Ontology and Instance Matching for the Linked Open Data Cloud

BY

FEDERICO CAIMI
B.S., Politecnico di Milano, 2009

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2012

Chicago, Illinois

ACKNOWLEDGMENTS

I want to thank my thesis advisor, Prof. Isabel Cruz, for all her advices and suggestions, and for believing in my capabilities from the beginning of this thesis work. Additionally, I express my appreciation for Prof. Emanuele Della Valle's help regarding the organization and coordination of this work at Politecnico di Milano.

I want to thank also all my friends in the ADVIS laboratory. In particular Alessio Fabiani and Cosmin Stroe, who accompained me through all the good times we had in the laboratory and made this an experience of a lifetime.

Un enorme ringraziamento a tutta la mia famiglia. A mio papà, che mi ha insegnato a pensare in modo razionale e scientifico, e a stare con i piedi per terra. A mia mamma, a cui devo la creatività, l'interesse per la musica e la curiosità. A mia sorella Gaia, allo zio Massimo e alla nonna Libera, un grazie per tutto il sostegno e l'affetto che mi avete dato in questi anni.

A Federica, con cui ho condiviso momenti importanti nella mia crescita personale e con la quale mi aspetto di condividere ancora tanto. Un sincero grazie ad Antonio Faraldo, Alberto Gnemmi e Francesco Costa, che sono stati come una famiglia qui a Chicago.

A tutti gli amici del Politecnico e della UIC. Un particolare grazie a Francesco De Liva, che non mi ha mai fatto mancare i suoi consigli e la sua amicizia.

Infine, mi scuso con tutti coloro che non ho citato ma che comunque hanno contribuito al raggiungimento di questo traguardo.

FC

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER PAGE

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASM	Advanced Similarity Matcher
API	Application Programming Interface
CNA	Compound Names Analysis
DPLC	Distance-based Polysemic Lexical Comparison
EME	Equality Mappings Extension
GM	Global Matching
LIM	Label Instance Matcher
LOD	Linked Open Data
OMS	Ontology Matching System
OWL	Web Ontology Language
RDF	Resource Description Framework
STIM	Statements Instance Matcher
TIM	Token-based Instance Matcher

SUMMARY

The linked data paradigm has become a reality as more and more people and organizations publish their data following its principles. It envisions a web made by interlinked datasets that are easy to retrieve, query, and integrate. The main peculiarity of this technology is the presence of links between the data sources as well as the machine-processability of the data, achieved with the use of Semantic Web standards. However, since generating links between those datasets is costly and time-consuming, the need for automatic methods keeps increasing. For this reason ontology matching and instance matching, the fields studying how to automatically match semantic data sources, are being heavily investigated.

In this work we present an extension of AgreementMaker, a successful state-of-the-art ontology matching system, to effectively align ontologies and datasets available in the Linked Open Data cloud both at the schema and instance level. To achieve both of the goals, two research directions have been followed: the former is how to improve a general ontology matching system when matching LOD ontologies, while the latter is how to extend it to match instances maximizing the reuse of the components developed for ontology matching.

CHAPTER 1

INTRODUCTION

The role of data in our lives is growing rapidly in importance. Many applications make intensive use of data such as the temperature outside, the fuel consumption of our cars, the prices of different products, the feedback of customers and many others. A better use of all this information would help in making better decisions, starting from the single person experience to the global economy.

Imagine a place in the Web where all of these data can be queried as if they were in a giant database, and all kinds of information would be integrated to answer those queries. This is what Linked Open Data (LOD) is about. To achieve such a goal, a paradigm that takes ideas from today's Web and applies them to structured data has been proposed. Machine-processability of the data (e.g., use of structured formats) and the presence of links between datasets (e.g., specifying that an entity described in a data source is the same real-world object as an entity in another data source) are two of the main requirements.

There are many challenges to succeed in developing such an infrastructure, such as scalability, correctness of the information, automatic generation of datasets from unstructured sources, and the discovery of links between the data sources. All of these problems are being currently investigated in the literature. In this work, we will be focusing on the discovery of links between data sources.

The creation of links between heterogeneous data sources is a tedious work, which is costly and time-consuming. This is because it requires a detailed analysis to be performed by a domain expert. The quantity of links required for Linked Open Data to be an effective resource to be used in advanced tasks (e.g., question answering and improving Web search) is huge, and it keeps increasing at a fast pace as new datasets are added. For these reasons, the availability of efficient and reliable automatic or semi-automatic interlinking tools becomes a crucial factor for the success of the whole LOD paradigm.

The problem of establishing links between datasets in an automatic (or semi-automatic) fashion has been investigated in the Databases and Semantic Web communities. Links can be created at the schema level (e.g., concepts and classes) or at the instance level (e.g., individuals of classes and concepts). Depending on whether the considered links are at the schema or instance level, the two problems are considered separately in the literature and are referred to using different names. In the former case, the problem is known as schema matching in the databases community or ontology matching in the Semantic Web community, while in the latter respectively as record linkage or instance matching.

There are some differences between the problems tackled by the Databases and Semantic Web communities, due to the different underlying data models. Both ontologies and database schemata provide a vocabulary of terms used for describing knowledge in a domain of interest. However, relational databases do not provide the explicit and formal semantics, as they are specified at design-time in Entity-Relationship models but not encoded in the final schema. Ontologies instead are sets of formal axioms with explicit semantics (e.g., subclass axioms), which are exploited during the matching process. Moreover, the graph-oriented structure of the RDF model and ontologies is more flexible than the tabular structure present in relational databases. While all the rows in a database table share the same attributes, in ontologies it may happen that different combinations of attributes (properties) are used to specify instances of the same type. This leads to another type of heterogeneity that has to be solved in the matching process.

In this work we present an extension of AgreementMaker (1), a state-of-the-art ontology matching system, to effectively align ontologies and datasets available in the Linked Open Data cloud, both at the schema and instance level. These two types of links are equally important, as they are the basis for data integration at different levels. The former allows for querying different data sources unified under a common model. For instance, the mappings at the schema level can be used to query for all the entities of type 'Person' in different datasets. It may happen in some datasets that the concept of 'Scientist' is defined, and all of its instances should be returned as well. The latter allows for integrating the information about the same real world object from heterogeneous sources. These datasets may cover different aspects of the same entity, which would be all accessed through a single query.

The two problems are inter-connected and there is an overlapping in the techniques used, though there are some differences as well. Ontology matching is a consolidated research area, while instance matching is still at its own beginning. For the former there are a number of benchmarks and evaluation sets available, while for the latter only a few. There are techniques such as the use of vocabularies and the discovery of subclass relations that are effective and useful when matching concepts, but less when dealing with instances. Moreover, scalability is a crucial factor in instance matching, where the number of comparisons required to match data sources is significantly higher. For these reasons, and also for the strong separation present in the literature, we decided to divide the thesis in two parts, one for ontology matching and one for instance matching, which will be presented and evaluated independently.

Our contributions to ontology matching address the following research questions, reflecting the improvements needed to align LOD ontologies: How can a system like AgreementMaker be extended to handle mappings other than equivalence mappings (e.g., subclass mappings)? Can AgreementMaker achieve a good trade-off between accuracy and efficiency in the LOD domain? The proposed extension to the system has been evaluated against gold standards available in the literature, and proved to be better than other state-of-the-art tools. A preliminary version of this research has been published in (2).

In instance matching we address the following questions: how can we extend our system to match instances maximizing the reuse of the components already implemented for ontology matching? Can we provide an infrastructure that reduces the number of comparisons needed? Can AgreementMaker achieve good accuracy and efficiency in the LOD domain? The proposed extension to the system has been evaluated competing with other systems in the challenge organized by the Ontology Aligment Evaluation Initiative (OAEI), achieving competitive results. Part of this work has been published in (3), where our novel instance matching infrastructure is described.

In this document, we focus on the recent contributions to the system, explaining all the new matching features introduced in AgreementMaker. Before doing that, we provide a brief introduction to Linked Open Data and its principles in Chapter 2. In Chapter 3 we introduce ontologies, OWL, and SPARQL, then we define the ontology matching problem and give an overview of the techniques and tools available in the literature. As AgreementMaker is part of these systems, we include also an overview of its infrastructure and matching techniques. In Chapter 4, we cover our contributions to ontology matching, describing our novel matching methods which include the concept of Global Matching (GM), and a novel probabilistic algorithm called Distance-based Polysemic Lexical Comparison (DPLC) for discovering mappings using a mediator ontology such as WordNet. In Chapter 5 we move to the instance matching problem, providing a problem definition and an overview of the techniques and tools available, starting from the record linkage state-of-the-art. While the information provided in Chapter 2 can be easily found in the literature, in Chapter 3 and Chapter 5 detailed analyses and syntheses have been made to give an overview of the matching tools, also modifying the tables available in many surveys. The sixth chapter covers our contributions to instance matching, which include the design of an extensible infrastructure based on candidates retrieval and disambiguation, and the matching methods including Label Instance Matcher (LIM), Token-based Instance Matcher (TIM), and Statements Instance Matcher (STIM). Finally, we will end providing conclusions and future developments in Chapter 7.

CHAPTER 2

LINKED OPEN DATA

2.1 Motivation

An increasing number of organizations are sharing their data on the Web: examples are companies such as *Google* and *Amazon*, governmental entities in Europe and in the USA, scientific organizations, newspapers such as *The New York Times*. This data is then used by other users or organizations to offer new services and share aggregate information.

