[1] `http://www.google.com/products`
[2] `http://shopping.yahoo.com`
[3] `http://www.pricegrabber.com`

Our specific contributions are as follows: (1) we identify features specific to products and propose similarity functions tailored to them, (2) we introduce a metric for measuring the discriminative power of these similarity functions, and (3) we construct real-world test sets for two different e-commerce segments and present experimental results on them.

The remainder of this paper is organized as follows. We describe the procedure of constructing our test sets and their main properties in Section 2. Next, in Section 3, we introduce the PDP metric that we use for estimating the value of various similarity functions proposed in Section 4. We review related work in Section 5 and conclude with a summary of findings and an outline of future research directions in Section 6.

2 Data collection

Our main goals in constructing the data collection are twofold: (1) to gain insights into the differences between two e-commerce segments—electronics and toys—, and (2) to obtain a test set for evaluating product resolution techniques.

2.1 Approach

An exhaustive and statistically robust comparison would involve crawling and processing all web-shops from the segments in question. A reasonable compromise is to limit the set of web-shops considered to a certain region, for example, to a given country; this also removes the barriers of cross-language comparisons. Another rational solution to reducing efforts associated with collection building is to focus on prominent web-shops, i.e., the ones that attract many customers. Identifying all prominent web-shops of an e-commerce segment and (automatically) extracting their contents would be a stimulating research challenge on its own that goes beyond the scope of this work. As a middle ground, we chose to examine query-biased product sets: product pages returned by a web search engine in response to an ad-hoc product query. While this set is not comprehensive, it reasonably represents the most important web sources available for purchasing the given product. To eliminate potential discrepancies stemming from automatic product attribute extraction, we decided to extract features manually. This naturally imposed constraints on the number of products we were able to examine.

2.2 Collecting product pages

The process of constructing our two test sets is as follows. We issued a product query against a web search engine (in our experiments: Google) and collected the top N results returned (here: $N = 30$); search was restricted to a selected country (Hungary). Note that there is nothing language-dependent in our approach, the reasons for this choice were purely pragmatic (i.e., our access to data). Both test sets contain 30 queries issued by actual users. Electronics queries were selected from "popular searches" of a price comparator site. Toys queries were picked from the search logs of one of the biggest Hungarian online toy stores. Table 2 presents a few examples from the query sets (translated to English wherever needed for convenience). It is important to note that we only use these queries to collect products. The query itself is not used in the product comparison procedure.

Electronics	**Toys**
lenovo thinkpad edge 11	candamir
kingston microsd 8gb	eichhorn railway kit
samsung led tv 32"	eiffel tower puzzle 3d

Table 1: Sample queries.

Non-product pages (including results from product comparison sites) were removed manually from the result set. We also filtered out duplicates in cases when the exact same content was available under multiple URLs (as a result of various search engine optimization techniques). The remaining product pages were then manually inspected; we refer to them as the *candidate set*. Pages representing the same product were grouped together into *equivalence classes*. This manual clustering serves as our ground truth. Table 2 provides descriptive statistics.

	Electr.	**Toys**
#queries	25	30
#queries with >1 clusters	6	17
avg query length in characters	19.56	21.6
avg query length in terms	3.4	3.0
avg #product pages per query	4.6	5.7
min/max #product pages per query	1/8	1/11
avg #equivalence classes per query	1.24	2.66
min/max #equivalence classes per query	1/2	1/7
#different web-shops	46	38

Table 2: Statistics of the query sets.

Although we initially started with 30 queries for both segments, 5 of the electronics queries had no product pages returned within the top 30 results. Electronics queries are noticeably less ambiguous than toys queries, in terms of the number of different product clusters they exhibit. While query lengths do not differ markedly, electronics queries are inherently more specific as they always contain the manufacturer and most of the time indicate the specific model too. We also observe that in general there are less product pages returned for electronics queries. We noticed that top web search results are dominated by price comparison websites and pages presenting product reviews; these type of results were much less apparent so for toys. Future plans for improving the acquisition of product pages using web search engines are outlined in Section 6.

2.3 Extracting product attributes

We manually extracted the following attributes[4] from each product page: *name*, *price*, *manufacturer*, *productID*, *categories*, and *description*. Additionally, the followings were identified as common attributes among toy stores, thereby we collected them: *brand*, *age*, and *group* (boys, girls, or both). *Size* and *weight* information was also provided in some cases,

[4]We use product attributes and features interchangeably.

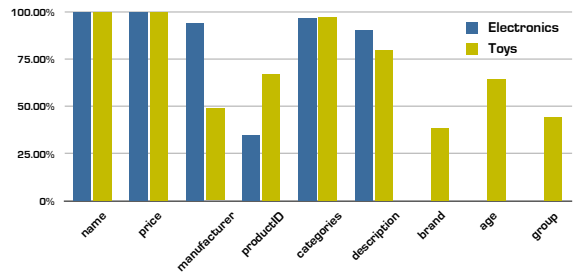


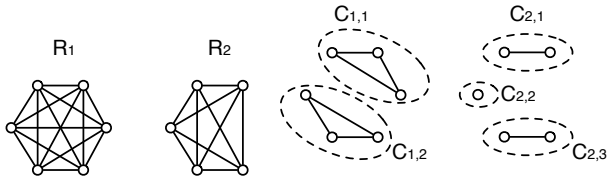
Figure 1: Statistics of product attributes. Bars show the fraction of product pages with the given attribute provided.

but for less than 15% of all product pages, therefore we did not include those features for further analysis. Figure 1 gives an overview of the presence of various attributes in product pages. Selected features are further discussed in Section 4.

3 Measuring the discriminative power of similarity functions

Our ultimate goal in this paper is to develop similarity functions specific to particular product attributes. To be able to measure how well these functions can set apart entities[5] within equivalence classes from other entities in the candidate set (i.e., the set of entities to be resolved), we introduce a metric called *pairwise discriminative power (PDP)*.

The intuition behind PDP is the following. Similarity functions cannot be compared based on the raw values they return, even if these values were normalized. Let us take an example with three entities in the candidate set, where A and B belong to the same equivalence class and C does not. If one similarity function returns $\mathrm{sim}_1(A,B) = 0.9$ and $\mathrm{sim}_1(A,C) = 0.8$, while another similarity function assigns $\mathrm{sim}_2(A,B) = 0.6$ and $\mathrm{sim}_2(A,C) = 0.3$, then sim_1 is less discriminative (hence, less useful) than sim_2 despite the higher values in absolute terms. We generalize this notion and define PDP as the ratio of the average intra-cluster similarity (i.e., similarity in equivalence classes—Figure 2(b)) over the average similarity of all pairwise comparisons in (all) the candidate sets (Figure 2(a)).



(a) Candidate sets (b) Equivalence classes

Figure 2: Pairwise similarities considered in PDP: (a) over candidate sets and (b) over equivalence classes.

[5]Throughout this section we will refer to products as entities, since the metric discussed here is a generic one, applicable to any type of entity in a resolution scenario.

Formally, let R_i denote the candidate sets that are to be resolved, where $|R_i|$ is the cardinality of the set. $C_{i,j}$ stands for equivalence classes within R_i, where $|C_{i,j}|$ indicate the number of entities in that class. $\mathrm{sim}(e_k, e_l) \in [0..1]$ is a function of similarity between entities e_k and e_l (note that it does not need to be symmetric). Note that we only consider entities to be elements of R_i and $C_{i,j}$ if sim is defined for them. The average of all pairwise similarities in all candidate sets (Figure 2(a)) is computed as follows:

$$\mathrm{avg}(\mathrm{sim}_R) = \frac{\sum_i \sum_{e_k \in R_i} \sum_{e_l \in R_i, k \neq l} \mathrm{sim}(e_k, e_l)}{\sum_i (|R_i| \cdot (|R_i| - 1))}. \quad (1)$$

Similarly, we calculate the average of all pairwise similarities in equivalence classes (Figure 2(b)):

$$\mathrm{avg}(\mathrm{sim}_C) = \frac{\sum_{i,j} \sum_{e_k \in C_{i,j}} \sum_{e_l \in C_{i,j}, k \neq l} \mathrm{sim}(e_k, e_l)}{\sum_{i,j} (|C_{i,j}| \cdot (|C_{i,j}| - 1))}. \quad (2)$$

Finally, PDP is defined as the ratio between the above two averages:

$$PDP = \frac{\mathrm{avg}(\mathrm{sim}_C)}{\mathrm{avg}(\mathrm{sim}_R)}. \quad (3)$$

A PDP value close to 1 indicates that entities in equivalence classes are not that different from those in the candidate set w.r.t. the particular similarity function used. Similarity metrics that exhibit high PDP values are more discriminative, therefore, more desired.

The attentive reader might wonder why we estimate PDP using micro-averaged statistics (i.e., by considering all pairwise similarities in all candidate sets), instead of macro-averaged ones (i.e., by calculating the average similarity values over a specific candidate set and the equivalence sets formed by elements of that set, then averaging over all candidate sets). The reason is that micro-averaging is expected to result in a more robust estimate in our case, as macro-averaging would bias the results by over-emphasizing candidate sets consisting of few products or few equivalence classes (which happens to be the case for our test sets, and especially for electronics). Nevertheless, it would be worth looking at both types of averaging for a data set with more products and equivalence classes.

4 Similarity measures for product attributes

In this section we investigate means of measuring pairwise similarities between products based on particular attributes. Our main question throughout this section is this: Which product attributes possess the most discriminative power? And, do findings hold for both the electronics and toys domains, or the two display different behaviour? We introduce specific similarity functions for four main product features (name, price, manufacturer, and productID) and evaluate them using the PDP measure proposed in the previous section. In this section we will only use "local" information, i.e., data that is extracted directly from the webpages of the two products that are subjects of the pairwise comparison, and do not consider any contextual information, global statistics, or other products in the candidate set.

4.1 Product name

Name is the most obvious candidate to serve as a basis for the comparison of entities. We employ various string similarity functions that fall into two main categories: character-based and term-based. We take the raw product names, without any cleaning or other transformation step (apart from lowercasing) and use implementations of the SimMetrics Java library[6].

Character-based distance functions. We consider five edit-distance based functions. The simple *Levenstein distance* assigns a unit cost to all edit operations. The *Monge-Elkan* approach is an example of a more complex, well-tuned distance function [Monge and Elkan, 1996]. Another character-based metric, which is based on the number and order of the common characters between two strings, is the *Jaro* metric [Jaro, 1995]. We also consider an adjustment of this due to Winkler [1999] that gives more favourable ratings to strings that share a common prefix. Additionally, we use the *q-grams* approach, which is typically used in approximate string matching by comparing sliding windows of length q over the characters of the strings [Gravano *et al.*, 2001a].

Term-based distance functions. We consider five vector-based approaches, where vectors' elements are terms. *Matching coefficient* simply counts the number of terms, on which both vectors are non-zero (i.e., a vector-based count of co-referent terms). *Dice's coefficient* is defined as twice the number of common terms in the compared strings divided by the total number of terms in both strings. *Overlap coefficient* is similar to the Dice coefficient, but considers two strings a full match if one is a subset of the other. The *Jaccard similarity* is computed as the number of shared terms over the number of all unique terms in both strings. *Cosine similarity* is a very common vector based distance metric, where the Euclidean cosine rule is used to determine similarity. The cosine similarity is often paired with other term-weighting approaches, such as TF-IDF (here, we refrain from using that as TF-IDF would require global term statistics).

Results. Table 3 presents the results. The first observation is that PDP scores are much higher for toys than for electronics. Another finding is that term-based distance functions outperform character-based ones in general, although the differences are minor for electronics. These results indicate that toys belonging to the same real-world entity are likely to be named similarly, while their names differ from that of other products in the candidate set. Specifically, toys' names within equivalence classes share a lot of common terms (as witnessed by the highest overall performance of the Jaccard similarity). On the other hand, electronic products retrieved for a particular query do not markedly differ in their naming.

4.2 Price

We use the following simple function to establish similarity based on price:

$$\mathrm{sim}_{price}(p_i, p_j) = \frac{\min(p_i.price, p_j.price)}{\max(p_i.price, p_j.price)}.$$

[6] http://staffwww.dcs.shef.ac.uk/people/S.Chapman/stringmetrics.html

Distance function	Electr.	Toys
Levenshtein distance	1.0343	1.2556
Monge-Elkan distance	1.0248	1.1456
Jaro distance	1.0184	1.1508
Jaro-Winkler distance	1.0167	1.1309
q-grams distance	1.0470	1.2961
Matching coefficient	1.0511	1.3387
Dice coefficient	1.0501	1.3181
Overlap coefficient	1.0491	1.2804
Jaccard similarity	1.0602	1.4190
Cosine similarity	1.0501	1.3123

Table 3: PDP scores for product name based similarity, using various string distance metrics.

This function always returns a value in $[0..1]$ and is 1 iff the two products have the exact same price. Price-based similarity results in a PDP score of 1.0306 for electronics and of 1.1882 for toys. As with names, we conclude that toys display a higher degree of diversity in terms of price.

4.3 Manufacturer

We applied the same distance metrics as in Section 4.1 for comparing strings holding the manufacturer's name. In the interest of space we only present the main findings. Character-based and term-based distance scores are very close to each other; this is an expected behaviour given that most manufacturer names consist of a single term. The best performing functions from the two main categories are the same as for names: q-grams (PDP=1.0276 for electronics and 1.0979 for toys) and Jaccard similarity (PDP=1.0275 for electronics and 1.1056 for toys).

4.4 Product ID

Unlike with other textual fields, we do not want to allow fuzzy matches for productIDs. As IDs of products from the same manufacturer are likely to differ only in a small number of characters, applying character-based string distance functions would do more harm than good. Therefore, we use a strict string matching method for this attribute and, consequently, take similarity to be a binary function.

4.5 Discussion

Table 4 reports PDP values for the four attributes discussed (using the best performing similarity metric where more options are available), as well as the coverage in the candidate set (%R) and within equivalence classes (%C). By coverage we mean the fraction of comparisons established (provided by the availability of the given attribute for both products) out of all possible pairwise comparisons. Despite the relatively small data set, the relative ordering of attributes by PDP values are consistent across the two domains. However, PDP scores are much smaller for electronics than for toys, in absolute terms; a possible explanation stems from the fact that most electronics queries resulted in a single product cluster.
Next, we turn to individual fields, out of which product name has already been discussed in Section 4.1. As to price, we have to note that while this field is available for all products,

	Electronics			Toys		
Attribute	%R	%C	PDP	%R	%C	PDP
Product name	100.0	100.0	1.0602	100.0	100.0	1.4190
Price	96.9	98.2	1.0306	89.6	99.1	1.1882
Manufacturer	95.3	94.6	1.0276	30.8	36.4	1.1056
ProductID	10.9	9.4	1.3333	45.0	44.2	1.9212

Table 4: Summary of the coverage and PDP values of attributes. %R and %C indicate the fraction of product pairs for which pairwise similarity could be established in candidate sets and in equivalence classes, respectively.

its content was not always successfully parsed as a numeric value—hence the imperfect coverage. Interestingly, the two segments substantially differ in the availability of manufacturer and productID attributes. While manufacturer name is available for most electronic products, this is the least distinctive feature of all. This is not surprising given that the queries used for collecting data also included the manufacturer, therefore most products in the candidate set are from the same company. Further, the identification of the specific model is often made available as part of the product name (e.g., "DeLonghi EC-8"). Conversely, toys are often labelled more creatively (e.g., "laser sword game") and keep the productID as a separate attribute. It is not unexpected that productID is the most discriminative feature of all.

5 Related work

Entity resolution (ER) is an important information integration problem that has been considered in many different disciplines (under many different names). In the Database community ER (also known as deduplication, record linkage, reference reconciliation, fuzzy grouping, or object consolidation) is the task of identifying records that represent the same real-world entity and reconciling them to obtain one record per entity. Most of the traditional, domain-independent approaches are variants of the statistical formulation introduced by [Fellegi and Sunter, 1969]; ER is viewed as a binary classification problem: given a vector of similarity scores between the attributes of two entities, classify it as "match" or "non-match." A separate match decision is made for each candidate pair. Additionally, transitive closure may be taken over the pairwise decisions. Attribute similarity measures are often based on approximate string-matching criteria [Gravano *et al.*, 2001b]. More sophisticated methods make use of domain-specific attribute similarity measures and often use adaptive approaches to learn them from the data [Bilenko and Mooney, 2003]. Utilizing the context of entities (i.e., references to other entities) brings in further performance improvements [Dong *et al.*, 2005; Bhattacharya and Getoor, 2004]. Another line of research focused on scaling ER to large databases by avoiding the quadratic number of pairwise comparisons [Baxter *et al.*, 2003; Benjelloun *et al.*, 2009]. Finally, there has been a great amount of work on non-pairwise ER, where match decisions for candidate pairs are not made independently; see [Singla and Domingos, 2006] for a generalization of these approaches using a unified framework based on Markov logic.

There are a number of problems related to *entity name resolution* within the Text Mining field. (Entity) name disambiguation (or name discrimination) is the task of grouping the representations of referents from the source documents so that each cluster contains all documents associated with each referent [Pedersen *et al.*, 2005]. Cross-document coreference resolution is the task of determining whether an entity name (most often of type person, organization, or location) discussed in a number of documents refers to the same entity or not [Gooi and Allan, 2004]. Essentially, name disambiguation and cross document co-reference resolution are two sides of the same coin. A great deal of work has focused specifically on the resolution of person names [Wan *et al.*, 2005; Artiles *et al.*, 2005; Balog *et al.*, 2009].

Much of the research related to products focused on *mining reviews* for opinion and sentiment classification [Dave *et al.*, 2003; Cui *et al.*, 2006], summarization [Hu and Liu, 2004; Meng and Wang, 2009], and discovery and extraction of product attributes [Ghani *et al.*, 2006; Raju *et al.*, 2009]. There is not much work published on *product search*. [Nurmi *et al.*, 2008] introduce a grocery retrieval system that maps shopping lists written in natural language into actual products in a grocery store. [Pu *et al.*, 2008] develop a framework for evaluating general product search and recommender systems.

6 Conclusions and Future work

In this paper we addressed the task of product resolution in a heterogeneous environment: product pages from online stores. We conducted a data-driven exploration of using various product attributes as bases of pairwise product similarity comparisons. Further, we introduced a novel *pairwise discriminative power* (PDP) measure and performed an experimental evaluation of the attribute-specific similarity functions against the PDP metric. Our study focused on two different e-commerce segments: electronics and toys. They markedly differ in the usage of product attributes, still, our findings concerning the attribute-specific similarity functions seem to be consistent across the two.

Our work is a first step towards the ambitious task of automatic product (web)page resolution, and as such, has limitations. So far we only focused on pairwise similarity functions for a limited number of attributes (namely: name, price, manufacturer, and productID). These pairwise similarity functions are core ingredients to be used as building blocks in more sophisticated product comparison methods. There are two product-specific features that proved too complex to fit within this study: categories and description. Most webshops (as shown in Figure 1) organize their products into a multi-level categorization; these categories, however, need to be aligned. An obvious starting point is to compare term-based similarity metrics and techniques for ontology alignment [Omelayenko, 2000]. As for product descriptions, our data set revealed that these are used very differently in the two segments. For most electronics products, description entails a list of property-value pairs, while for toys, it is most often a short blurb, targeted to appeal to the customer. The two call for very different treatment.

The proposed PDP measure is a simple and intuitive one that is capable of estimating the value of similarity functions.

We recognize though that the current study lacks the validation of this measure; it remains to be tested whether higher PDP values indeed correspond to better performance when the actual clustering of products is performed.

We acknowledge that the size of our data set does not allow for a statistically robust comparison. We plan to repeat these experiments with a larger test set. We demonstrated that finding product pages using web search engines is a viable method. Initial results suggest that a more complete candidate set could be achieved by considering product comparison sites too for crawling and content extraction. We intentionally refrained from any modifications to the queries and used them unedited; as a result of that 1 out of 6 web search results were product pages, on average. This ratio could easily be improved by issuing more targeted queries. A cheap way of achieving that is to append the currency to the query, thereby excluding pages that do not contain a price. More advanced techniques might involve various reformulations of the query, e.g., by using blind relevance feedback techniques.

Acknowledgments

This work has been supported by the COMIDOR project, funded by the Norwegian Research Council.

References

[Artiles *et al.*, 2005] J. Artiles, J. Gonzalo, and F. Verdejo. A testbed for people searching strategies in the www. In *28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 569–570, 2005.

[Balog *et al.*, 2009] K. Balog, L. Azzopardi, and M. de Rijke. Resolving person names in web people search. In *Weaving Services and People on the World Wide Web*, pages 301–323. Springer, 2009.

[Baxter *et al.*, 2003] R. Baxter, P. Christen, and T. Churches. A Comparison of fast blocking methods for record linkage. In *ACM SIGKDD'03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[Benjelloun *et al.*, 2009] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *The VLDB Journal*, 18:255–276, January 2009.

[Bhattacharya and Getoor, 2004] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 11–18, 2004.

[Bilenko and Mooney, 2003] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48, 2003.

[Cui *et al.*, 2006] H. Cui, V. Mittal, and M. Datar. Comparative experiments on sentiment classification for online product reviews. In *21st national conference on Artificial intelligence - Volume 2*, pages 1265–1270, 2006.

[Dave *et al.*, 2003] K. Dave, S. Lawrence, and D. M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *12th international conference on World Wide Web*, pages 519–528, 2003.

[Dong *et al.*, 2005] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *2005 ACM SIGMOD international conference on Management of data*, pages 85–96, 2005.

[Fellegi and Sunter, 1969] I. P. Fellegi and A. B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[Ghani *et al.*, 2006] R. Ghani, K. Probst, Y. Liu, M. Krema, and A. Fano. Text mining for product attribute extraction. *SIGKDD Explor. Newsl.*, 8:41–48, June 2006.

[Gooi and Allan, 2004] C. H. Gooi and J. Allan. Cross-document coreference on a large scale corpus. In *Human Language Technology/North American chapter of Association for Computational Linguistics annual meeting*, pages 9–16, 2004.

[Gravano *et al.*, 2001a] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, and D. Srivastava. Using q-grams in a dbms for approximate string processing. *IEEE Data Engineering Bulletin*, 24:28–34, 2001.

[Gravano *et al.*, 2001b] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *27th International Conference on Very Large Data Bases*, pages 491–500, 2001.

[Hu and Liu, 2004] M. Hu and B. Liu. Mining and summarizing customer reviews. In *10th ACM SIGKDD intl. conf. on Knowledge discovery and data mining*, pages 168–177, 2004.

[Jaro, 1995] M. A. Jaro. Probabilistic Linkage of Large Public Health Data Files. *Statistics in Medicine*, 14:491–498, 1995.

[Meng and Wang, 2009] X. Meng and H. Wang. Mining user reviews: from specification to summarization. In *ACL-IJCNLP 2009 Conference Short Papers*, pages 177–180, 2009.

[Monge and Elkan, 1996] A. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *2nd International Conference on Knowledge Discovery and Data Mining*, pages 267–270, 1996.

[Nurmi *et al.*, 2008] P. Nurmi, E. Lagerspetz, W. Buntine, P. Floréen, and J. Kukkonen. Product retrieval for grocery stores. In *31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 781–782, 2008.

[Omelayenko, 2000] B. Omelayenko. Integration of product ontologies for b2b marketplaces: a preview. *SIGecom Exch.*, 2:19–25, December 2000.

[Pedersen *et al.*, 2005] T. Pedersen, A. Purandare, and A. Kulkarni. Name discrimination by clustering similar contexts. In *Comp. Linguistics and Intelligent Text Proc.*, pages 226–237, 2005.

[Pu *et al.*, 2008] P. Pu, L. Chen, and P. Kumar. Evaluating product search and recommender systems for e-commerce environments. *Electronic Commerce Research*, 8:1–27, 2008.

[Raju *et al.*, 2009] S. Raju, P. Pingali, and V. Varma. An unsupervised approach to product attribute extraction. In *31th European Conference on IR Research on Advances in Information Retrieval*, pages 796–800, 2009.

[Singla and Domingos, 2006] P. Singla and P. Domingos. Entity resolution with markov logic. In *6th International Conference on Data Mining*, pages 572–582, 2006.

[Wan *et al.*, 2005] X. Wan, J. Gao, M. Li, and B. Ding. Person resolution in person search results: Webhawk. In *14th ACM international conference on Information and knowledge management*, pages 163–170, 2005.

[Winkler, 1999] W. E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.

Ranking Query Results from Linked Open Data Using a Simple Cognitive Heuristic

Arjon Buikstra*, Hansjörg Neth†, Lael Schooler†, Annette ten Teije*, Frank van Harmelen*
*Dept. of Computer Science, VU University Amsterdam
†Max Planck Institute for Human Development, Berlin

Abstract

We address the problem how to select the correct answers to a query from among the partially incorrect answer sets that result from querying the Web of Data.

Our hypothesis is that cognitively inspired similarity measures can be exploited to filter the correct answers from the full set of answers. These measures are extremely simple and efficient when compared to those proposed in the literature, while still producing good results.

We validate this hypothesis by comparing the performance of our heuristic to human-level performance on a benchmark of queries to Linked Open Data resources. In our experiment, the cognitively inspired similarity heuristic scored within 10% of human performance. This is surprising given the fact that our heuristic is extremely simple and efficient when compared to those proposed in the literature.

A secondary contribution of this work is a freely available benchmark of 47 queries (in both natural language and SPARQL) plus gold standard human answers for each of these and 1896 SPARQL answers that are human-ranked for their quality.

1 Introduction

The Web of Data has grown to tens of billions of statements. Just like the traditional Web, the Web of Data will always be a messy place, containing much correct, but also much incorrect data. Altough there has been surprisingly little structured research on this topic, anecdotal evidence shows that even the highest rated and most central datasets on the Web of Data—such as DBPedia and Freebase—contain factually incorrect and even nonsensical assertions. Consider the following results from some of the benchmark queries that we will discuss later, when executed against a combination of DBPedia, Geonames and Freebase:

- "AmeriCredit" is not an American car manufacturer (instead, it is a financial company owned by General Motors to help customers finance their cars).
- "Richard Bass" is not one of the highest summits on the seven contintents (instead, he was the first mountaineer that climbed all of them).
- "Cosima" is not a Nobel Prize for Literature Laureate (instead, it is a novel written by Grazia Deledda, who received the 1926 Nobel Prize for Literature).
- "Stig Anderson" was not one of the members of ABBA (instead, he was their manager).