It is extremely important for the re-usability of data that it has a well defined *structure*. The more it is structured, the more it becomes reliably usable by third parties. However, the web nowadays is prevalently unstructured: the format for publishing web documents, HTML, is presentation-oriented instead of data-oriented. This is because the initial idea of the web was a collection of interlinked *textual* documents. The structured information is hidden into tags telling the browser how to visualize them. Therefore, to extract data of interest from raw HTML pages, some further processing is needed. This is usually non-trivial, because there is a lot of ambiguity in documents without a clear structure.

To address the problem of sharing data on the web that is re-usable by machine without efforts, mainly two approaches have been introduced. One is using *microformats*, which means attaching semantics to alternatively uncategorized text in web pages. With microformats, one can specify that a fragment of text is an entity of a particular type such as person or organization, and then specify some known and agreed-upon relations about them. This makes the automated extraction process easier and allows for more complicated use of the data. The main problem is that using microformats requires a big effort from the publishers and also it is still limited to a small set of types and relationships. The second is the use of Application Programming Interfaces (APIs), which allow the access to some websites' structured data over the HTTP protocol. This is becoming more and more common and led to the possibility for end users to develop *mashups*, small applications aggregating data from several different APIs to create new services, or even new businesses. Even though this is an important step forward for the use of data on the web, every API is something that requires a big effort to be integrated for several reasons: every API has its own rules, methods and formats to access the data it provides. Moreover, the data obtained accessing an API is strongly local, in the sense that it shows no links to other datasets and the identifiers will work only on those data. This opposed to the basic principle of the web, where the strength is the possibility to navigate related information through links. In the following sections Linked Open Data (LOD) will be introduced, which is an attempt to overcome the limitations of the web today as a data source.

2.2 What is Linked Open Data?

The term *Linked Data*[1] has been first introduced by Tim Berners-Lee, and it refers to a set of best practices for publishing and connecting structured data on the Web. Starting from 2006, an increasing number of data providers adopted these practices, leading to the creation

[1]http://www.w3.org/DesignIssues/LinkedData.html

of a global data space containing billions of facts, also known as the Web of Data. The Web has then been extended with structured data from diverse domains such as people, companies, publications, books, films, music, television and radio programmes, genes, drugs, scientific data, reviews and many others. The Web of Data enables new types of applications: browsers which allow users to navigate along links into related data sources, Linked Data search engines that crawl the Web of Data by following links between data sources and provide expressive query capabilities over aggregated data, similar to how a local database is queried today. Unlike mashups, which work exploiting a fixed set of APIs, Linked Data applications operate over a global, unlimited data space.

2.3 Principles

Tim Berners-Lee described four basic principles of Linked Data in (4). In one of his presentations at the TED 2009 conference, he stressed out that it's all about "extremely simple" rules. They are enumerated exactly how they appear in (4):

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs. so that they can discover more things.

The first principle consists in using Uniform Resource Identifiers (URIs) to identify things. URIs are the standard identifiers for resources on the Internet. These resources are usually web sites and documents available on the web. The principle is asking for a step forward: using the same name system also for real world objects, classes of concepts, and relationships. The idea is that everything can be described and identified with a string similar to the address of a web page. In the context of the Semantic Web, using URIs is something well understood and used already in many domains.

Using the HTTP protocol it is possible to retrieve the document associated with a certain URI available on the Internet. This is a universal lookup mechanism well-understood and agreed-upon by everyone. The second principle advocates the same should be with URIs associated with structured documents. Also URIs referring to real-world objects and concepts have to be dereferenced using the HTTP protocol.

As HTML has become the standard for publishing documents on the web, the Web of Data needs an accepted standard for structured documents. This is stated in the third principle, and the data model proposed is Resource Description Framework (RDF). RDF will be discussed with more details in the next section.

The fourth and last principle is about interlinking between structured documents. In hypertext web sites the value of the information provided is related to the value of what it links to. It is difficult to find on a single website all that we might want to know about a certain thing, but navigating the links to other pages the likelihood of finding satisfactory descriptions increases. Interlinking has to be extended to structured documents, with the added value of having *typed* links: while in hypertext there is just one type of link, RDF allows to interlink things specifying the *relation* between the linked objects.

All the above principles can be summarized in applying the fundamental concepts of the Web to the problems of publishing, sharing and using structured data. URIs, HTTP lookup and links are features that made possible the development of the web as it is now, a global space where everybody can publish and access information about everything. Linked Open Data is an attempt to reuse all the successful characteristics of the web in an even more ambitious project: the evolution of the web to a *global data space* (5).

Tim Berners-Lee proposed also a ranking for datasets:

1. Available on the web in any format, but with an open license
2. Available as machine-readable structured data (e.g. Microsoft Excel instead of image scan of a table)
3. All the above plus non-proprietary format (e.g. CSV instead of Microsoft Excel)
4. All the above plus using open standards from W3C (RDF and SPARQL) to identify things, so that people can link them
5. All the above plus the presence of linking other data sources, as to provide context

2.4 Resource Description Framework (RDF)

The Resource Description Framework (RDF) is a standard data model proposed by the World Wide Web Consortium (W3C). Its characteristics make it particularly suitable for data interchange on the Web, where there is a strong need for merging and evolving different schemas.

RDF can be summarized in three fundamental concepts: resources, properties and statements (6). Resources are objects and concepts in the real world that one may want to describe. The concept of resource is very generic and it embraces everything that can be thought of.

Every resource is identified with a Uniform Resource Identifiers (URI), using the same name system used on the Internet. Properties are special resources which are used to describe the relationships between other resources. Statements are triples consisting of a resource, a property and another resource or literal (e.g., standard datatype such as string or integer). They are instantiations of properties relating a subject (resource) to an object (resource or literal). Usually statements (or triples) are represented using the following notation: $\langle subject, property, object \rangle$.

The model expressed by RDF generalizes the linking structure of the Web. Statements can be seen as the links in web pages with the addition of a type (relation). This model forms a directed labeled graph, where the nodes are resources and the edges represent the typed link between them. An example of a triple is ⟨*http://www.example.com/Federico_Caimi, studiesAt, http://www.example.com/UIC*⟩, which intuitively expresses the fact that a resource named *Federico Caimi* studies at the *http://www.example.com/UIC* and is represented in the graph in Figure 1.

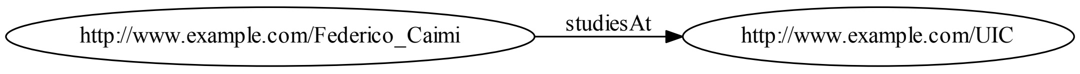

http://www.example.com/Federico_Caimi
studiesAt
http://www.example.com/UIC

Figure 1. Graph representation of a triple.

Listing 2.1. List of Statements

```
<http://www.example.com/Federico_Caimi, studiesAt, http://www.example.com/UIC>
<http://www.example.com/Federico_Caimi, plays, "Guitar">
<http://www.example.com/Federico_Caimi, type, http://www.example.com/Person>
<http://www.example.com/UIC, type, http://www.example.com/Organization>
<http://www.example.com/UIC, label, "University of Illinois at Chicago">
```

When multiple statements (triples) about the same resources are available, the graph becomes more expressive. Consider the following triples:

Figure 2 shows the graph representing the above RDF statements. In this example, more information is expressed about http://www.example.com/Federico_Caimi and http://www.example.com/UIC. The oval nodes are resources, while the rectangular ones are literals.

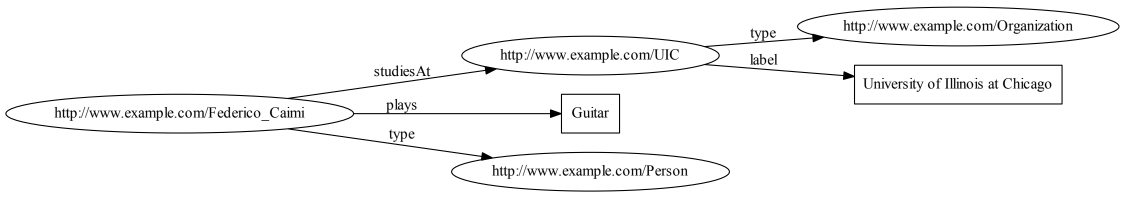

http://www.example.com/Federico_Caimi
studiesAt
plays
type
http://www.example.com/UIC
Guitar
http://www.example.com/Person
type
label
http://www.example.com/Organization
University of Illinois at Chicago

Figure 2. Graph representation of a set of triples.

Listing 2.2. Example of a SPARQL query

```
PREFIX ex:<http://www.example.com>
SELECT ?student WHERE {
        ?student ex:studiesAt ex:UIC
    } LIMIT 1000
```

RDF documentation is provided and mantained by the W3C. In particular, a description of the current status[1] of RDF and a detailed tutorial[2] can be found on-line.

2.5 SPARQL

RDF is also provided with a query language, the SPARQL Protocol and RDF Query Language (SPARQL), which has become the standard query language for RDF. Since the RDF is a directed graph-based model, SPARQL had to be defined as a graph-matching query language. Its syntax is similar to SQL because of the use of keywords such as SELECT, FROM, and WHERE, which are the same as in SQL.