These examples (which are just a few of many) illustrate *the central problem* that we tackle in this paper:

> Given a query to the Web of Data and the resulting answer set, how to separate the correct from the incorrect answers.

For well over a decade, influential cognitive scientists have been proposing the notion of *fast and frugal heuristics*: heuristics that are surprisingly simple (sometimes even seemingly naive), but that on closer inspection perform very well on complex cognitive tasks. Their findings have shown convincingly that such simple heuristics are not only justified by gaining computational efficiency at the expense of output quality, but that such simple heuristics can even outperform complex decision rules [Gigerenzer *et al.*, 1999].

The main finding of this paper is that cognitively inspired heuristics can indeed be exploited to filter the correct answers from the noisy answersets obtained when querying the Web of Data. Perhaps the most surprising finding is that such heuristics are extremely simple when compared to those proposed in the literature, while still producing good results.

The overall benefit from this work is that it is now possible to efficiently select the most likely correct answers when querying the Web of Data. Our approach has as additional benefit that our selection heuristic can be tuned to favour either recall or precision.

An important secondary contribution of this work is the construction of a benchmark of general knowledge queries with their Gold Standard answers. Each of these has also been formulated as a SPARQL query, and the 1896 answers to these queries have been manually ranked on their quality. This collection is freely available for other researchers as an important tool in benchmarking their query strategies over the Web of Data.

The paper is structured as follows: In Section 2, we first discuss the construction of this benchmark. In Section 3, we report on how good a human subject is in recognising the Gold Standard correct answers for these benchmark questions. In Section 4 we discuss some of the cognitive science literature that justifies the definition of our "fast and frugal" computational heuristic. In Section 5 we then assess the performance of this heuristic, and we show that its performance

is comparable to that of the human subject. In Section 6 we compare our approach to related work in the literature. In the final section 7 we compare our heuristic to those presented in the Semantic Web literature.

2 A Benchmark for Querying the Web of Data

Over the past decade, the Semantic Web community has built and adopted a set of synthetic benchmarks to test storage, inference and query functionality. Some of the most well known benchmarks are the Lehigh LUBM benchmark, [Guo *et al.*, 2005], the extended eLUBM benchmark [Ma *et al.*, 2006] and the Berlin SPARQL benchmark [Bizer and Schultz, 2009][1]. However, all these are *synthetic* datasets. There is a shortage of *realistic* benchmarks that provide realistic queries plus validated ("Gold Standard") answers. The sample queries on the webpages of Linked Life Data[2] and FactForge[3] are examples of such realistic queries, but they do not come with a validated set of Gold Standard answers.

Set of questions. For an experiment investigating how people search for information in their memory, [Neth *et al.*, 2009] designed a set of general knowledge questions. Each question identifies a natural category by a domain label (e.g.,'Geography') and a verbal description (e.g., 'African countries') and asks participants to enumerate as many exemplars as possible (e.g., 'Algeria', 'Angola', 'Benin', etc.). Questions were drawn from diverse areas of background knowledge (e.g., arts, brands, sciences, sports) and included "Name members of the pop band ABBA", "Name Nobel laureates in literature since 1945", etc.

Gold Standard answers. [Neth *et al.*, 2009] determined a set of correct answers for each question. The number of true exemplars varied widely between categories (from 4 to 64 items). Particular care was given to the completeness of the answer set by including alternative labels (e.g., 'Democratic Republic of the Congo', 'Zaire') and spelling variants ('Kongo').[4]

SPARQL queries. We have developed a set of 47 SPARQL queries, made to resemble the questions from [Neth *et al.*, 2009]. For this translation, we used a number of well-known namespaces, such as DBPedia, Freebase, Geonames, Umbel, etc. As an example, the question about ABBA members translates to the SPARQL query shown in Figure 1.

SPARQL answers. To complete this benchmark collection, we executed all of our queries against FactForge[5]. FactForge [Bishop *et al.*, 2010a] is a collection of some of the most central datasources in the Linked Open Data cloud. It hosts 11 datasets, including DBPedia, Freebase, Geonames, UMBEL, WordNet, the CIA World Factbook, MusicBrainz, and others. Several schemata used in the datasets are also loaded into FactForge, such as Dublin Core, SKOS and FOAF. FactForge uses the OWLIM reasoner [Bishop *et al.*, 2010b] to materialise all inferences that can be drawn from the datasets and their schemata. This results in some 10 billion retrievable statements, describing just over 400 million entities. Although FactForge is a subset of the entire Web of Data, it is currently one of the the largest available subsets that is both closed under inference and queryable.

```
SELECT DISTINCT ?member ?label
 WHERE {
  ?member skos:subject dbp-cat:ABBA_members
  ?member rdfs:label ?label
 FILTER(lang(?label) = "en")
}
```

dbpedia:Agnetha_Fältskog	Agnetha Fältskogen
dbpedia:Agnetha_Fältskog	Agneta øase Fältskogen
dbpedia:Anni-Frid_Lyngstad	Anni-Frid Lyngstaden
dbpedia:Anni-Frid_Lyngstad	Frida Lyngstaden
dbpedia:Benny_Andersson	Benny Anderssonen
dbpedia:Björn_Ulvaeus	Björn Ulvaeusen
dbpedia:Ola_Brunkert	Ola Brunkerten
dbpedia:Stig_Anderson	Stig Andersonen

Figure 1: Example query and answer-set

Running our 47 queries against FactForge[6] resulted in 1896 answers. An example answer-set is shown in Figure 1.

The entire resource (original questions, their SPARQL translations, the Gold Standard answers, as well as query-results against FactForge) are available online[7].

3 Human Performance

In order to judge how good our heuristics will be at recognising correct answers, we measured how good a human was at this task. A human subject (educated at university level, using the Web as a reference source, and asked to do this at reasonable speed) ranked all 1896 answers on a 5 point scale, with 5 indicating the answers on which the subject was most confident that they are correct, and 1 indicating answers on which the subject was most confident they were incorrect[8,9]. We are now interested in the question whether a human subject can identify correct answers with sufficiently high confidence. For this, we introduce the following notations:

Notation: We use Q to indicate the query, with Q ranging from #1 to #47 in our benchmark collection. The set of Gold Standard answers to query Q is written $G(Q)$. The set of retrieved answers to query number Q are written $A(Q)$. The set of answers to query Q that were scored with a confidence ranking of T or higher is written as $A_T(Q), T = 1, .., 5$.

[1] More benchmarks are described at `http://www.w3.org/wiki/RdfStoreBenchmarking`.

[2] `http://linkedlifedata.com/sparql`

[3] `http://factforge.net/sparql`

[4] The answers to some questions (e.g., the teams in particular leagues) are subject to periodic changes. This requires updating some of the answers when using them on current data sets.

[5] `http://factforge.net/`

[6] version of August 2010

[7] `http://www.larkc.eu/resources/published-data-sources`

[8] These human rankings are also available from the aforementioned URL.

[9] Because this was only a single human judge, we cannot measure how reliable the scoring is, since we have no measurement for inter-subject agreement. This would be a useful piece of future work to strengthen the value of our dataset.

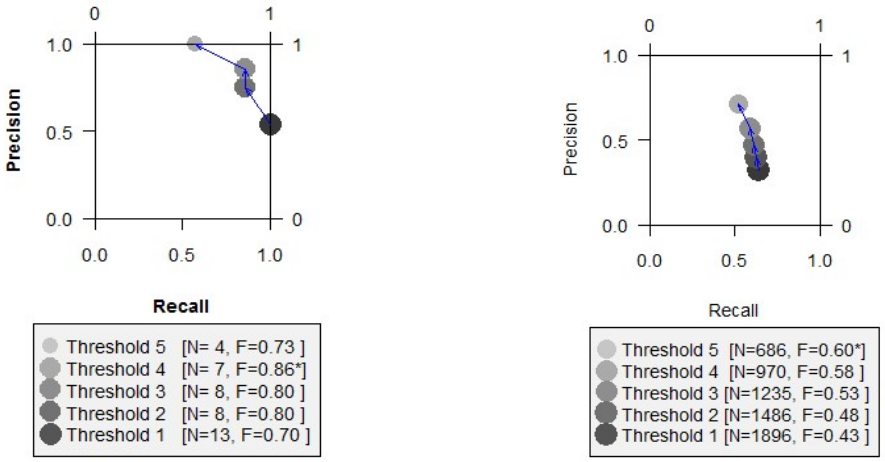


Figure 2: Performance of human subject: (a) on an example query and (b) accumulated results

Obviously, $A_1(Q) = A(Q)$ (all answers are included at confidence threshold $T = 1$), and the size of $A_T(Q)$ decreases with increasing T.

In our experiment described below, the size of $G(Q)$ is typically a few dozen items (since this is how the cognitive scientists designed their queries). The size of $A(Q)$ varies greatly from a dozen to several hundreds, showing that some answer sets contain many wrong results, i.e. $A(Q) \not\subseteq G(Q)$, for some Q. We will see below that also $G(Q) \not\subseteq A(Q)$ for some Q, i.e., FactForge is not complete for all of our queries.

To judge the performance of our human subject in recognising correct answers, we plot the recall and precision of $A_T(Q)$ as a function of his confidence threshold T, where the correctness of the answers in $A_T(Q)$ is determined against $G(Q)$. The comparison of the SPARQL results in $A_T(Q)$ against the (natural language) elements in $G(Q)$ is done using the `rdfs:label` of the elements in $A_T(Q)$.

Example query. As an illustration, Figure 2(a) shows the performance of our subject on query $Q = \#31$: "What are the highest mountains (peaks) of each continent". At threshold level $T = 5$ (i.e. when aiming to select only the answers about which he is most confident that they are correct), the subject scores a precision of 1.0 but recognises only $N = 4$ out of the seven summits, i.e., the recall is only .57. When including answers at lower confidence levels, the recall increases, finally reaching 1.0 at $T = 1$. This shows that FactForge does indeed contain all correct set answers for this query, i.e. $G(\#31) \subset A(\#31)$. However, the increase in recall comes at the cost of also including some incorrect answers, with precision dropping to a final value of .5. The maximal performance (using the macro-averaged F-measure to combine precision and recall) is $F = .86$, and is reached at confidence threshold $T = 4$.

Accumulated results. Figure 2(b) shows the recall and precision figures of our human subject accumulated over all 47 queries. It shows that even at $T = 1$ the recall is only just above 0.6. This tells us that FactForge is indeed incomplete for our set of queries, and it is simply impossible for any subject (human or machine) to do any better on this set of queries.

Figure 2(b) shows a near perfect performance by our human subject: when increasing his confidence levels T, the precision of $A_T(G)$ increases from 0.25 to 0.75, while paying almost no penalty in decreasing recall (dropping from 0.6 to 0.5). In other words: when stepping up the confidence level from T to $T + 1$, the sets $A_{T+1}(G)$ have lost some of the wrong answers that were still in $A_T(G)$ while maintaining most of the correct answers in $A_T(G)$. Or stated informally: our subject is actually rather good at identifying the correct answers among $A_T(G)$. In terms of the graph in Figure 2(b), a perfect performer would result in a vertical plot (increasing precision at no loss of recall). The human subject comes close to that perfect plot. Consequently, the highest score ($F = .60$) is obtained at confidence threshold $T = 5$.

4 Definition of Selection Heuristic

Fast and Frugal Heuristics. Biological cognitive agents (be they human or animal) have to perform a similar task on a daily basis: given a set of possible alternatives, which ones are "the best", ranging from distinguishing edible from inedible foods to deciding if another agent is friend or foe. In 1969, Herbert Simon noted that this selectivity is based on rules of thumb, or heuristics, which cut problems down to manageable size [Simon, 1969]. The basic idea is that the world contains an abundance of information and the best solution is not necessarily to integrate as much information as possible, but rather to select some information and use that for reasoning.

In the late 20th century there was a debate between decision theorists on whether these rules of thumb had positive or negative consequences for the quality of decisions humans were making. In the *heuristics-and-biases* program examples of faulty human decisions under uncertainty are presented, when comparing them to the normative standard of probability theory [Tversky and Kahneman, 1974]. A different view is advocated by the *fast and frugal heuristics* program, which has demonstrated that simple rules requiring little information often perform as well as (and sometimes even better

than) more complex optimization algorithms [Gigerenzer *et al.*, 1999].

Such fast and frugal heuristics, which in given situations outperform optimization methods despite using substantially less information and computational resources, are important beyond the realm of cognitive science. Regardless of their psychological validity, we can apply such heuristics to issues of complex decision making in computational settings.

The relevance to the Semantic Web lies in that these algorithms could improve the yield of data retrieval without excessively damaging its quality, thus producing the results we want at lower computational costs, but not guarantying optimal quality or completeness of results. In a term coined by Simon, we wish to *satisfice* [Simon, 1956].

Similarity as a Heuristic. If one thinks about a query as defining a target region in some semantic space, one would expect the results of the query to be clustered in that target region. That is, the results that are most similar to each other are most likely to be close to the center of the targeted region. It seems reasonable to assume the farther away a result is from the center of this estimated target region, the more likely it is to have been included in the results due to error.

Two classical approaches to formalizing similarity are featural approaches, epitomized in Tversky's contrast model [Tversky, 1977], and spatial models [Shepard, 1957]. In spatial models, similarity is defined as the distance in a defined metric space between two items. Spatial models have predefined spaces and each item is a separate point in the space, making symmetrical similarity natural. Tversky's contrast model focuses on features shared between items and features not shared between items. Similarity is then defined by the proportion of shared features in the total features of an item

Computational definition of similarity. Tversky's similarity model based on shared features fits very naturally with the datamodel underlying RDF: an "item" is a URI s_1, a "feature" is a triple $\langle s, p, o\rangle$, and two features are shared between two items s_1 and s_2 if they have the form $\langle s_1, p, o\rangle$ and $\langle s_2, p, o\rangle$. For example, two objects share a feature if they both have a `skos:subject` property with object `dbp-cat:ABBA_members`. Formally:

Definition 1 *The similarity $S(s_1, s_2)$ between two resources s_1 and s_2 in a graph G is defined as:*

$$S(s_1, s_2, G) = ||\{(p, o)|\langle s_1, p, o\rangle \in G \text{ and } \langle s_2, p, o\rangle \in G\}||$$

i.e. similarity is defined as the number of feature-value pairs in G that are shared between s_1 and s_2. This looks even simpler as a schematic SPARQL query:

```
SELECT COUNT(?p)
 WHERE {<s1> ?p ?q
        <s2> ?p ?q}
```

where `<s1>` and `<s2>` must be replaced by specific URIs.

This similarity measure can now be used to define a heuristic confidence estimate for query-answers:

Definition 2 *The confidence estimate $C(a, Q, G)$ for an answer $a \in A(Q)$ to a query Q over a graph G is defined as*

$$C(a, Q, G) = \Sigma_{a' \in A(Q)} S(a, a', G)$$

i.e. the confidence estimate of an answer a is simply the aggregate similarity of a to every other answer a'. This similarity heuristic is similar to the "clustering hypothesis" as it is known from Information Retrieval [Tombros and Van Rijsbergen, 2001], namely that relevant documents tend to be more similar to each other than to non-relevant ones, and therefore tend to appear in the same clusters.

Alternatives. Of course a number of variations on this definition would be possible. Instead of counting the total number of shared features $\langle s_1, p, o\rangle$, we could calculate the *fraction* of shared features, as suggested in [Tversky, 1977]. Because of the fairly uniform nodes in RDF graphs, we would not expect this to make much difference.

We could also have used the weaker definition of only counting shared properties p without demanding that they have the same values: $\langle s_1, p, \_\rangle$ and $\langle s_2, p, \_\rangle$. For example, two objects are similar if they both have a `dbp-prop:manufacturer` property, even if that property has different values. However, due to the weak nature of many of the features (e.g. `rdf:type`, `skos:subject`) we expect that this will generate too high similarity ratings.

More reasonable would be to include shared *inverse* features $\langle o, p, s_1\rangle$ and $\langle o, p, s_2\rangle$. This would account for inverse modelling in the RDF graph, for example using `is-manufacturer` instead of `manufactured-by`. Such inverse properties are rare in FactForge, but this would be worth further investigation.

5 Heuristic Performance

We are now in position to measure how good the heuristic from Def. 2 is at selecting the correct answers for a query. In order to use the same evaluation procedure as for the human subject in Section 3, we divide for every query Q the interval $[min_{a \in A(Q)} C(a, Q, G),\ max_{a \in A(Q)} C(a, Q, G)]$ uniformly in five equal steps.

Figure 3(a) shows the performance of our similarity based confidence estimate on the same "seven summits" query as in Figure 2. Trivially, the heuristic performance at the lowest confidence level equals that of the human performance at the lowest confidence level, at a reasonably high F-value of .43, achieved with trivially accepting all answers as correct. This is caused by the high quality of FactForge. Just as the human subject, the heuristic achieves a precision of 1.0 at the highest confidence level, but only manages to do so at a very low recall of .28 (2 out of 7), whereas the human subject managed to maintain a recall of .57 (4 out of 7).

Figure 3(b) shows the performance of the similarity based confidence estimate accumulated over all queries (as Figure 2(b) did for the human subject). The conclusion from this comparison is mixed: On the one hand, the human recall-precision curve lies everywhere above the heuristic curve, on the other hand the highest heuristic F-score ($F = .53$) is within 10% of the highest human F-score ($F = .60$). This is all the more surprising since our heuristic uses no background knowledge whatsoever, and only counts the number of shared feature-value pairs between the members of the answer set. This lends some support to the conclusion that well chosen very simple fast and frugal heuristics can achieve high performance levels.

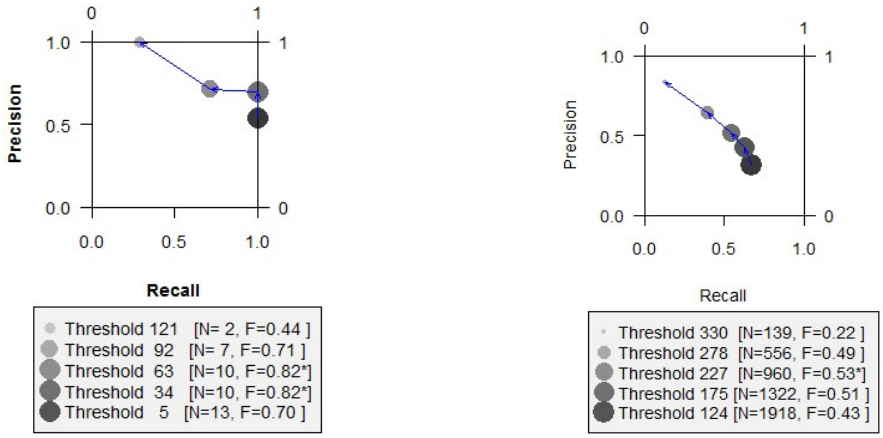


Figure 3: Performance of the similarity heuristic: (a) on an example query and (b) accumulated results

6 Related Work

The topic of ranking query results has been studied since the early days of the Semantic Web, and is itself based on even longer lines of research in fields such as Information Retrieval. In fact, our heuristic is closely related to the "clustering hypothesis" as it is known from Information Retrieval [Tombros and Van Rijsbergen, 2001]. Our space is insufficient here to provide an extensive literature survey. Instead, we will discuss a few salient differences between our approach and the literature:

Some of the literature on ranking is concerned with *relevance*: determining which answers are relevant to an unstructured query in natural language, or relevant for a user based on their profile (see [He and Baker, 2010; Stojanovic *et al.*, 2003; Anyanwu *et al.*, 2005; Hurtado *et al.*, 2009] and others). Although interesting and important, this work is not pertinent to the current paper, since we start with SPARQL queries (hence query-relevance is not an issue), and we are not considering user-profiles, but we are trying to recognise objectively true answers.

Another part of the literature is concerned with ranking answers by *importance*. Typically, this is done by a variety of pagerank-style analysis of the structure of the Semantic Web, trying to locate which resources are more important, more authoratitive, more trustworthy, etc. [Bamba and Mukherjea, 2005; Ding *et al.*, 2005; Anyanwu *et al.*, 2005]. Our approach differs from all this work in an important way: we do not do any a priori analysis of the structure of the large RDF graph that we are querying (a graph with billions of edges and hundreds of millions of nodes). Instead, we only take the URIs that are returned as a result of a query, and we compute some very simple local properties of these URIs (namely the number of shared feature-value pairs). As we have shown in Section 4 this is, surprisingly, already enough to rank the answers such that the best answers get a high ranking, performing within a 10% range of human performance.

Some of the literature on ranking deals with ranking different kinds of objects from what we consider: [E.Thomas *et al.*, 2005; Alani *et al.*, 2006; Tartir and Budak Arpinar, 2007] and others rank ontologies, Swoogle ranks Semantic Web documents [Ding *et al.*, 2005], [Vu *et al.*, 2005] and others rank services, etc. These works rely on fairly sophisticated analyses of the object-to-be-ranked: internal structure of the ontologies, semantic descriptions of the functionality of the services, etc. Instead, we rank only sets of atomic URIs (the members of $A(Q)$), and Although it might seem harder to rank such simple objects, since they come with very little structure to base the ranking on, we have shown in Section 4 that a simple analysis of very little information is sufficient to obtain good ranking results, in line with the fast and frugal heuristics program proposed by [Gigerenzer *et al.*, 1999]. It would be interesting to investigate if such simple (or seemingly simplistic) analysis would also yield good results when applied to more complex objects such as ontologies or services, potentially replacing the more sophisticated ranking techniques found in the literature until now.

A work that is quite close in aim to ours is [Lopez *et al.*, 2009]. Their "semantic similarity" is similar in spirit to ours: it tries to spot wrong answers through their large semantic distance to many of the other answers. However, the semantic distance in [Lopez *et al.*, 2009] is calculated as the distance in a shared ontology. Ours is a much simpler method: we need no ontology at all, and distance is simply calculated as the number of shared feature-value pairs.

7 Conclusion

In this paper, we have shown that a simple cognitively inspired heuristics can be used to select the correct answers from among the query results obtained from querying Linked Open Data sets. Our heuristic is extremely simple and efficient when compared to those proposed in the literature, while still producing good results, on a par with human performance.

Our work differs from previous work in the following important ways: Firstly, we do not require any expensive prior pagerank-style analysis of the structure of the entire dataspace we are querying. Instead, we do a simple count over information local to the URIs that are returned as query results. In the worst case, the cost of our analysis is limited to retrieving all properties of all elements in the answer set, a cost

which is negligible when compared to the large scale network analysis needed for most ranking approaches. Also, any such a prirori large scale link-analysis is likely to be outdated when it is needed for querying. The information that our heuristic needs is so simple that it can be retrieved at query time itself, and is hence always up to date.

Secondly, we do not require any background knowledge or inferencing. No reference is made to any ontological background knowledge, it is not necessary to relate any answers to a shared ontology, and no inference of any kind is performed by our fast-and-frugal heuristic. All we require is to simply retrieve the properties of the elements in the answer set. These are simple atomic queries of the form $\langle s, ?, ? \rangle$, that are efficiently supported by the index-structures of any triple-store.

References

[Alani *et al.*, 2006] H. Alani, C. Brewster, and N. Shadbolt. Ranking ontologies with aktiverank. In *ISWC*, vol. 4273 of *LNCS*, pag. 1–15. Springer, 2006.

[Anyanwu *et al.*, 2005] K. Anyanwu, A. Maduko, A. Sheth. Semrank: ranking complex relationship search results on the semantic web. In *WWW '05*, pag. 117–127. ACM, 2005.

[Bamba and Mukherjea, 2005] B. Bamba and S. Mukherjea. Utilizing resource importance for ranking semantic web query results. In *Semantic Web and Databases*, vol. 3372 of *LNCS*, pag. 185–198. Springer, 2005.

[Bishop *et al.*, 2010a] B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev, and R. Velkov. Factforge: A fast track to the web of data. *Semantic Web Journal*, 2010. under submission.

[Bishop *et al.*, 2010b] B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev, and R. Velkov. Owlim: A family of scalable semantic repositories. *Semantic Web Journal*, 2010.

[Bizer and Schultz, 2009] C. Bizer and A. Schultz. The berlin sparql benchmark. *Int. J. On Semantic Web and Information Systems*, 5(1):1–24, 2009.

[Ding *et al.*, 2005] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and ranking knowledge on the semantic web. In *ISWC 2005*, vol. 3729 of *LNCS*, pag. 156–170. Springer, 2005.

[E.Thomas *et al.*, 2005] E.Thomas, H. Alani, D. Sleeman, and C. Brewster. Searching and ranking ontologies on the semantic web. In *K-CAP*, pag. 57–60, 2005.

[Gigerenzer *et al.*, 1999] G. Gigerenzer, P. M. Todd, and the ABC Research Group. *Simple heuristics that make us smart*. Oxford University Press, 1999.

[Guo *et al.*, 2005] Y. Guo, Z. Pan, and J. Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semantics*, 3(2-3):158 – 182, 2005.

[He and Baker, 2010] X. He and M. Baker. xhrank: Ranking entities on the semantic web. In *ISWC2010*, 2010.

[Hurtado *et al.*, 2009] C. Hurtado, A. Poulovassilis, and P. Wood. Ranking approximate answers to semantic web queries. In *ESWC*, vol. 5554 of *LNCS*, pag. 263–277. Springer, 2009.

[Lopez *et al.*, 2009] V. Lopez, A. Nikolov, M. Fernandez, M. Sabou, V. Uren, and E. Motta. Merging and ranking answers in the semantic web: The wisdom of crowds. In *ASWC*, vol. 5926 of *LNCS*, pag. 135–152. Springer, 2009.

[Ma *et al.*, 2006] L. Ma, Y. Yang, G. Qiu, Z.and Xie, Y. Pan, and S. Liu. Towards a complete owl ontology benchmark. In *ESWC*, vol. 4011 of *LNCS*, pag. 125–139. Springer, 2006.

[Neth *et al.*, 2009] H. Neth, L. Schooler, J. Quesada, and J.s Rieskamp. Analysis of human search strategies. LarKC project deliverable 4.2.2. Technical report, The Large Knowledge Collider (LarKC), 2009.

[Shepard, 1957] R N Shepard. Stimulus and response generalization: A stochastic model relating generalization to distance in psychological space. *Psychometrika*, 22(4):325–345, 1957.

[Simon, 1956] H Simon. Rational choice and the structure of the environment. *Psychological Review*, 63(2):129–38, 1956.