A simple example of a SPARQL query is reported in Listing 2.2. The keyword $PREFIX$ is used to allow the use of a short name (prefix) instead of a complete URI in the rest of the query. In this case, the prefix *ex* will stand for the entire corresponding URI (`http://www.example.com`). The keyword SELECT is used to list the variables that have to be returned in the query solution, in this case only ?*student*. The keyword WHERE allows to

[1]http://www.w3.org/standards/techs/rdf#w3c_all

[2]http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

specify the pattern matching part of the query, in the form of triples, but also optional matching (OPTIONAL keyword), unions of patterns (UNION keyword), nesting and filtering of values (FILTER keyword) are allowed. The pattern in the example query asks for all the resources ?*student*, for which exists a statement whose property and object are respectively *ex*:*studiesAt* and *ex*:*UIC*. A natural language interpretation of this query would be: list all the resources which study at the University of Illinois at Chicago (all the UIC students). If the query is run against the simple model presented in Section 2.4, the resource `http://www.example.com/Federico_Caimi` will be returned. The language also supports solution modifiers, which modify the results returned by the pattern matching part in terms of ordering, number of results and other features. In the example query, LIMIT 1000 is used, and means that no more than one thousand results will be returned.

Many LOD datasets offer a SPARQL endpoint, which is an on-line service capable of answering to SPARQL queries. Those are extremely important because as the size of the cloud grows, the integration of different datasets has to be performed using multiple machines, which may communicate using endpoints. The SPARQL endpoints offered by datasets in the LOD cloud can be found on-line[1].

[1]http://www.w3.org/wiki/SparqlEndpoints

2.6 Evolution of the Cloud

In this section some statistics about LOD and its evolution will be reported. Detailed information can be found on the *State of the LOD cloud* [1] document. All this information is based on the LOD data set catalog[2] that is maintained on CKAN[3], a registry for open-licence datasets available on the Web in any format.

TABLE I

LINKED OPEN DATA STATISTICS

Domain	Number of datasets	Triples	%	(Out-)Links	%
Media	25	1,841,852,061	5.82 %	50,440,705	10.01 %
Geographic	31	6,145,532,484	19.43 %	35,812,328	7.11 %
Government	49	13,315,009,400	42.09 %	19,343,519	3.84 %
Publications	87	2,950,720,693	9.33 %	139,925,218	27.76 %
Cross-domain	41	4,184,635,715	13.23 %	63,183,065	12.54 %
Life sciences	41	3,036,336,004	9.60 %	191,844,090	38.06 %
User-generated content	20	134,127,413	0.42 %	3,449,143	0.68 %
All	295	31,634,213,770	100 %	503,998,829	100 %

Table I shows some statistics about LOD datasets categorized by domain. Most of the datasets cover a single specific domain (e.g., publications, life sciences, and media), while a

[1] http://www.lod-cloud.net/state/

[2] http://thedatahub.org/group/lodcloud

[3] http://www.ckan.net

13.3% of them are cross-domain. An example of a cross-domain dataset is DBPedia (7), which is a structured version of Wikipedia generated by crawling the infoboxes (i.e., tables included in some of the Wikipedia pages). The presence of cross-domain dataset is crucial for the successful interlinking of the cloud, which would be otherwise formed by disconnected subgraphs (also called data-islands). Other datasets range from media and entertainment (e.g., BBC program), geography and spatial information (e.g., GeoNames), government (e.g., Data.gov[1]), science (e.g., DBLP), and many others. In total, the LOD cloud is composed by 295 datasets containing more than 31 billions of triples.

Figure 3, Figure 4 and Figure 5 provide an effective graph visualization of the LOD cloud. The nodes are the dataset in the cloud, while the arcs represent the presence of links between them. These pictures were made available online[2] by Richard Cyganiak and Anja Jentzsch. The three figures show how rapidly the cloud is growing in the number of datasets, and it can be noted that it roughly doubles its size every two years.

[1]http://www.data.gov/semantic

[2]http://lod-cloud.net/

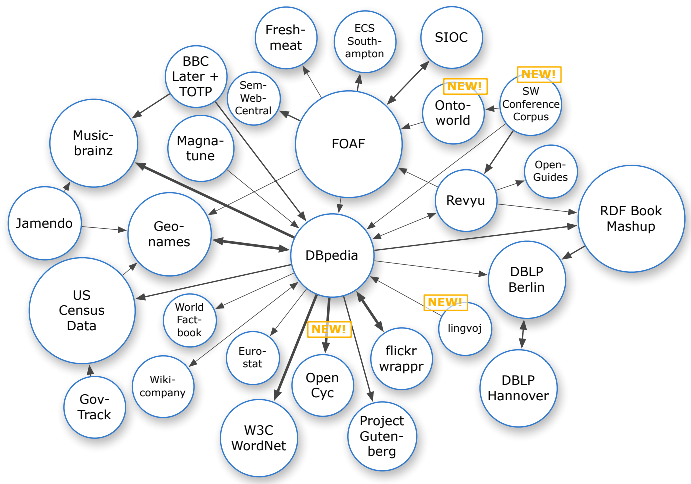


Figure 3. The Linked Open Data Cloud, November 2007 (8).

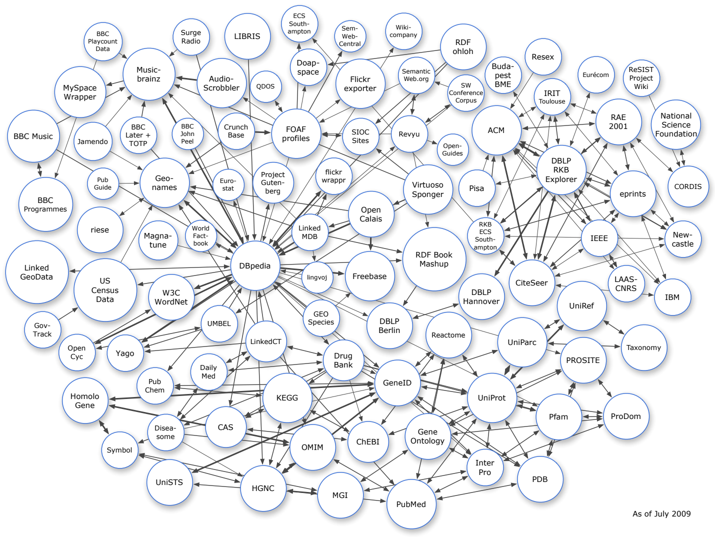


Figure 4. The Linked Open Data Cloud, July 2009 (8).

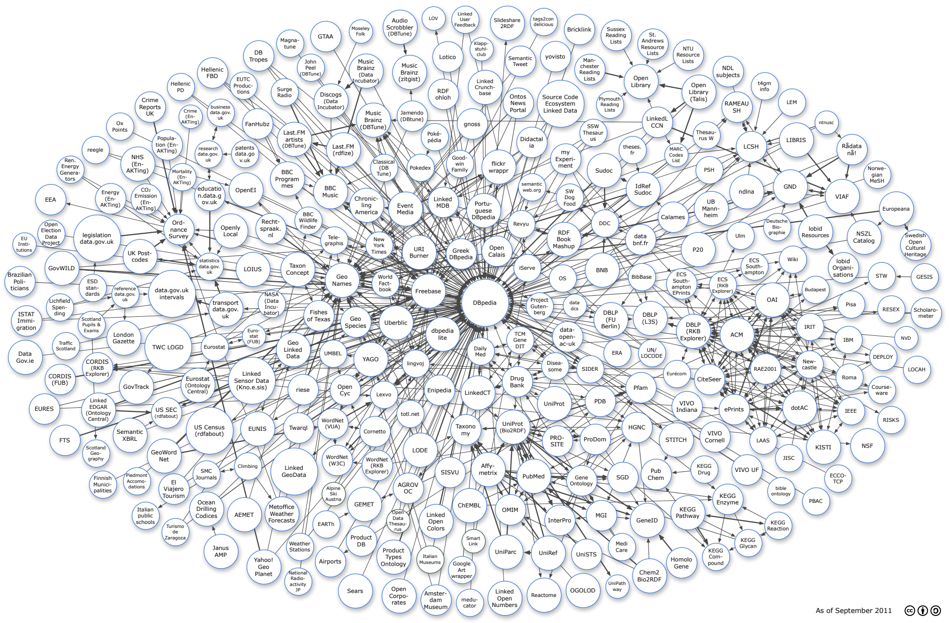


Figure 5. The Linked Open Data Cloud, September 2011 (8).

CHAPTER 3

ONTOLOGY MATCHING

3.1 Ontologies and OWL

The concept of ontology is defined as an explicit specification of a conceptualization (9). The term comes from philosophy, where it means the philosophical study of the nature of existence, entities, and the relations between entities. In computer science, ontologies are representations of a domain of interest based on the definition of concepts and the relationships between them. They are used to model some area of interest, enabling the sharing of knowledge and the development of applications which make use of it.

The main components of an ontology are:

- *Classes* are sets of real-world entities (e.g., Person, Place, and Organization).
- *Instances* are members of a particular class (e.g., John is an instance of Person).
- *Attributes* are characteristics or features describing entities or individuals (e.g. age for the type Person, for which a possible instantiation is John *hasAge* 22).
- *Properties* are relationships (binary predicates) between classes, individuals, or other properties (e.g. Person *worksFor* Organization).

Ontologies are characterized by their formalization, achieved with the use of semantics expressed using logic. An example is the *subclass* relationship. When a class A is declared as subclass of another class B, all the instances of A are also instances of B. Other semantics can be attached to user-defined properties, such as declaring a property as the *inverse* of another property.

Ontologies can either represent a single domain or multiple domains. Examples of the first category are domain-specific ontologies such as the Music Ontology or the Open Biomedical Ontologies. Multi-domain ontologies are also very popular as they try to cover any type of knowledge. Examples are Freebase and DBpedia, which is a Semantic Web version of Wikipedia. Since the concept of ontology is very general, also taxonomies such as Yahoo Categories and vocabularies such as Wordnet can be considered ontologies.

After many languages have been developed by separate groups for representing ontologies in a machine-readable format, these works have been unified under a common standard which is known as the Web Ontology Language (OWL), which allows to express all the previously described ontology features. It is compatible with the architecture of the World Wide Web, since resources are identified using URIs, and the preferred serialization is based on RDF/XML.

The expressive power of the language is enforced by a logic inference that can be performed to infer new statements from the ones explicitly included in the ontology. There are a number of available reasoners, which actually perform the previously discussed logic inference.

3.2 Problem statement

Ontology Matching (or Ontology Alignment) is defined as the process of finding correspondences between semantically related entities of different ontologies (10). It can be performed either automatically or semi-automatically, where in the latter case users take part in the process. The correspondences are called mappings, and the algorithms used to discover them are called matchers. Matchers can be either simple and take into account a single aspect of the concepts to be matched, or more complex combinations of simple matchers.

The problem can be formally defined as follows: Given a *source ontology* S and a *target ontology* T, a *mapping* is a triple $\langle c_S, c_T, r\rangle$ where $c_S \in S$ and $c_T \in T$ are concepts of the ontologies, and r is a semantic relation that holds between c_S and c_T. The relation that has been mostly investigated in the literature is the equivalence relation, but there exist also others such as the *subclass* relation.

A set of mappings is called an *alignment.* A *reference alignment* is an alignment found by experts, and it is used as a gold standard against which the accuracy of other alignments is measured in terms of precision and recall.

3.3 Evaluation

The growing interest in ontology matching by the scientific communities led to the development of many matching tools. As in every scientific field, evaluation methods are needed to help developers assess the quality of their systems and end-users understand which tool fulfills best their needs. The Ontology Alignment Evaluation Initiative [1] is an international initiative recognized by the ontology matching community as the standard for evaluation in this field. OAEI prepares a yearly evaluation event in which the systems are compared against several ontology matching tasks. Detailed results analysis is then performed by the organizers.

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

In order to evaluate a system in an ontology matching task, a *gold standard* has to be provided. This consists of the set of actual mappings that are usually discovered by a domain expert, and it is also known as reference alignment. The evaluation is then performed comparing the alignments generated by the system against the reference alignment. The comparison metrics selected for the evaluation are *precision* and *recall*, which originated in the field of information retrieval. Precision and recall are the ratio of the number of true positives to the retrieved correspondences and those expected (belonging to the reference alignment $|R|$) respectively. Since there is usally a trade-off between precision and recall, the two metrics are then combined in a final score (F-measure) which takes into account both precision and recall.

Precision. *Given a reference alignment R, the precision of the aligment A generated by an ontology matching system is computed as:*

$$P(A,R) = \frac{|R \cap A|}{|A|}$$

Recall. *Given a reference alignment R, the recall of the aligment A generated by an ontology matching system is computed as:*

$$R(A,R) = \frac{|R \cap A|}{|R|}$$

F-measure. *Given a reference alignment A, the F-measure of the aligment R generated by an ontology matching system is computed as:*

$$F(A,R) = \frac{2 \times P(A,R) \times R(A,R)}{P(A,R) + R(A,R)}$$

3.4 Ontology Matching techniques and systems

The field of ontology matching is a consolidated research area and a number of surveys are available (11; 12; 13). Figure 6 shows a taxonomy of the ontology matching techniques, obtained by slightly modifying the classification proposed in (14). The techniques used in ontology matching can be split into mainly three categories: similarity-based, reasoning-based, and instance-based.

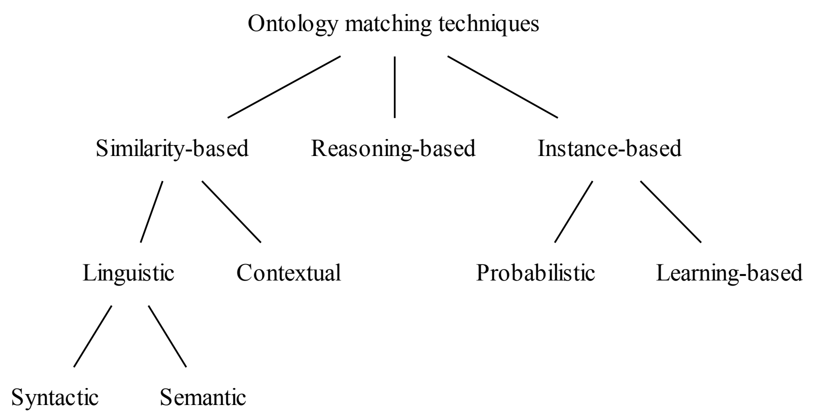


Figure 6. Classification of the Ontology Matching approaches (14).

Similarity-based. Similarity-based techniques compute the degree of similarity between concepts based on syntactic, linguistic or structural (contextual) features. The *syntactic similarity* involves the comparison of strings such as the name of concepts and the name and values of their properties. This is usually performed using string similarity metrics, automata or bit-parallelism. The *semantic similarity* considers also the meaning of the concepts being matched. It is computed using vocabularies such as WordNet, which contain relationships such as synonymy, hypernymy, and hyponymy. Semantic similarity metrics range from simple synonymy look-up, to the computation of distances between concepts in the graph built on hypernymy/hyponymy relationships. The *contextual similarity* encompasses all the metrics which make use of the concepts directly related to the concepts being evaluated. The most used techniques are graph algorithms that propagate the similarity of concepts to their neighbors in the ontology.

Reasoning-based. Reasoning based-techniques consist in modeling ontology matching as a logic inference problem. Starting from a set of high-quality mappings discovered by a similarity based matcher or defined by a user, new mappings are inferred using reasoning. Reasoning-based matchers are usually based on satisfiability or description logics. The alignments generated by this category of tools are consistent, meaning that they do not generate contradictions, an important property for the usability of these mappings in tasks such as data integration.

Instance-based. In some cases the information provided in schemata is not sufficient for determining matches between equivalent concepts. Some of these mappings can instead be inferred from the instance level, following the assumption that equivalent classes have similar instances. Instance-based matchers compare instances to derive a similarity between concepts. These approaches are based on set similarity measures, probability such as Bayesian theory, or machine learning techniques.

A number of ontology matching tools have been developed using the previously described techniques. An overview of these tools is reported in Table II, which shows the categories of the techniques used. Some of these tools implement a specific algorithm (e.g., BLOOMS and S-match, GLUE), while others provide configurable frameworks covering a wide range of methods (e.g., AgreementMaker, COMA++). Only a few of them provide a graphical user interface (GUI) to help users analyze the alignments produced (e.g., AgreementMaker, COMA++, SAMBO).

TABLE II

ONTOLOGY MATCHING TOOLS

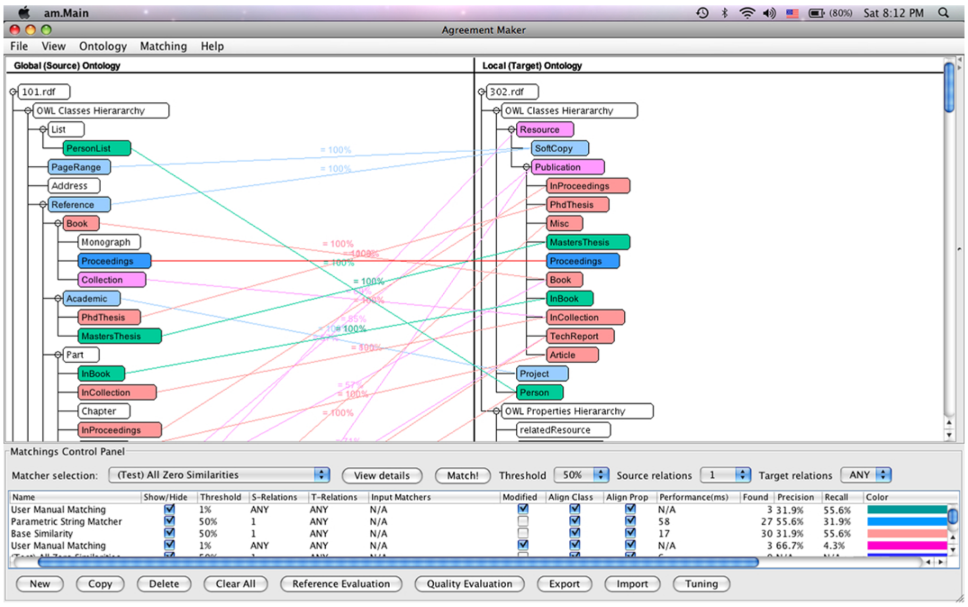
	Techniques Used				
Tool	Syntactic	Semantic	Contextual	Reasoning-based	Instance-based
AFlood (15)	✓	✓	✓		✓
AgreementMaker (1)	✓	✓	✓		✓
AROMA (16)		✓	✓		
ASMOV (17)	✓	✓	✓		
BLOOMS (18)		✓			
CODI (19)					
COMA++ (20)	✓	✓	✓		✓
DSSim (21)		✓		✓	
GLUE (22)					✓
LogMap (23)				✓	
RiMOM (24)	✓		✓		
SAMBO (25)	✓	✓	✓		
S-Match (26)		✓			

As can be noted in Table II, the great majority of the existing systems make use of similarity-based algorithms. The reason is that techniques such as string similarity metrics, token-based metrics, and graph propagating algorithms (e.g., similarity flooding) have proven to be effective in this context. Reasoning and logics have also been explored by tools such as DSSim and more recently LogMap. These tools perform particularly well in case of rich and axiomatized ontologies. Some tools make use of information encoded at the instance to improve their schema matching (e.g., AFlood, AgreementMaker, COMA++), while GLUE is entirely based on instances and uses a probabilistic approach to classify pair of concepts as match or non-match.