[Simon, 1969] H A Simon. *The Sciences of the Artificial*. The MIT Press, 1969.

[Stojanovic *et al.*, 2003] N. Stojanovic, R. Studer, and L. Stojanovic. An approach for the ranking of query results in the semantic web. In *ISWC 2003*, vol. 2870 of *LNCS*, pag. 500–516. Springer, 2003.

[Tartir and Budak Arpinar, 2007] S. Tartir and I. Budak Arpinar. Ontology evaluation and ranking using ontoqa. In *Int. Conf. on Semantic Computing (ICSC)*, pag. 185 –192, 2007.

[Tombros and Van Rijsbergen, 2001] A. Tombros and C. Van Rijsbergen. Query-sensitive similarity measures for the calculation of interdocument relationships. In *Proc. of the 10th int. conf. on Information and knowledge management CIKM01*. ACM Press, 2001.

[Tversky and Kahneman, 1974] A Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–31, 1974.

[Tversky, 1977] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[Vu *et al.*, 2005] L. Vu, M. Hauswirth, and K. Aberer. Qos-based service selection and ranking with trust and reputation management. In *OTM Conferences*, vol. 3760 of *LNCS*, pag. 466–483. Springer, 2005.

RULIE: Rule Unification for Learning Information Extraction

Alexiei Dingli
University of Malta
Malta
alexiei.dingli@um.edu.mt

Dale P. Busuttil
University of Malta
Malta
dbus0007@um.edu.mt

Dylan Seychell
University of Malta
Malta
info@dylanseychell.eu

Abstract

In this paper we are presenting RULIE (Rule Unification for Learning Information Extraction), an adaptive information extraction algorithm which works by employing a hybrid technique of Rule Learning and Rule Unification in order to extract relevant information from all types of documents which can be found and used in the semantic web. This algorithm combines the techniques of the LP^2 and the BWI algorithms for improved performance. In this paper we are also presenting the experimental results of this algorithm and respective details of evaluation. This evaluation compares RULIE to other information extraction algorithms based on their respective performance measurements and in almost all cases RULIE outruns the other algorithms which are namely: LP^2, BWI, RAPIER, SRV and WHISK. This technique would aid current techniques of linked data which would eventually lead to fullier realisation of the semantic web.

1 Introduction

Information Retrieval and Extraction is a major area of interest in the field of computer science and the study of intelligent systems. Besides having various sources of information, we always strive to optimise our use of this information and devise new methods of how to understand this information and access it efficiently.

Search Engines retrieve information by using techniques based on keywords which map documents. However, the results returned by the search engines are usually various documents with most of them containing irrelevant information and therefore few documents which contain the information that is needed by the user.

In designing RULIE we kept in mind the fact that to help intelligent agents understand what is written in the page, the web pages need to be annotated [Berners-Lee and Fischetti, 1999] [Berners-Lee, 2001]. Annotations are metadata that are attached to pieces of text which can be used to give meaning to the content of the page. RULIE is an information extraction algorithm which can annotate data in a semi-automatic way thus relieving humans from doing so. In order to do this, such an algorithm must have rules upon which to act in order to make annotations feasible. These rules can be used to populate an ontology either automatically or manually through the use of a learning algorithm.

In the first part of this paper we will explain the fundamental concepts of information extraction together with a brief background of techniques used to enhance the results of information extraction engines. We will also give an overview of the types of texts which are handled by such algorithms. Subsequently we will briefly cover basic fundamentals of information extraction algorithms and analyse in details the LP^2 and BWI algorithms. The foundations of these two algorithms are used as a base for RULIE. They were chosen because they produce the best results overall as per [Department and Ciravegna, 2001]. Later in this paper we will present the design including the algorithm of RULIE and subsequently its evaluation. The positive results of RULIE are compared with other existing algorithms and we conclude this paper by presenting possible future directions for RULIE.

2 Information Extraction

Moens [Moens, 2006] defines Information Extraction as *"the process of selectively structuring and combining data that are explicitly stated or implied in one or more natural language documents"*. Information Extraction engines are built on two key principles which are mainly the Identification of Relevant Data and the Storing of relevant extracted data in Appropriate Structures[Téllez-Valero *et al.*, 2005].

The architecture of an information extraction system is normally composed of two key elements [Siefkes and Siniakov, 2005]. These are namely the *Learning Phase* which is a set of steps that will use part of the Training Corpus to build the underlying extraction model. The *Testing Phase* involves using the model produced by the learning phase with the test corpus to extract information on unseen cases and thus evaluate the model itself. [Siefkes and Siniakov, 2005].Sometimes Information Extraction systems also involve two other phases: the Pre-Processing Phase and the Post Processing Phase. The Pre-Processing Phase comes before the training phase and generally involves getting the input ready for learning by adding further information to the text in the corpus. This usually involves identifying the important term in the text and using Natural Language Processing techniques to find the linguistic properties of the text thus making it easier to identify the relevant data. Some common shallow NLP components

used for this purpose are Tokenization, Part of Speech, Sentence Splitting [Choi, 2000] and Semantic tagging. The Post-Processing Phase involves formatting the output to accommodate the structured representation so that it can be easily processed [Siefkes and Siniakov, 2005]. Semantic annotation is then the process of finding occurrences of a particular entity or element in a larger text domain [Mika *et al.*, 2008]. Such processes are dependent on the quality of the training given to the algorithm[Mika *et al.*, 2008].

Information extraction finds its relevance in the semantic web since it aids further understanding of the text being processed. In their paper 'Linked data on the Web', Bizer *et al* explain linked data as the employment of *"RDF and HTTP to publish structured data on the web"* [Bizer *et al.*, 2008]. They go on by emphasising that the relevance of data stands in the connection between sources which hold data to make it more meaningful [Bizer *et al.*, 2008]. The term *'Linked Data'* was coined once again by Sir Tim Berners-Lee in 2006 [Berners-Lee, 2009]. In his publication, Berners-Lee made an effort in explaining that the semantic web is different from the other web because it is not just about putting data on a source. He emphasised that linking data is important so that *"a person or a machine can explore the web of data"* [Berners-Lee, 2009]. It is exactly at this point where one starts to appreciate the value of information extraction as a method of finding important parts of the document to be able to relate to others, relevantly.

3 Information Extraction Algorithms

Algorithms which involve learning are normally categorised into two main classes: *Supervised* and *Unsupervised*. In supervised learning, human intervention occurs before the algorithm starts learning (example in IE: annotating the training corpus going to be used as input) or at particular stages in the learning stages (example in IE: modifying the rules developed by the algorithm). For unsupervised learning, human intervention is kept to a minimum and usually only involves selecting the training corpus (annotation and rule modifying is done automatically by the algorithm) [Mitchell, 1997].

Another characteristic of Extraction algorithms is the way in which algorithms learn after inducing a rule or a pattern. The approach can be either *Bottom Up* or *Top Down*.In a Top-Down approach learning basically starts from scratch by getting a generic rule or pattern and develops it into a new rule. The generic rule or pattern has a high recall but a low precision. The precision will eventually increase through the algorithms customization of the rule. When the precision increases to its maximum, the algorithm will stop modifying the rule. In a Bottom-Up approach learning starts with a fully customized rule or pattern (i.e. it contains all features available) and reduces it to a more generic one by dropping some of its features. This will generally start with a rule or pattern high in precision but low in recall and will know it has perfected the rule when a maximum threshold error value defined by the user is reached [Kushmerick and Thomas, 2003].

Information Extraction Systems can be used to create database records with concepts learned from the documents [Chieu and Ng, 2002]. These concepts are normally referred to as slots. *Single Slot* and *Multi Slot* extraction refer to the number of concepts learnt simultaneously in each document. Single-Slot extraction means that, for each document, only one concept can be learnt at a time. On the other hand, in Multi-Slot extraction, each document can contain several concepts [Chieu and Ng, 2002] and the system can also extract relationships between those concepts.

There are three main types of Information Extraction algorithms [Siefkes and Siniakov, 2005] which are namely *Rule Learning*, *Knowledge Based* and *Statistical Approaches*. We will focus on the former type and below follows a detailed illustration of the major sub-classes in which Rule Learning Algorithms are organised:

Covering Algorithms The algorithms in this category adopt a special type of inductive learning that is based on divide and conquer. They rely on a predefined target structure and require a fully annotated training corpus where all the relevant information is labelled [Siefkes and Siniakov, 2005]. The algorithms then induce rules based on the annotated training corpus that extracts the slot fillers that represent the relevant information in the text. The instances that are mapped by the learned rules are removed from the corpus and the algorithms continue to learn rules for the remaining instances in the corpus until every instance is covered by a rule or according to a predefined setting [Siefkes and Siniakov, 2005]. An example of this algorithm is the LP^2 which is described in section 3.1.

Wrapper Induction Muslea *et al* [Muslea *et al.*, 2003] refer to Wrapper Induction (WI) as algorithms aiming to learn extraction patterns, called *wrappers*, by extracting relevant information usually from collections of semi-structured or structured documents that share the same domain specific information. At execution, wrappers are used on unseen collections of documents (which share the same domain specific information used in training) to fill predefined data structures with the information extracted [Muslea *et al.*, 2003]. Two prominent algorithms which are placed in this class are the STALKER algorithm [Sigletos *et al.*, 2004] and the Boosted Wrapper Induction (BWI) which is described in more detail in section 3.2.

Pattern and Template Creation This category reduces the amount of human effort and knowledge needed for preprocessing the corpus that is usually adopted in other rule-based learners to generate the extraction rules. Syntactic and lexical resources are provided to cover word semantics in the domain in order to compensate the lack of human interaction. Usually the induced patterns have a simple syntactic structure and the final patterns are chosen using statistical approaches from a large amount of initial patterns[Siefkes and Siniakov, 2005].

Relational Learners This group of learners is very similar to the covering algorithms category in the fact that they remove the instances that are covered by the rules induced and continue to work with the remaining instances in the corpus. However in this category, the algorithms consider relations between unlimited combinations of

features instead of limiting to a predefined one [Siefkes and Siniakov, 2005]. An example of a relational learner is the SRV learner.

3.1 LP^2

Ciravegna [Ciravegna, 2001] introduces LP^2 as *"an algorithm able to perform implicit event recognition on documents of different types, including free, structured and mixed ones."*. This type of covering algorithm borrows some ideas from Wrapper Induction and avoids data sparseness in natural language texts by making use of shallow NLP to add semantic meaning, while keeping the same effectiveness on semi-structured and structured texts. This algorithm splits the corpus into the *Training Corpus*, which will be used in the learning stage to induce the rules, and the *Testing Corpus*, which will be used to evaluate the learned rules [Ciravegna, 2001]. LP^2 performs Adaptive Information Extraction because it is capable of being ported to new scenarios without the need of an IE expert to configure it. Also it outperforms other learning algorithms and gives the best results in a large number of test cases made up of from different domains [Ciravegna, 2001]. It was for these reason that this algorithm has been chosen to be a base algorithm, together with BWI, for the purpose of this project.

Before inducing rules, LP^2 uses external linguistic tools over the corpus to give syntactic and semantic meaning to the text. A linguistic pre-processor is used to perform tokenization, morphological analysis, part of speech tagging, gazetteer lookup and generic dictionary lookup on the text corpus. This algorithm then uses the annotated text to generate rules for extracting the relevant information which is labelled with SGML tags by the user.

3.2 BWI

Freitag and Kushmerick [Freitag and Kushmerick, 2000] claim BWI to be a system *"that performs information extraction in both traditional (natural text) and wrapper (machine-generated or rigidly-structured text) domains"*. This algorithm learns simple extracting procedures, called wrappers, while applying boosting (a machine-learning technique) to improve the performance of simpler algorithms. The basic theory behind this technique is that *"finding many rough rules of thumb can be a lot easier than finding a single, highly accurate prediction rule"* [Schapire, 2003]. Generally wrapper induction algorithms try to generate one accurate rule that optimally has high precision and recall. In BWI, many low precision rules will be induced that individually are considered as weak but collectively make-up an accurate classifier. AdaBoost, the boosting algorithm, is applied by Freitag and Kushmerick in Wrapper Induction, to induce a number of weak learners (having a confidence value) that will form the ultimate wrapper.

4 Methodology

In this section we will show how we combined the algorithms illustrated in sections 3.1 and 3.2 to create RULIE. RULIE exploits the main features of the LP^2 and the BWI algorithms to produce a set of extraction rules which contain most of the advantages of both algorithms.

This system uses the Carnegie Mellon University (CMU) Seminar Announcements dataset as corpus for both training and testing. The CMU Seminar Announcements is a dataset 485 e-mails labelled by Freitag [Freitag, 1998] which contain information on seminars that were held at the CMU. This is a classic corpus used to evaluate Information Extraction algorithms.

4.1 Design

RULIE is divided into two parts as illustrated by Siefkes and Siniakov [Siefkes and Siniakov, 2005], i.e. the **Training Function** which represents the Learning Phase where extraction rules are learned from the training corpus and the **Testing Function** which represents the information extraction phase where the extraction rules learned in the first part are used to extract information from the Test Corpus.