3.5 AgreementMaker

AgreementMaker is an extensible framework to perform, evaluate, and compare ontology matching algorithms (1). It has been designed for matching real-world schemas and ontologies, with particular attention on providing high configurability and an intuitive user interface, which is shown in Figure 7.

The system comprises several matching methods ranging from syntactic and semantic comparison of concepts, structural matching, and reasoning-based discovery of contradictions. All of these matching methods can be combined using a specific evaluation module. The quality of the generated alignment can be evaluated when the reference alignment is provided. The AgreementMaker has been used and tested in practical applications and in the Ontology Alignment Evaluation Initiative (OAEI) competition.



Matcher selection: (Test) All Zero Similarities Threshold 50% Source relations 1 Target relations ANY

Name	Show/Hide	Threshold	S-Relations	T-Relations	Input Matchers	Modified	Align Class	Align Prop	Performance(ms)	Found	Precision	Recall	Color
User Manual Matching	✓	1%	ANY	ANY	N/A	✓	✓	✓	N/A	3	31.9%	55.6%	🟩
Parametric String Matcher	✓	50%	1	ANY	N/A	☐	✓	✓	58	27	55.6%	31.9%	🟦
Base Similarity	✓	50%	1	ANY	N/A	☐	✓	✓	17	30	31.9%	55.6%	🟥
User Manual Matching	✓	1%	ANY	ANY	N/A	✓	✓	✓	N/A	3	66.7%	4.3%	🟪

Figure 7. Graphical User Interface as implemented in AgreementMaker (1).

The matching process in AgreementMaker is organized into three layers. The matchers of the first layer compare the concepts to be matched based on lexical features, such as string similarity metrics and TF-IDF vectors. The second layer uses the structure of the ontologies to refine the mappings discovered by the matchers in the first layer. In the third layer, a combination matcher aggregates the results generated by the previous matchers to provide a single final alignment.

3.5.1 AgreementMaker matchers

Lexical matchers. The Base Similarity Matcher (BSM) is a basic string matcher that computes the similarity between concepts by comparing all the strings associated with them. The Parametric String-based Matcher (PSM) is a more in-depth string matcher, which by default

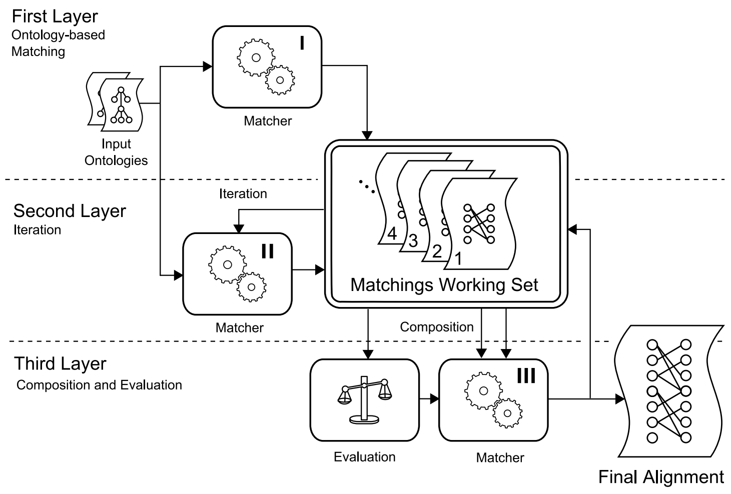


Figure 8. AgreementMaker three layer architecture (1).

is set to use a substring measure and an edit distance measure. The Vector-based Multi-Word Matcher (VMM) compiles a virtual document for every concept of an ontology, transforms the resulting strings into TF-IDF vectors and then computes their similarity using the cosine similarity measure. The Advanced Similarity Matcher (ASM) compares local names, providing better similarity evaluation in particular when compound terms are used. ASM outperforms generic string-based similarity matchers because it is based on a deeper linguistic analysis. All of these matchers can also use a lexicon, a data structure which keeps track of all the synonyms and definitions that may be provided in the ontologies, or in a third one called the mediator ontology.

Structural matchers. Structural matchers include the Descendants Similarity Inheritance (DSI) matcher. This matcher propagates the similarity of two nodes to their descendants. The

Group Finder Matcher (GFM) identifies groups of concepts and properties in the ontologies and assumes that two concepts (or properties) that belong to two groups that were not mapped by the input matcher will likely have different meanings and should not be mapped. The Iterative Instance Structural Matcher (IISM) is an iterative algorithm that compares concepts based on the properties defined on them and then properties based on the classes which make use of them till convergence. Also instances are taken into account to compare properties.

Combination matchers. The Linear Weighted Combination (LWC) receives as inputs the aligments generated by multiple matchers (e.g., the ones previously described) and, using a local confidence quality measure provided by the evaluation module, automatically assigns weights to each result computed by the input matchers. After this step, we have a single combined set of alignments that includes the best alignments from each of the input matchers.

CHAPTER 4

ONTOLOGY MATCHING FOR LOD

4.1 Introduction

The linked data paradigm identifies a set of best practices to publish and share data on the web (28). 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 (29).

A first problem to solve in order to match a set of input data and several LOD ontologies is to develop ontology matching systems that achieve a good trade-off between quality of the mappings and efficiency. As an example, good and efficient ontology matching techniques for LOD ontologies could improve the capability of tools such as DBpedia Spotlight (30), which extracts LOD entities from unstructured documents at runtime, to link the extracted data across several datasets.

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 (18). One of the reasons is that LOD ontologies have some peculiarities like poor textual descriptions, flat structure (e.g., GeoNames), cross-domain coverage, and use of concepts im-

Acknowledgement: The work in this chapter has been presented elsewhere (27).

ported from external ontologies. Another reason is that many ontology matching systems are better tailored to discovering equivalence relations. This is clearly a drawback in matching LOD ontologies because only few equivalence relations can be found among concepts in different ontologies. Since LOD ontologies are designed with the goal of maximizing their reuse, their overlap in terms of equivalent concepts is limited. For this reason relationships other than the equivalence relationship gain importance in the LOD domain. Therefore, the capability to discover subclass relations becomes crucial when the number of links among LOD sources increases.

Prior work in matching LOD ontologies has been performed by the BLOOMS system (18). 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 Ontology Alignment Evaluation Initiative (OAEI) competition (31; 32; 33).

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.

We extend AgreementMaker (1), an ontology matching system for ontologies expressed in a wide variety of languages (including XML, RDF, and OWL) that has obtained some of the best results in the OAEI competition (34) (35) (3), with the objective of testing its viability in the

LOD domain. Therefore, in this research 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 four ontology matching methods. A first category of matchers adopts a direct ontology matching approach, where concepts of a source and a target ontologies are compared: this category includes (i) an *Equivalence Mappings Expansion* method, which uses a set of equivalence mappings discovered with high confidence so as to infer subclass and superclass mappings, and (ii) a *Compound Noun Analysis* method, which discovers subclass and superclass mappings by analysing the compound local names that are often used to identify ontology concepts. A second category of matchers exploit third party ontologies used as mediators for the matching approach: this category includes (i) a *Distance-based Polysemic Lexical Comparison* method, which automatically annotates ontology concepts with possibly more than one lexical concepts taken from a background terminology, and compares these lexical annotations in order to discover subclass and superclass mappings, and (ii) a *Global Matching* method that infers subclass and superclass mappings by looking at how the concepts have been used in popular ontologies available on the Web. All these methods are new to our AgreementMaker system and are novel with respect to matching approaches proposed so far.

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 and average F-measure (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 in terms of execution time than BLOOMS and has the advantage that it consists of methods that can be integrated with an existing ontology matching system. To the best of our knowledge, AgreementMaker is currently the only system that achieves top performance both in the "classic" and LOD domains.

The chapter is organized as follows. Related work is discussed in Section 4.2. The proposed methods to improve ontology matching in the LOD domain are described in Sections 4.3 and 4.4. The experimental evaluation of the proposed approach, based on previously proposed reference alignments (18) is discussed in Section 4.5.

4.2 Related Work

In what follows we discuss related work whose main focus is on schema-level mappings (as opposed to instance-level matchings (36)). 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 (37). 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 (38). 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 (18). 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 (39), S-Match (40), and AROMA (16) that have participated in the Ontology Alignment Evaluation Initiative (OAEI) (32). 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 but much better results when discovering subclass mappings (32).

The ontology matching system BLOOMS+, which is an enhanced version of BLOOMS, has been used to align a set of LOD ontologies to the upper level ontology PROTON (41); however, the evaluation context is different since PROTON is a well-designed and well-described large ontology, more similar to the ontologies considered in more traditional ontology matching scenarios. In addition, there is no evidence that the efficiency of the system has been improved.

The SCARLET system introduces the idea of looking for clues in background ontologies available on the Web to discover mappings between two ontologies (42); our mediator-based matching algorithms present some similarities to the SCARLET approach, although there are significant differences. SCARLET searches the local names of the concepts on external Web ontologies and uses the subclass relations defined in the external ontologies to derive new mappings. Our DPLC algorithm uses polysemic lexical annotations and a probabilistic scoring function to determine whether a mapping has to be established between two concepts while SCARLET uses only logic-based rules. Our GM technique looks for useful information about the concepts on the Web, which is at the core of the SCARLET approach; our approach is very different though: by looking for the concepts' URIs, we consider only the external use of the concepts that have to be matched (instead of other concepts with similar names), and we look into a pool of trusted Web ontologies in order to achieve high precision. Finally, SCARLET has not been evaluated in the LOD domain, which presents several new challenges to ontology matching systems.

Significant efforts have been recently carried out to support ontology matching systems with more accurate lexical annotation methods; these efforts considered both the interpretation of compound names (43) and the disambiguation problem (44). A method for interpreting endocentric compound names has been proposed to include a terminology concept denoted by a compound name in an existent terminology. The method used in our CNA algorithm is inspired by this method; however, (43) establish semantic relations between the terminology concepts, while we use the interpretation of compound names in order to directly infer subclass relations between the ontology concepts. Word sense disambiguation techniques have been proposed to handle polysemic lexical annotations, and in particular, to assign a probability score to each annotation associated with an ontology concept (44). A set of probabilistic semantic relations is inferred among the ontology concepts based on this score. In our DPLC algorithm, word sense disambiguation is used to filter the set of lexical annotations of the ontology concepts. Subclass mappings between concepts (interpreted according to the usual OWL semantics) are then inferred by comparing the whole sets of lexical annotations and obtaining a polysemic subclass evidence score; to the best of our knowledge, the approach adopted in the DPLC method, which takes explicitly into account that hyponym and subclass relations have different semantics, defining the hyponym-to-subclass conversion factor and a distance-driven score, is an original contribution.

4.3 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. We use the *Advanced Similarity Matcher (ASM)* to compute the similarity $sim(c_S, c_T)$ between two concepts c_S and c_T; ASM is a very efficient matcher that evaluates the string-based similarity between two concepts using their local names and their labels (35). Two concepts are considered equivalent when their similarity is higher than a threshold $th^{\equiv}$.

We slightly modified ASM with the addition of detecting different spellings of the same word, e.g., $(Organization, Organisation)$ and $(Theater, Theatre)$. These apparently small differences are not always captured by string similarity algorithms, and very simple linguistic rules lead to a significant improvement in the capability to discover equivalence mappings.

4.3.1 Equality Mappings Expansion (EME)

The Equivalence Mappings Extension matcher computes the similarity values between all the possible pairs of concepts and stores the results in a similarity matrix.

For each pair of concepts and a threshold $th^{\equiv}$, such that $sim(c_S, c_T) \geq th^{\equiv}$, the mapping $\langle c_S, c_T, \equiv \rangle$ is included in the set of equivalence mappings $EME^{\equiv}$

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

The selection of the equivalence mappings must be even more accurate in the LOD domain than what is required in traditional ontology matching scenarios (33); this is a consequence of the importance of subclass and superclass mappings. When equivalence mappings are used for inferring subclass mappings, 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 (35).

4.3.2 Compound Names Subclass Matcher (CNS)

When the names of the concepts in the ontologies are *compound* (i.e., formed by multiple words), matchers such as ASM, which is highly specialized on the equivalence relation, are not able to capture other relations that are implicitly specified in the compound. For example, *SportsEvent* denotes a narrower concept than *Event*, thus a subclass relation should be directly inferred from their names (under the assumption that the two concepts are sharing the same meaning of the term *Event*).

A classification of compounds in English has been proposed and is shown in Figure 9 (45). The majority of the compounds shows a *modifier-head* structure, where the *head*, the most

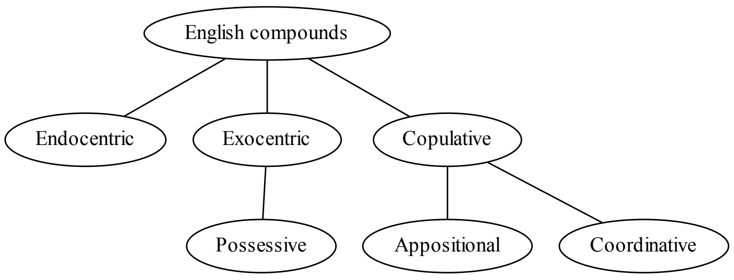


Figure 9. Classification of compounds (i.e., compound words) in English.

important unit, usually determines the gender, part-of-speech, and the general meaning. This general meaning is then modified by the other terms, restricting the meaning of the compound to a more specific concept. In the previous example, *Event* is the head and *Sports* is a modifier.

When the head appears inside the compound, these compounds are referred to as *endocentric*. *SportsEvent* is clearly an example of this category. In case the head is outside (i.e., it doesn't occur in the terms forming the compound) they are called *exocentric*. Examples of this category are the *possessive* compounds, which denote entities characterized by the properties expressed in the compound (e.g., greybeard and loud-mouth are instances of person, instead of respectively beard and mouth). Another category is called *copulative*, whose compounds do not have a head as the terms equally contribute to the meaning. In case they specify entities that are instances of multiple classes (e.g., poet-translator), they are called *appositional* compounds, while if they specify relations between the terms involved (e.g., doctor-patient gap), they are called *coordinative*.

When the names of the concepts to be matched are compound, we use a best effort approach that produces good results in practice. We consider only *endocentric* compounds, since they are the vast majority in English and cover up to 78% of the compounds used in schema and ontology concept names according to a recent study (43). For these compounds, we are interested in detecting the head, as it provides meaningful information for inferring subclass relations. For this purpose, we use a very simple rule which works well in English: the head of a compound always occurs on the right-hand side (46). We use this knowledge to extract the heads and then attempt to find correspondences between these main nouns and the names of the concepts using ASM; based on these correspondences we extrapolate subclass and superclass mappings. In particular, let $head(c)$ be the head of a compound denoting the concept c. If $sim(head(c_S), c_T)) \geq th^{\equiv}$, then
$\langle c_S, c_T, \sqsubseteq \rangle \in CNA^{\sqsubseteq}$; if $sim(c_S, head(c_T)) \geq th^{\equiv}$, then $\langle c_S, c_T, \sqsupseteq \rangle \in CNA^{\sqsupseteq}$.

4.4 Mediator-based Mapping Discovery

We consider two different types of mediators, namely background terminologies and Web ontologies. *Web ontologies* are ontologies represented in a semantic Web language (e.g., RDFS or OWL (29)) and available on the Web.

A *background terminology* is any knowledge structure organized in a concept hierarchy; a background terminology can be represented by a triple $O^T = (C, T, \preceq)$, where C is a set of *terminology concepts*, T is a set of *terms* (also called *labels*) and $\preceq$ is a *hyponymy* relation defined by a partial order over C; given two terminology concepts w_1 and w_2, the relation $w_1 \preceq w_2$ means that w_1 is *more specific* than w_2; in this case we can say that w_1 is a hyponym of w_2, and, conversely w_2 is a hypernym of w_1. Each concept is associated with a set of terms, which are synonyms of the concept (*synonyms*). Conversely, a term can be associated with one or more concepts (*polysemy*).

Background terminologies encompass knowledge structures such as lexicons and other taxonomies where multiple labels are associated with a concept. We use WordNet as background terminology, whose concepts are called synsets, each one usually associated with more than one term.

Although background terminologies and Web ontologies share a similar hierarchical structure, the semantics of the relations on which their respective hierarchies are based is different: while in a Web ontology $c_1 \sqsubseteq c_2$ means that c_1 is subclass of c_2, i.e., every instance of c_1 is also an instance of c_2, in a terminology the hyponym relation can not be assumed to have such formal semantics; in other words, it can be the case that $w_1 \preceq w_2$ and $w_1 \not\sqsubseteq w_2$. Furthermore, although concepts in Web ontologies are often associated with labels, Web ontologies do not handle polysemy and synonymy. The consideration of these important differences leads to the design of different matching methods depending on the type of mediator.

4.4.1 Distance-based Polysemic Lexical Comparison (DPLC)

We compare every concept of the source ontology with every concept in the target ontology: the key idea of our algorithm is that given a source concept c_S lexically annotated with a terminology concept w_S and a target concept c_T lexically annotated with w_T we can add a subclass mapping $\langle c_S, c_T, \sqsubseteq \rangle$ when $w_S \preceq w_T$ holds in the terminology (or, conversely, a superclass mapping when $w_T \preceq w_S$ holds in the terminology).

However the simple idea sketched above encounters two problems:

1. It can be very difficult to annotate an ontology concept with exactly one terminology concept for at least two reasons: the information needed to automatically disambiguate among several candidate annotations can be inadequate, e.g., in Figure 10 there are three sets of synonyms associated with the concept *Person* (highlighted in blue), and there is no empirical evidence of one being more appropriate than the others, therefore they are all considered in the following steps; the terminology can provide several concepts having similar meaning, which can all be considered correct annotations for the ontology concept (44). In Figure 10, the two sets of synonyms associated with the concept *Actor* (highlighted in red) are very similar and can both be considered correct annotations for the ontology concept. In other words, the matching algorithm has to handle the case in which concepts are associated with multiple lexical annotations;
2. In general, the semantics of the relation $\preceq$ is different from the semantics of the subclass relation $\sqsubseteq$; therefore, the more distant two terminology concepts are in the terminology hierarchy, the higher the probability that they can not be considered one subclass of the other, and also the higher the probability that the inferred mapping among the ontology concepts is wrong. The length of the path (distance) on the terminology hierarchy between two lexical annotations can be used to give a confidence score to the inferred mapping.