Training Function

This process starts by first taking the corpus containing the examples and adding further annotations. Using GATE [Cunningham, 2002], it assigns syntactical and semantic information to each term in the document. The resultant *Annotated Corpus* is a representation of the training corpus but with added semantics. Subsequently the rule learning procedures start by taking the annotated corpus and generating rules which are too specific and considered to be weak. Later, the *Weak Rules* are generalised by reducing their length and relaxing their constraints in order to make them more generic thus increase recall and precision. Obviously, some rules will be discarded since this process would render them useless. The *Generalised Rules* are then tested by mapping them to documents in the training corpus. These *Mapped Rules* are then unified together by using the AdaBoost algorithm. After these rules are tested by calculating their recall and precision, they are sorted according to their recall and precision rating. This function then returns the final set of Extraction Rules.

Testing Function

In this function, a process similar to the training function is followed. An unseen corpus without annotated examples is used. This is then annotated using GATE as illustrated in section 4.1. By testing the annotated corpus with the *Extraction Rules* which were learned in the Training Function, RULIE extracts the relevant information present in corpus. The *Extracted Information* is then compared and statistical data on the effectiveness of the rules together with the overall performance of the system are computed. This function then returns these *Quantitative Metrics* together with the information extracted.

4.2 The Algorithm

The RULIE algorithm as illustrated in Section 4.1 is designed to induce a rule for every tag present in the document. Like the LP^2 and the BWI algorithms, RULIE will induce rules for the opening tags and the closing tags separately and thus it would learn a rule for an opening tag and then it would learn another rule for its closing tag. A rule is composed of two sets. A set representing the words found before the tag and another represent the words after. The size of the window of words can be set by the user during the training phase.

For each word **w**, the system will also consider its semantic category, its lexical category, the case of its first letter and its lemma. The position of the tag will also be stored for future reference. This rule representation is more similar to the of the LP^2 algorithm rather than the BWI approach. The LP^2 can give more significant and expressive wild cards than the BWI since it uses an external linguistic pre-processor. RULIE also uses an external pre-processor to develop additional semantic wildcards for the words in the rule.

Figure 1 shows the flowchart of the RULIE algorithm. The flow of the algorithm shows that it is based on two main loops where in the first loop, rules are initialised as shown by algorithm 4.1 and subsequently generalised as shown in algorithm 4.2. In the second loop, the mapped rules are revisited and the unification takes place as explained below.

In the function 'CreateInitialRule' illustrated in algorithm 4.1, a tag is found in the instance and stored in 'TagLocation'. Rule **R** is then built by individually storing the words before the tag and the words after the tag. Finally, the Rule is returned by the function.

Algorithm 4.1: CREATEINITIALRULE($Instance$)

$TagLocation \leftarrow position(Tag, Instance)$
$R.WordsBefore \leftarrow wordsbeforeTagLocation$
$R.WordsAfter \leftarrow wordsafterTagLocation$
return (R)

The generalisation of the rules illustrated in algorithm 4.2 occurs by going through all the constrains in a particular rule **R**. Each constrain is removed from the rule one-by-one in order to relax the rule and each time a constraint is removed, a new rule is created. The resultant rules are finally returned by the function.

Algorithm 4.2: GENERALISATIONS(R)

for each $Constrain : C \in R$
do $GeneralisedRule : G \leftarrow R - C$
return (G)

The last key element of the RULIE algorithm is the Unification module. Similarly to the BWI, this module employs the AdaBoost algorithm to combine the mapped rules in a single rule which is finally returned by the module.

5 Experimental Results

In Section 4 we showed how the CMU Seminar Announcements [Freitag, 1998] were used as a corpus to test this system and how these announcements were going to be divided into a training corpus and a test corpus. This division is obtained using a N-Fold Cross Validation approach, an evaluation technique that randomly partitions the data into N sets of equal size and each time the learning algorithm is run one of the N sets will be the test corpus while the remaining N-1 sets will be the training corpus [Zhu and Rohwer, 1996].

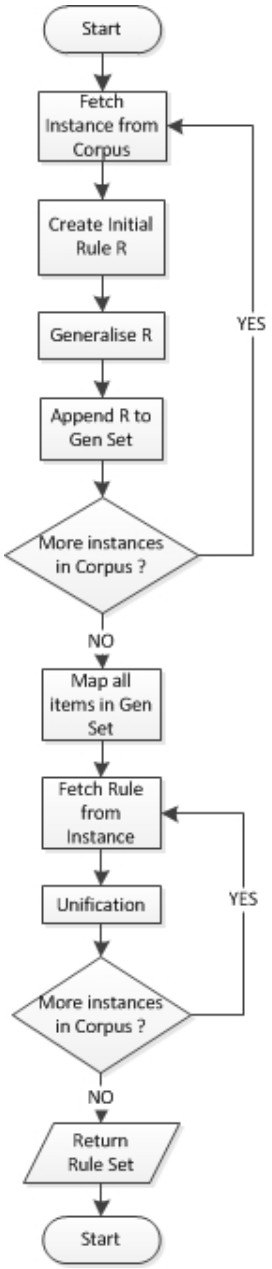


Figure 1: Flowchart of RULIE Algorithm (Source: Authors)

For the purpose of this project we are going to use 2-Fold, 5-Fold and 10-Fold Cross Validation to ensure that the system is properly evaluated on different division of the corpus and so that we can compare the results from one fold with the results of another. This ensures that no announcement used in the training phase would be used in the testing phase and vice versa. The system basically evaluates the four main fields in the CMU Seminar Announcements: The **Speaker** that is going to talk at the seminar; the **Location** where the seminar is going to be held; the **Start Time** and **End Time** of the seminar.

In our evaluation of RULIE, we compared the perfor-

	Precision	Recall	F-Measure
RULIE	0.895	0.908	0.898
LP^2	0.918	0.830	0.865
BWI	0.893	0.803	0.838
RAPIER	0.905	0.725	0.790
SRV	0.733	0.795	0.758
WHISK	0.765	0.633	0.655

Table 1: Comparison of IE Algorithms

mance measures of this algorithm with those of other prominent information extraction algorithms namely: LP^2, BWI, RAPIER, SRV and WHISK. The CMU Seminar Announcement was used since it can be considered as being a Gold Standard to test the performance measures of the above mentioned algorithms.

Table 1 presents a comparison of the Precision, Recall and F-Measure of all these algorithms including RULIE. The resuls in table 1 show that RULIE performs significantly better than the other algorithms used in information extraction. These results are pictorially displayed in figure 2.

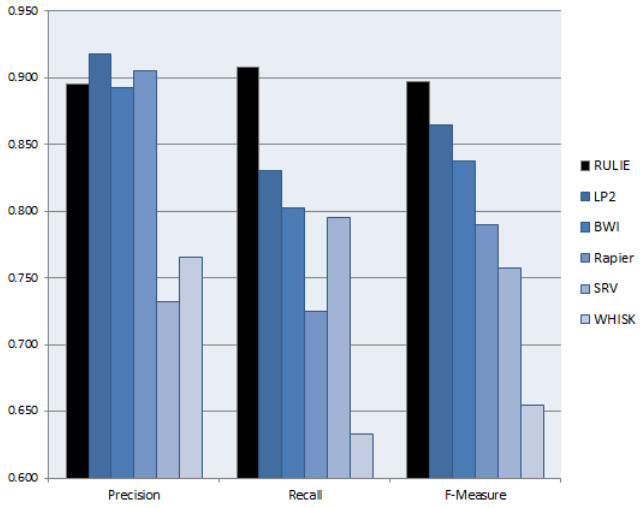


Figure 2: Performance measure of RULIE against other IE algorithms (Source: Authors)

6 Future Improvements

The efforts till now in RULIE were naturally revolving around its design, development and devising a system to evaluate it. More effort will be dedicated to evaluate the RULIE algorithm on different datasets.

The semantic web is the ultimate goal for this algorithm. A system implementing RULIE can be easily transformed into a web service and thus making it portable and accessible by both humans and other agents on the web. By creating an online version, the system can be used to produce liked-data [Berners-Lee, 2009].

7 Conclusion

In this paper we explored the importance of Information Extraction algorithms. This was done to develop RULIE our information extraction algorithm and to subsequent create comparisons with other Information Extraction algorithms.

The key algorithms were studied and out of them LP^2 and the BWI algorithm were examined in more detail. We chose these two algorithms to act as models for RULIE since they share positive attributes and our research showed that they had the potential to be combined. The rule induction technique in LP^2 and the rule unification in BWI have been unified under RULIE in order to get more expressive types of rules that apply to more situations. It is important to outline that although RULIE was created on the LP^2 and BWI technique, it introduces new ideas while refining the modelling algorithms. The major change was to relate the opening and closing tag rules with one another in order to remove the need of correction rules (used in the LP^2 algorithm) and to make the BWI algorithm handle richer rules.

Although as discussed in section 6 there is still room for improvement in RULIE, the results obtained till now are remarkable and promising. When considering the fact that RULIE outruns algorithms which were commercially employed, one cannot fail to notice the potential of this algorithm.

References

[Berners-Lee and Fischetti, 1999] Tim Berners-Lee and Mark Fischetti. *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper San Francisco, September 1999.

[Berners-Lee, 2001] James; Lassila Ora Berners-Lee, Tim; Hendler. The semantic web. *Scientific America*, May 2001.

[Berners-Lee, 2009] Tim Berners-Lee. Linked data. w3 Website, 06 2009. http://www.w3.org/DesignIssues/LinkedData.html.

[Bizer *et al.*, 2008] Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the web (ldow2008). In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 1265–1266, New York, NY, USA, 2008. ACM.

[Chieu and Ng, 2002] Hai Leong Chieu and Hwee Tou Ng. A maximum entropy approach to information extraction from semi-structured and free text. In *Eighteenth national conference on Artificial intelligence*, pages 786–791, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[Choi, 2000] Freddy Y. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[Ciravegna, 2001] Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of the 17th international joint conference*

on Artificial intelligence - Volume 2, pages 1251–1256, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[Cunningham, 2002] H. Cunningham. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254, 2002.

[Department and Ciravegna, 2001] Fabio Ciravegna Department and Fabio Ciravegna. an adaptive algorithm for information extraction from web-related texts. In *In Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, 2001.

[Freitag and Kushmerick, 2000] Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 577–583. AAAI Press, 2000.

[Freitag, 1998] D. Freitag. Information extraction from html: Application of a general learning approach. *Proceedings of the Fifteenth Conference on Artificial Intelligence AAAI-98*, pages 517–523, 1998.

[Kushmerick and Thomas, 2003] Nicholas Kushmerick and Bernd Thomas. Intelligent information agents. In Matthias Klusch, Sonia Bergamaschi, Pete Edwards, and Paolo Petta, editors, ., chapter Adaptive information extraction: core technologies for information agents, pages 79–103. Springer-Verlag, Berlin, Heidelberg, 2003.

[Mika *et al.*, 2008] P. Mika, M. Ciaramita, H. Zaragoza, and J. Atserias. Learning to tag and tagging to learn: A case study on wikipedia. *Intelligent Systems, IEEE*, 23(5):26 –33, septoct. 2008.

[Mitchell, 1997] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[Moens, 2006] Marie-Francine Moens. *Information Extraction: Algorithms and Prospects in a Retrieval Context (The Information Retrieval Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[Muslea *et al.*, 2003] Ion Muslea, Steven N. Minton, and Craig A. Knoblock. Active learning with strong and weak views: A case study on wrapper induction. In *In Proc. 18th Int. Joint Conference on Artificial Intelligence*, pages 415–420, 2003.

[Schapire, 2003] Robert Schapire. The boosting approach to machine learning: An overview, 2003.

[Siefkes and Siniakov, 2005] Christian Siefkes and Peter Siniakov. An overview and classification of adaptive approaches to information extraction. In *JOURNAL ON DATA SEMANTICS, IV:172212. LNCS 3730*. Springer, 2005.

[Sigletos *et al.*, 2004] Georgios Sigletos, Georgios Paliouras, Constantine D. Spyropoulos, and Michalis Hatzopoulos. Mining web sites using wrapper induction, named entities, and post-processing. In Bettina Berendt, Andreas Hotho, Dunja Mladenic, Maarten van Someren, Myra Spiliopoulou, and Gerd Stumme, editors, *Web Mining: FromWeb to SemanticWeb*, volume 3209 of *Lecture Notes in Computer Science*, pages 97–112. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-30123-3_6.

[Téllez-Valero *et al.*, 2005] Alberto Téllez-Valero, Manuel Montes-y Gómez, and Luis Villaseñor Pineda. A Machine Learning Approach to Information Extraction. *Computational Linguistics and Intelligent Text Processing*, pages 539–547, 2005.

[Zhu and Rohwer, 1996] Huaiyu Zhu and Richard Rohwer. No free lunch for cross-validation. *Neural Comput.*, 8:1421–1426, October 1996.

Understanding Web Data Sources for Search and Exploration

Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Emanuele Della Valle, Silvia Quarteroni
Politecnico di Milano, Dipartimento di Elettronica e Informazione
Milano, Italy
{firstname.lastname}@polimi.it

Abstract

As information sources become increasingly available on the Internet, searching and extracting information from them is getting more important. We present a data model and a general service framework for supporting search and exploration of heterogeneous data sources. In our approach, concepts and abstractions find suitable and efficient implementations. In this paper, we especially focus on the support of dynamicity and flexibility offered by our framework, from the definition of data sources to the interaction options offered to users when expressing queries and interacting with results.

1 Introduction

While search is the most adopted paradigm for accessing information over the Internet, users are becoming increasingly demanding, submitting queries that are getting more and more challenging in several senses. Search activities take the form of an interaction process instead of a single query; each single query itself becomes more complex, in terms of amount and extension of information the user asks for; and the information is searched also in the so called "deep Web", with valuable information (perhaps more valuable than what can be crawled on the "surface Web"). When a query is routed to several search-supporting data sources, results are typically ranked by each of them, thereby raising the challenging task of combining results into "globally good" answers to the original complex query.

Recent years are witnessing an exponential growth of Web data. Providers offer a plethora of ways of accessing their data sources, spanning from APIs (such as Google APIs, location based APIs, etc.) to proprietary query languages (such as YQL, the Yahoo! Query Language) and end-points accessible via standard query languages (e.g., SPARQL). This trend is associated with the increased tendency to labeling, tagging, and semantic linking of data, as motivated also by social networking applications (e.g., facebook Open Graph protocol[1]). These sources expose their data as semi-structured information (e.g., JSON, XML) and an increasing number of them also provide information as linked in the so-called Linked Data cloud [Bizer et al., 2009], with URI-based references between the resources.

This major change of paradigm is altering traditional Web publishing and challenging current approaches to Web navigation and information collection by end users [Baeza-Yates et al. 2010], [Kumar et al. 2009]. New methods are needed, which demand for cross fertilization between different disciplines: exploratory search approaches should be merged with usability studies and cognitive science in the purpose of identifying the best interaction paradigms on such new data sources; Web engineering approaches should be extended with data integration and Semantic Web/Linked Data-based practices (such as knowledge exploration tools) in the aim of connecting linked and non linked data, and providing proper navigational applications to the end users. All this must be as flexible as possible in terms of a continuous expansion of supported sources: users should be able to progressively "navigate" the space of services offered by search facilities, by expanding the results of their queries with those coming from related queries (e.g. schools or transportation information in the context of real estate search).

In this paper, we present Search Computing (SeCo), a framework introduced in [Ceri Brambilla 2010], [Ceri Brambilla 2011] for supporting search and exploration of heterogeneous data sources. In Section 2, we briefly illustrate the SeCo framework. In Section 3 we highlight concepts, models and methods with a special focus on extensibility and adaptation. We dedicate Section 4 to related works and we draw some conclusions in Section 5.

2 Search Computing Framework

Search Computing aims at developing new concepts, methods, algorithms, and tools for formulating and executing search queries over Web-based data sources. The framework focuses on the integration of existing search engines and data sources, so as to provide users with capabilities that go much beyond what can be queried from a single data

[1] http://developers.facebook.com/docs/opengraph/

source. Indeed, Search Computing queries are multi-domain, i.e. they involve a combination of data sources.

As our running example throughout the paper, we use a scenario where a user plans a leisure trip and wants to search for upcoming concerts (described in terms of music type, e.g., Jazz, Rock, Pop, etc.) close to a specified location (described in terms of kind of place, e.g. beach, lake, mountain, seaside), considering also the availability of good close-by hotels. We assume the availability of specialized data sources returning music events, place descriptions, hotels and restaurants, respectively. A sample multi-domain query in this scenario is "Where can I listen to a good jazz concert close to a nice beach, and stay at a nearby three-star hotel? Where can I find a good vegetarian restaurant close to the hotel?"

The first task in this framework is the segmentation of the multi-domain query into sub-queries (in the example: "Where can I listen to a good jazz concert in a location close to a beach?"; "Which three-star hotel nearby has the best rates?"; "Where can I find a good vegetarian restaurant close to the hotel?"). A second task is to bind each sub-query to the respective relevant data source.

In the Search Computing framework, we assume data sources to be exposed as Web services, with appropriate search interfaces capable of extracting items in ranked order. For this purpose, we have defined a number of concepts, methods and algorithms, used for the registration and ontological annotation of services, for query formulation (both as a single step and as an arbitrarily long process), for query optimization and execution over state-of-the-art, cloud-based architectures, for effective presentation of query results via visualization tools, and for result diversification and exploration paradigms. For each task, we design the most suitable user interfaces for reducing the interaction complexity, so as to foster the emergence of information from the "deep Web" in different ways.

The Search Computing framework is constituted by several sub-frameworks, as illustrated in Figure 1. The *service mart framework* provides the scaffolding for wrapping and registering data sources as services. Data sources can be heterogeneous (e.g. Web site wrappers, RESTful data sources, WSDL Web services, YQL data sources, SPARQL endpoints); registration requires a standardization of their interface to comply with our service invocation protocol. The *user framework* provides functionality and storage for registering users with different roles and abilities. The *query framework* supports the management and storage of queries as first class citizens: a query can be executed, saved, modified, and published for other users to see. The *service invocation framework* masks the technical issues involved in the interaction with the registered services, such as Web service protocol and data caching issues, registered in the service repository. Starting from this mapping, a *query planner* produces an optimized query execution plan, which dictates the sequence of steps for executing the query. Finally, the engine executes the query plan, by calling designated services through a *service invocation framework*, building the query results by combining the outputs produced by service calls, computing the global ranking of query results, and producing the query result outputs in an order that reflects their global relevance.

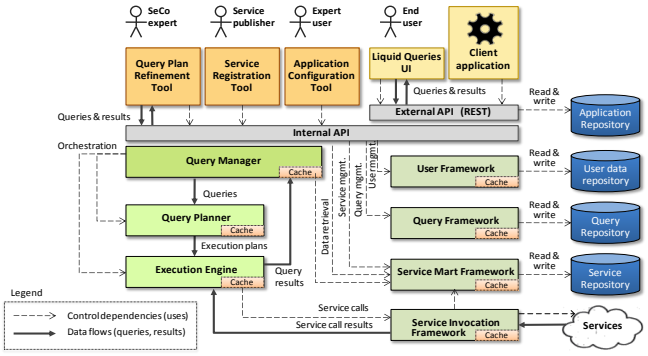


Fig. 1. Overview of the Search Computing framework

A Liquid Query interface [Bozzon et al. 2010], instantiated during the application configuration, supports the "search as a process" paradigm, based on the continuous evolution, manipulation, and extension of queries and results. In this configuration, the query lifecycle consists of iterations of the steps of *query submission*, when the end user submits an initial liquid query; *query execution*, producing a result set that is displayed in the user interface; and *result browsing*, when the result can be inspected and manipulated through appropriate interaction primitives, which update either the result set (e.g., by re-ranking or clustering the results) or the query (e.g., by expanding it with additional service marts or requesting for more results). This approach to development takes into account the trend towards "empowerment of the user", as witnessed in the field of Web mash-ups [Yu et al. 2007]. Indeed, only service development and service mart adaptation require programming expertise. All remaining design activities take place at service registration and application configuration time, so that designers do not need to perform low-level programming.

3 Concepts, Models and Methods

In this section we describe the main concepts involved in registering and querying services. As a general approach, we perform as much work as possible at service registration time, so as to simplify the query task. Service registration is therefore accomplished by system administrators who want to publish services; in doing so, they make services conformant to an interaction protocol. Adapting existing services to the framework includes the creation of service wrappers and data description, normalization, and registration as service marts. These tasks are performed by service publishers, which need limited programming skills for the construction of data and protocol adaptation components, as

they are supported by a template-based approach which uses standard adapter components, tailored to many service description formalisms. In the current implementation we support Web site wrappers, RESTful data sources, WSDL Web services, YQL data sources, and SPARQL end-points.

3.1 Semantic Service Description

We model services via an abstract representation, where all the services referring to a given real world object appear in the same cluster, so as to factor together similar services and offer a simple model of their interactions. In our model, represented as an Entity-Relationship diagram, the "focal" concept of each service class is represented as an entity (e.g., services returning restaurant information are "focused" on the *Restaurant* entity). Each entity includes a collection of strongly typed attributes, either atomic (defined by single-valued, basic types, e.g. *name*), or composed (defined by a set of sub-attributes, e.g. *chef*), or multi-valued (allowing multiple instances, e.g. *phone number*). Service associations are expressed in terms of named binary relationships, e.g. *near(Restaurant,Hotel)*. Fig. 2(a) shows an Entity-Relationship schema encompassing the services of our running example.

The conceptual representation as an abstract resource graph serves as a high-level view of the modeled domain. Further to such conceptual level, we model a lower-level logical representation describing the access to services. At this level of abstraction, an **access pattern** describes a legal way to invoke a service in terms of input (I), output (O), and ranking (R) attributes. While not defined at the conceptual level, additional access or ranking attributes that are not inherent properties of the domain entities (e.g., user location/ distance to the object) may appear at the logical level, with the purpose of allowing ranked access based on external properties. Finally, the logical level represents entity relationships as **connection patterns** consisting of a set of comparison predicates between pairs of attributes of the two services; these are interpreted as conjunctive Boolean expressions encoding the integration (join) between data instances associated with each access pattern.

At a yet deeper level of representation, **service interfaces** describe the physical implementation of services, and are directly linked to the information sources they access. Service interfaces are characterized by properties describing the cost of invocation (expressed as the response time and/or the monetary cost of invocation); how result items are clustered by the service at each call (a service is chunked when it returns a subset of the result objects at each call); the possibility of caching data (and in such case the expiration time); and **access limitations** (e.g., maximum number of allowed invocations per time unit). These aspects are crucial for cost-based service invocation strategies.
At registration time, data publishers are offered a number of facilities for connecting services to ontological knowledge, e.g. as provided by general-purpose systems such as Yago [Suchanek et al. 2007] and GeoNames[2]. Users are proposed to extract entity and attribute names from such ontologies, so as to guarantee greater "concept interoperability", which in turn may facilitate query understanding. In this way, SeCo concepts are explicitly mapped to ontological concepts, thereby opening to several research directions concerning query interpretation, rewrite, and understanding [Suchanek et al. 2010].

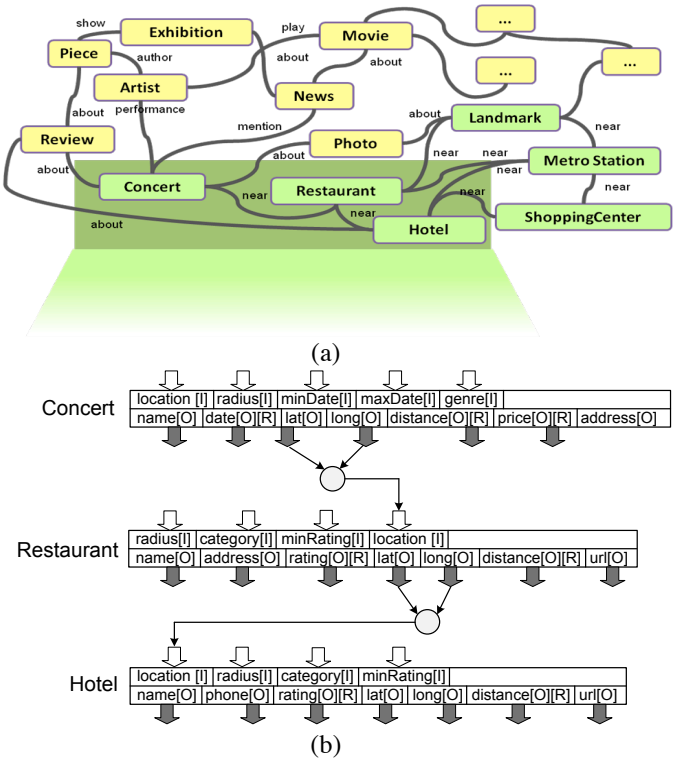


Fig. 2. Entity-Relationship conceptual description of Web services (a), and detailed view of the inputs and outputs of selected services, together with possible user inputs and query flows (b).

Three examples of access patterns suitable for our scenario are shown in Figure 2(b). Input, output and ranking attributes are highlighted. Two *join clauses* define the connection between the Concert, Restaurant and Hotel services based on their location. Figure 2(b) also exemplifies the submission of user input values for a query.

2.2 Query

A *Search Computing query* is a conjunctive expression over services, which includes two main aspects: the logical query clauses over the relevant data sources and the result ranking criterion. Although queries may be initially expressed over

[2] www.geonames.org

the conceptual description of services, eventually all queries are translated into adorned queries over service interfaces.

The query interface supports run-time customization of queries by choosing: optional selection predicates to restrict the objects retrieved by the query (e.g., a maximum price target for events or flights); default ranking criteria; visual preferences on the display of the result set (e.g., sorting, grouping, and clustering attributes or the size of the result list); a set of extra services usable by the end user to expand the current query (e.g., adding to selected movies their actors or reviews, or adding parking facilities next to the theatres where they are programmed).

3.3 Query Plan

A *query plan* is a well-defined scheduling of service invocations, possibly parallelized, that complies with their service interface and exploits the ranking in which search services return results to rank the combined query results [Braga et al. 2008]. Query plans are executed upon a specialized engine, called Panta Rhei [Braga et al. 2010], which emphasizes the flow of information extracted from services for producing query results. Panta Rhei represents a plan as a graph composed of operator nodes and edges, which describe the control- and data-flow between nodes. Several types of nodes exist, including service invocators, joiners, controllers and modifiers (chunkers, sorters, selectors). In order to support top-k answers (or approximations thereof), Panta Rhei provides *strategy* nodes that control the evaluation of joins by scheduling the corresponding service invocations. Our choice to define a new language rather than using or adapting an existing formalism was motivated by the specific requirements of evaluating complex queries over search services. For example, runtime adaptivity is achieved by the separation of the data flow and the control flow.