We addressed the above mentioned problems with an algorithm consisting of three steps.

Step 1. Polysemic Lexical Annotation with Word Sense Disambiguation: Each concept in the source (respectively, target) gets associated with a set of concepts in the background

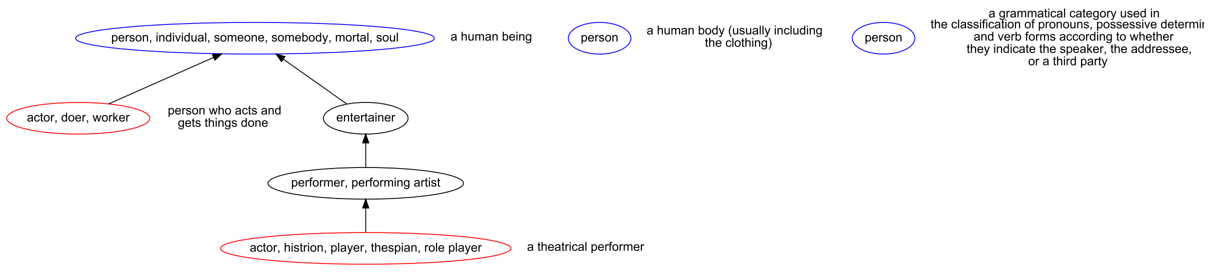

person, individual, someone, somebody, mortal, soul
a human being
person
a human body (usually including the clothing)
person
a grammatical category used in the classification of pronouns, possessive determi and verb forms according to whether they indicate the speaker, the addressee, or a third party
actor, doer, worker
person who acts and gets things done
entertainer
performer, performing artist
actor, histrion, player, thespian, role player
a theatrical performer

Figure 10. WordNet synsets for the ontology concepts *Actor* (source) and *Person* (target). Each ellipse represents a WordNet synset with its set of terms. The arrows represent the hyponym relation.

terminology. This association is made through the concept labels: every time a label matches exactly a concept in the source (respectively, target) ontology, then that terminology concept becomes associated with the source (respectively, target) concept. Given a concept c, the set of the terminology concepts associated with it is denoted BST_c (for *Background Synonym Terminology*). In Figure 10 and Figure 11, two graphs involving the terminology concepts are shown, where the elements of BST_{c_S} (respectively *Actor* and *Agent*) are highlighted in red and the BST_{c_T} concepts (respectively *Person* and *Group*), are highlighted in blue.

However, to improve the accuracy of lexical annotation, we apply *word sense disambiguation* techniques (44). Some concepts in the ontologies have a textual description (usually included in *rdfs:comment*), while in WordNet all the sets of synonyms are described in a *definition*. We create a virtual document associated with every concept/set of synonyms, which are then compared using a vector space model approach based on the cosine similarity measure, after we performed stop-words removal and stemming. These techniques were already implemented in one of our matchers called Vector-based Multi-word Matcher (VMM), extensively used in the OAEI competition. In addition to comments and definitions, we also included in the documents also the first level of the concepts' superclasses, since they proved to be particularly useful for disambiguation. After the similarity values are computed, the actual disambiguation is performed. If the degree of similarity between a concept and a related sets of synonyms is *significantly* high (higher than a threshold), only those will be kept for further processing, thus narrowing the set BST_c into a subset $\overline{BST}_c$. The threshold has been experimentally set to 0.3, a high value for cosine similarity. This leads to an improvement in precision, while not penalizing the recall.

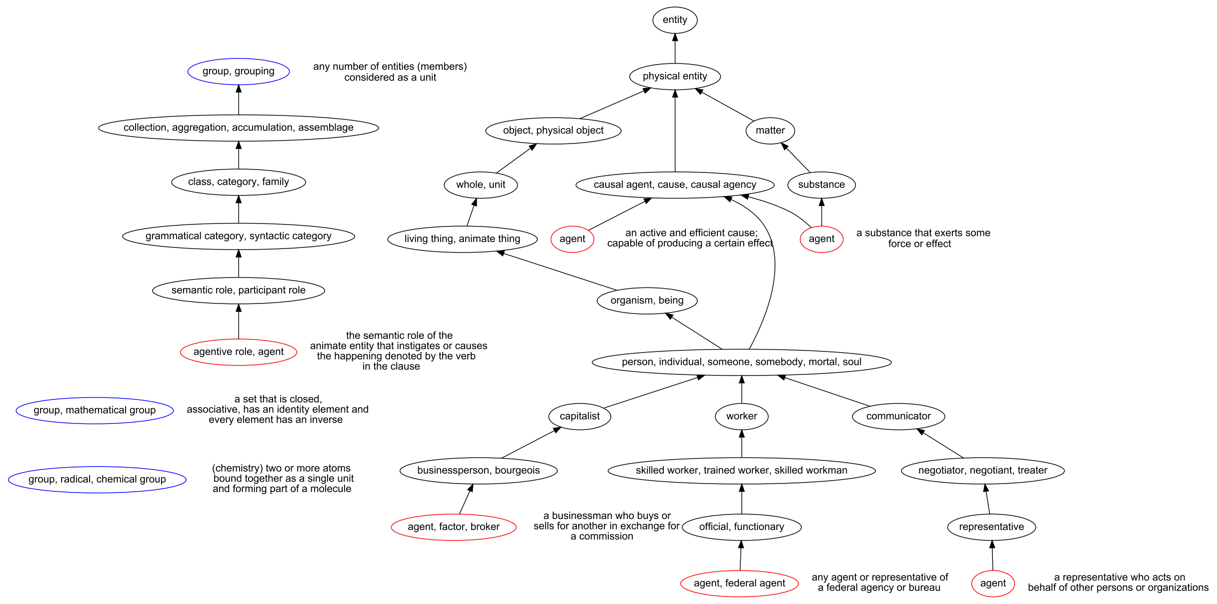

entity
physical entity
object, physical object
matter
whole, unit
causal agent, cause, causal agency
substance
living thing, animate thing
agent
an active and efficient cause; capable of producing a certain effect
agent
a substance that exerts some force or effect
organism, being
person, individual, someone, somebody, mortal, soul
capitalist
worker
communicator
businessperson, bourgeois
skilled worker, trained worker, skilled workman
negotiator, negotiant, treater
agent, factor, broker
a businessman who buys or sells for another in exchange for a commission
official, functionary
representative
agent, federal agent
any agent or representative of a federal agency or bureau
agent
a representative who acts on behalf of other persons or organizations
group, grouping
any number of entities (members) considered as a unit
collection, aggregation, accumulation, assemblage
class, category, family
grammatical category, syntactic category
semantic role, participant role
agentive role, agent
the semantic role of the animate entity that instigates or causes the happening denoted by the verb in the clause
group, mathematical group
a set that is closed, associative, has an identity element and every element has an inverse
group, radical, chemical group
(chemistry) two or more atoms bound together as a single unit and forming part of a molecule

Figure 11. WordNet synsets for *Agent* and *Group*.

Step 2. Background Hypernym Terminology Construction: Each concept in the source (respectively, target) gets associated with a set of hypernyms from the background terminology. This association is made through the previously built sets of synonyms. Given a concept c, we consider each concept in $\overline{BST}_c$ and extract its hypernyms in the background terminology. Finally, we perform the union all such sets, thus obtaining a set for each concept c denoted BHT_c (for *Background Hypernym Terminology*).

Step 3. Mapping Inference: We use the sets obtained in the previous two steps to build the sets of subclass and superclass mappings denoted respectively by $DPLC^{\sqsubseteq}$ and $DPLC^{\sqsupseteq}$.

Our mediator-based approach relies on the possibility to convert hypernym relations into subclass relations, the latter ones interpreted according to their well-known OWL semantics. We start by defining a *hyponym-to-subclass conversion factor* (*hsc*) as the probability that a source concept c_S is a subclass of a target concept c_T, given that there exist two terminology concepts w_S and w_T that are correct annotations respectively for c_S and c_T, such that w_S is a *direct hyponym* of w_T. This can be expressed by the following formula:

$$hsc = P(c_S \sqsubseteq c_T | w_S \preceq^1 w_T) \tag{4.1}$$

where $\preceq^1$ denotes the direct hyponymy relation. We note that the *hsc* factor can change depending on the terminology. We empirically estimated $hsc = 0.9$ in WordNet on the basis of a number of samples.

Now we can define a metric to assess the confidence degree under which two lexical annotations provide evidence for the existence of a subclass mapping between a source and a target concept; we call this confidence degree the *single-annotation subclass evidence score.* The metric is based on the propagation of the *hsc* factor when there exists a *hyponym* relation between two terminology concepts with a distance greater than one between them.