As an example of query plan model, Fig. 3 shows a pipe join which first extracts a list of *Concerts* from a search service and then uses these results to retrieve close-by *Restaurants* and *Hotels*, whose invocations are performed in parallel according to a predefined join strategy. Finally, results are joined by suitable join units, which execute the join of search engine results. Two strategy nodes—one for the outer pipe join and one for the inner parallel join—orchestrate the execution of the query by observing the output of data units (such as service invocators and joiners) and deciding the service to be invoked accordingly.

The execution engine supports flexibility in many ways. The system dynamically adapts to sudden changes of service performances by altering join strategies, e.g. by searching information from services which are dynamically performing better. Other elements of flexibility are offered by the architecture, which cleanly separates the execution aspects from the control aspects. Thanks to such separation, alternative join methods and strategies (e.g. heuristic or probabilistic vs. exact, pipe vs. parallel) can be added and dynamically selected depending upon needs.

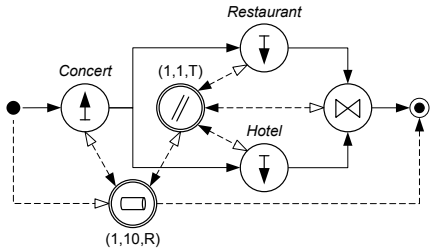


Fig. 3. Example of query plan in *Panta Rhei*

2.4 Liquid Interactions for Information Exploration

Liquid Query consists of a set of abstractions supporting exploratory search [Bozzon et al. 2010]. The Liquid Query interface visibly highlights the contribution of each search service to the composite result set, and supports users in fine tuning their requests to the underlying search services in several ways: by expanding the query with an extra search service, by adding or dropping attributes of an object, by asking for more results from a specific service, by aggregating results, by reordering them, by adding details (drill-down) or removing them (roll-up), and by choosing the best data visualization format at the level of individual object, combination or results set. In playing with the query and its result set, users alter the schema of the result set "container", and then results dynamically flow inside the new schema, adapting to it as a liquid to its container.

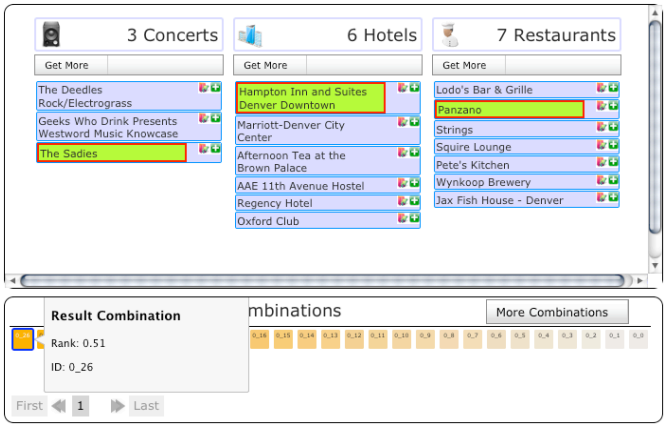
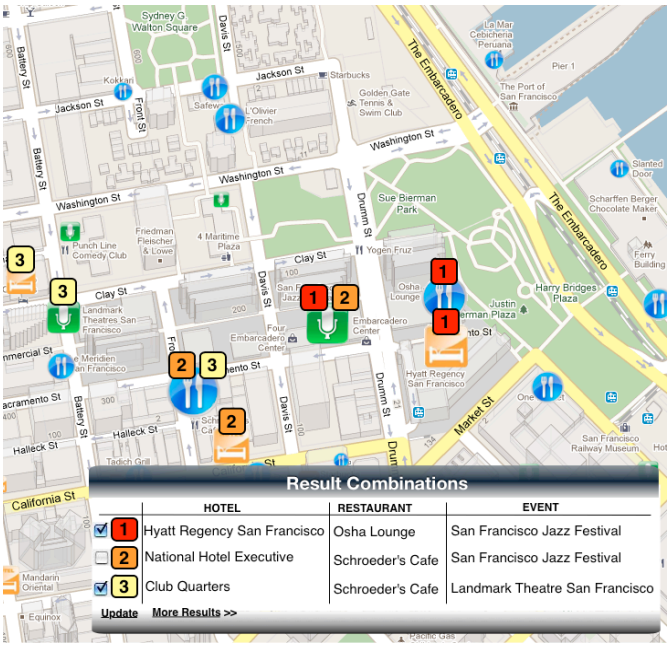


Fig. 4. Example of multi-domain query results

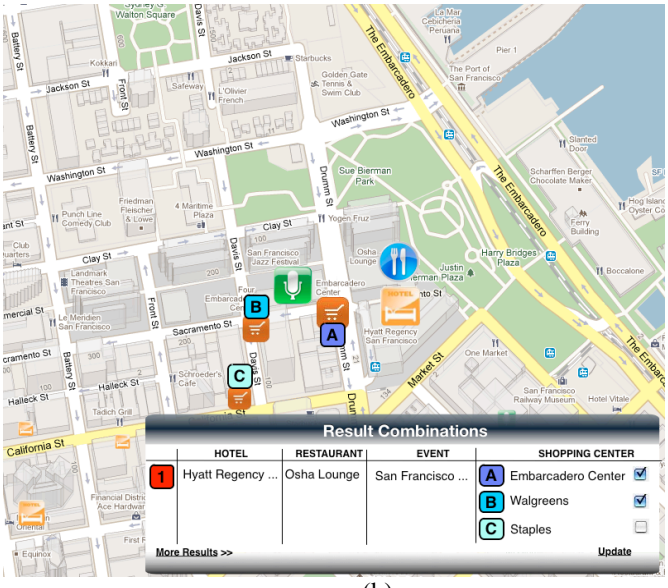
These operations apply to a tabular data representation similar to that of Google Square, which highlights the provenance of information from different search services. Fig. 4 shows an example of result table and a set of exploration options.

Thanks to the conceptual model representation, users can express their queries directly upon the concepts known to the system, such as hotels, restaurants, or concerts. Moreo-

ver, users can explore concepts by requesting for details on a specific object or by moving to other, semantically related concepts. For instance, they can select a concert and then relate it to other concepts such as performing artist, close-by transportation and parking facilities, other shows being played in the same night in town, etc. The query is focused (and restricted) to known semantic domains, offering great power in organizing the exploration of the search space as a continuous process.



(a)



(b)

Fig. 5. (a) Visualization of geo-referenced objects as ranked combinations; (b) Starting from a selected combination, the user explores related objects, such as shopping centers near the selected hotel

Query results are ranked; the user interface also allows users to specify ranking preferences, e.g. "distance", "quality", "price range" or a combination thereof [BrambillaCeri 2009]. At each stage, the user can select the most relevant object instances and continue exploration by choosing the next concept among the ones that can be connected to the current selection. Then, the user submits another object query, possibly by providing additional selection criteria; the system will retrieve connected object instances and form a "combination" with the previously retrieved ones. Backtracking is supported by means of a query orchestrator, which supports a query session and allows users to retract decisions and explore alternative options.

This exploratory process can be even more useful and efficient in some specific domains, such as geo-referenced objects. Fig. 5 shows a few simple exploration steps on geo-referenced objects via a map-based visualization. By selecting one or more combinations of objects (Fig. 5 (a)), users can prune the set of available options. Starting from an object, users can decide the exploration direction to follow towards other types of items based on nearness, such as metro stations or shopping centers. In this case, the new objects appear in the map and contribute to the newly calculated combinations (and rankings), as shown in Fig. 5(b). The exploration step can be repeated iteratively. The system is also automatically able to cluster non geo-referenced items within the semantically closest geo-referenced item.

In the context of the project, we use a conceptual model of query results, based on the knowledge of data types and of the ontological meaning of each attribute constituting the result itself, so as to match the results to the best representation widget. Of course, geo-references or time references can use maps and timelines. In addition, representation choices include the selection of the most significant representation dimensions and the use of data dependencies for positioning dependent attributes together with geo- or time-referenced attributes.

4 Related Work

Our work stands at the intersection of four main research trends in the field of software development: service integration, model-driven (Web) engineering, mash-up approaches and search-driven application development.

Service integration is a very prolific research field, comprising various directions: service definition (WSDL and ontologies such as WSMO and OWL-S), service registration (e.g., UDDI, and ebXML Registry), and service orchestration (WS-BPEL, BPMN, and interchange formats like XPDL). Associated tools comprise *service orchestration tools* based on BPEL and more general BPM tools. The Search Computing framework takes inspiration from the SOA vision, defining service-based invocation, collaboration between services for achieving a common result, and orchestration of the query plans.

From model-driven Web engineering approaches (e.g., WebML) the Search Computing framework borrows the ideas of *visual design and composition* of the applications. Modeling is actually ubiquitous in the SeCo approach, since it is applied in all the phases of the development.

Mash-up approaches (e.g., Yahoo Pipes) are even more inspiring, since they comply with the expert user need of an easy online tool to quickly configure and deploy applications. The Search Computing design tools rely on mash-up-like approaches, with emphasis upon simplifying user interaction [Yu et al. 2007].

Finally, search-based application development aims at accounting for the increase of sophistication and diversification of requirements for search applications. One example is the Symphony platform by Microsoft [Shafer et al. 2009], which enables non-developers to build and deploy search-driven applications that combine their data and domain expertise with content from search engines and other Web Services. Google Base API[3] and Yahoo! Query Language[4]) target the skilled software developer and rely on APIs or query languages. Google Squared[5] and Fusion Tables[6] produce tabular results of searches over web and proprietary information respectively; Kosmix[7] is a general-purpose topic discovery engine, which responds to keyword search by means of a topic page. These proposals miss some of the features offered by the Search Computing framework, including join of results, cost awareness, definition and optimization of query plans.

5 Conclusions

The Search Computing framework proposes a new paradigm where data are extracted from a constellation of cooperating services, and ranking of results is the dominant factor for composing them. This paradigm has proven very useful in supporting a flexible interaction over data sources.

Flexibility occurs at registration time, where users are provided with a number of tools and methods in order to adapt existing services and to describe their high-level semantics. Once such an investment is performed, the user is given a flexible high-level interface for browsing concepts at will, by dynamically adding and dropping data sources, by composing them arbitrarily, by asking for more results, by exploring solutions "forward" and "backward". While our current interfaces expect an active participation by users, we are also studying how to further simplify interfaces and adhere to classic, keyword-based or natural language interactions, by using references and annotations of services to ontological concepts in order to infer a complete meaning of complex queries from simple interactions.

Demonstrations of Search Computing results are taking place at WWW 2011 and ACM-Sigmod 2011 Conferences, and are available on www.search-computing.eu.

Acknowledgments

This work is funded by the Search Computing ERC IDEAS project. We wish to thank all the contributors to the project.

References

[Baeza-Yates et al. 2010] R. Baeza-Yates, P. Raghavan. Next Generation Web Search. In [Ceri Brambilla 2010].

[Bizer et al., 2009] C. Bizer, T. Heath, T. Berners-Lee. Linked Data - The Story So Far. Int. J. Semantic Web Inf. Syst. 5(3): 1-22, 2009.

[Bozzon et al. 2010] A. Bozzon, M. Brambilla, S. Ceri, P. Fraternali. Liquid Query: Multi-Domain Exploratory Search on the Web. Proc. WWW, 161-170, 2010.

[Braga et al. 2008] D. Braga, S. Ceri, F. Daniel, D. Martinenghi. Optimization of Multi-Domain Queries on the Web. Proc. VLDB 2008, pp. 562-573, 2008.

[Braga et al. 2010] D.Braga, S. Ceri, F. Corcoglioniti, M. Grossniklaus. Panta Rhei: A Query Execution Environment. Search Computing Challenges and Directions. In [CeriBrambilla 2010].

[Brambilla Ceri 2009] M. Brambilla and S. Ceri. "Engineering search computing applications: vision and challenges. ACM SIGSOFT ESEC/FSE '09, 365-372, 2009.

[Ceri Brambilla 2010] S. Ceri, M. Brambilla (Eds.). Search Computing Challenges and Directions. Springer LNCS, Vol. 5950, March 2010.

[Ceri Brambilla 2011] S. Ceri, M. Brambilla (Eds.). Search Computing Trends and Developments. Springer LNCS, Vol. 6585, March 2011.

[Kumar et al. 2009] R. Kumar, A. Tomkins. A Characterization of Online Search Behaviour. Data Engineering Bulletin, Vol. 32 N. 2, 2009.

[Yu et al. 2007] Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R. Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. IEEE Internet Computing 11(3), 2007.

[Shafer et al. 2009] Shafer, J.C., Agrawal, R., Lauw, H.W. Symphony: Enabling Search-Driven Applications. USETIM Workshop at VLDB, 2009.

[Suchanek et al. 2007] F. Suchanek, G. Kasneci, and G. Weikum. YAGO: a core of semantic knowledge. Proc. WWW 2007, pp. 697–706, 2007.

[Suchanek et al. 2010] F. Suchanek, A. Bozzon, E. Della Valle, A. Campi, S. Ronchi. Towards an Ontological Representation of Services. In [CeriBrambilla 2011].

[3] http://code.google.com/apis/base/
[4] http://developer.yahoo.com/yql
[5] http://www.google.com/squared
[6] http://tables.googlelabs.com
[7] http://www.kosmix.com