Let $dist(w_S, w_T)$ be the length of the path on the hyponym hierarchy connecting two terminology concepts w_S and w_T. The *single-annotation subclass evidence score* $saScore(c_S^{w_S}, c_T^{w_T})$ of two concepts c_S and c_T given two terminology concepts w_S and w_T, respectively associated with c_S and c_T, is defined as the probability that $c_S \sqsubseteq c_T$ given $w_S \preceq w_T$ according to the following formula:

$$saScore(c_S^{w_S}, c_T^{w_T}) = \begin{cases} P(c_S \sqsubseteq c_T | w_S \preceq w_T) & \text{if } w_S \preceq w_T \\ \\ 0 & \text{if } w_S \npreceq w_T \end{cases} \tag{4.2}$$

where $P(c_S \sqsubseteq c_T | w_S \preceq w_T)$ can be resolved according to the following equation:

$$\begin{aligned} P(c_S \sqsubseteq c_T | w_S \preceq w_T) &= \prod_{i=1}^{dist(w_S,w_T)-1} P(c_S \sqsubseteq c_T | w_i \preceq^1 w_{i+1}), \\ &\quad \text{where } w_1 = w_S,\ w_{dist(w_S,w_T)} = w_T \\ &= hsc^{dist(w_S,w_T)} \end{aligned} \tag{4.3}$$

Finally we have to consider that according to the polysemic lexical annotation strategy adopted, every ontology concept is annotated with possibly more than one terminology concept. We therefore define a *polysemic subclass evidence score* that assesses the confidence degree at which a source concept can be considered subclass of a target concept by aggregating the evidences provided by all the lexical annotations. The polysemic subclass evidence score $polyScore(c_S, c_T)$ is defined as follows:

$$polyScore(c_S, c_T) = \frac{\sum_{w_i \in \overline{BST}_{c_S}, w_j \in BHT_{c_T}} saScore(c_S^{w_i}, c_T^{w_j})}{log(|BHT_{c_T}|)} \tag{4.4}$$

The aggregation function sums all the single-annotation subclass evidence scores and adopts a normalization factor in the denominator based on the size of the Background Hypernym Terminology BHT_{c_T} associated to the target concepts. In fact, the bigger this set is, the higher the probability of finding matchings between sets of synonyms and hypernyms. The size of these sets grows rapidly when the depths of the synonyms increase. We therefore smooth the size using a logarithmic function.

In Figure 10, there are two paths connecting the source and target terminology concepts. The first (length one) gets associated with an *saScore* of 0.9, while the second one (length three) with a value of 0.729. These values are then summed and normalized applying the natural logarithm to the size of BHT_{c_S}, which in this case is 10 (the hypernyms of the matched person are not shown for simplicity). The overall score (0.707) is above the threshold we experimentally set, and therefore will be included in $DPLC^{\sqsubseteq}$.

In Figure 11, there is only one path connecting the source and target terminology concepts, even though the graph is significantly bigger than in the previous example. This path (length five) gets associated with an *saScore* of 0.59. After normalization the overall score obtained (0.186) is below the threshold, and therefore it will not be included in $DPLC^{\sqsubseteq}$.

The polysemic subclass evidence score can be adopted to infer both subclass and superclass relations. In fact, given a subclass score threshold $th^{\sqsubseteq}$ the set of subclass mappings and superclass mappings returned by this matcher can be defined as follows:

$$\begin{aligned} DPLC^{\sqsubseteq} &= \{\langle c_S, c_T, \sqsubseteq\rangle | polyScore(c_S, c_T) \geq th^{\sqsubseteq} \text{ and} \\ &\qquad polyScore(c_S, c_T) \geq polyScore(c_T, c_S)\} \\ DPLC^{\sqsupseteq} &= \{\langle c_S, c_T, \sqsubseteq\rangle | polyScore(c_T, c_S) \geq th^{\sqsubseteq} \text{ and} \\ &\qquad polyScore(c_T, c_S) \geq polyScore(c_S, c_T)\} \end{aligned} \tag{4.5}$$

4.4.2 Global Matcher (GM)

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. This does not usually happen in more traditional ontology matching scenarios where ontologies are not much interlinked. The *Global Matching* (GM) technique is introduced to improve matching over external concepts, in order to consider this peculiarity of LOD ontologies.

The GM technique is based on the consideration that several external concepts used in LOD ontologies, such as *wgs84_pos:SpatialThing* in the GeoNames ontology, are used across different ontologies, which could provide useful information in discovering additional mappings such as between *dbpedia:Person* and *wgs84_pos:SpatialThing.* One could arrive to this mapping by knowing that *foaf:Person* has been defined as subclass of *wgs84_pos:SpatialThing* elsewhere.

Our GM technique is defined as follows. For each concept c_S in S that has been imported from an external ontology E, we search across several LOD ontologies 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 $sim(x_E, c_T) \geq th^{\equiv}$ (respectively, $sim(c_S, x_E) \geq th^{\equiv}$) then $\langle c_S, c_T, \sqsupseteq\rangle \in GM^{\sqsupseteq}$ (respectively, $\langle c_S, c_T, \sqsubseteq\rangle \in GM^{\sqsubseteq}$).

The external ontologies that we use to search for external concepts are listed in a registry. We included in the registry ontologies available on the Web that either have been defined by a recognized institution such as the W3C consortium (e.g., Event Ontology, [1] WGS84 Geo Positioning, [2] and Media Ontology [3]) or are well known and used by a wide community of users

[1]http://motools.sourceforge.net/event/event.html

[2]http://www.w3.org/2003/01/geo/wgs84_pos

[3]http://www.w3.org/TR/mediaont-10/

(e.g., DBPedia,[1] FOAF, [2] and Freebase[3]). These ontologies, which often import third party ontologies to reuse their most important concepts, provide good background knowledge for the GM technique.

4.5 Results

Table III lists the ontologies that we have used for our experiments, which are the same that were considered by the BLOOMS system[4] (18), as no benchmark has been otherwise set for the LOD domain. Table III shows the number of concepts in the ontologies and the number of external ontologies that they import. The evaluation settings consist of seven matching tasks, involving different types of comparisons. For example, the Music Ontology and the BBC Program ontology are both related to entertainment, whereas some other comparisons involve general purpose ontologies, such as DBpedia.

We first compare the results obtained by our system to the results obtained by other systems; then, we provide an in-depth analysis of each matcher used in our system; we finally discuss some significant issues concerning the alignment of LOD ontologies that we believe of general interest for future research in this domain.

---

[1]http://dbpedia.org/ontology/

[2]http://xmlns.com/foaf/spec/

[3]http://rdf.freebase.com/rdf/base.fbontology

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

TABLE III

ONTOLOGIES IN THE EXPERIMENTAL DATASET.

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

4.5.1 Comparison with other systems

Table IV shows the comparison between the results obtained by AgreementMaker and the results previously obtained for the S-Match, AROMA, and BLOOMS ontology matching systems. We are omitting the baseline results (Alignment API) and the results of other systems (OMViaUO, and RiMoM) because their results are not competitive (18).

As can be seen in Table IV, 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. In particular, non-trivial mappings are discovered by our global matching technique described in Section 4.3, which allows us to find mappings using external ontologies and to propagate them through the subclasses of the involved concepts.

TABLE IV

COMPARISON BETWEEN AGREEMENTMAKER AND OTHER ONTOLOGY MATCHING SYSTEMS.

	S-Match			AROMA			BLOOMS			AgreementMaker		
Matching Task	Prec	Rec	F-m	Prec	Rec	F-m	Prec	Rec	F-m	Prec	Rec	F-m
FOAF-DBp	0.11	0.40	0.17	0.33	0.04	0.07	0.67	0.73	0.70	**0.80**	**0.90**	**0.85**
GN-DBp	0.23	**1.00**	0.37	0.00	0.00	0.00	0.00	0.00	0.00	**0.32**	**0.73**	**0.44**
MO-BBC	0.04	0.28	0.07	0.00	0.00	0.00	**0.63**	**0.78**	**0.70**	0.56	0.27	0.36
MO-DBp	0.08	0.30	0.13	0.45	0.01	0.02	0.39	**0.62**	0.48	**0.87**	0.46	**0.60**
SWC-AKT	0.06	0.40	0.10	0.38	0.03	0.06	0.42	**0.59**	**0.49**	**0.52**	0.41	0.46
SWC-DBp	0.15	0.50	0.23	0.27	0.01	0.02	0.70	**0.40**	**0.51**	**0.71**	0.39	0.50
SIOC-FOAF	0.52	0.11	0.18	0.30	0.20	0.24	0.55	**0.64**	**0.59**	**0.71**	0.45	0.55
Average	0.17	0.43	0.24	0.25	0.04	0.07	0.48	**0.54**	0.51	**0.64**	0.52	**0.57**

TABLE V

COMPARISON BETWEEN AGREEMENTMAKER AND ITS OLDER VERSION

	AgreementMaker (2010)			AgreementMaker		
Matching Task	Prec	Rec	F-m	Prec	Rec	F-m
FOAF-DBp	0.72	0.80	0.76	**0.80**	**0.90**	**0.85**
GN-DBp	0.26	0.68	0.38	**0.32**	**0.73**	**0.44**
MO-BBC	0.48	0.16	0.24	**0.56**	**0.27**	**0.36**
MO-DBp	0.62	0.40	0.49	**0.87**	**0.46**	**0.60**
SWC-AKT	0.48	**0.43**	0.45	**0.52**	0.41	**0.46**
SWC-DBp	0.58	0.35	0.44	**0.71**	**0.39**	**0.50**
SIOC-FOAF	0.56	0.41	0.47	**0.71**	**0.45**	**0.55**
Average	0.53	0.46	0.49	**0.64**	**0.52**	**0.57**

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, as the actual categories are encoded in properties at the instance level. However, S-Match has a perfect recall (100%), though precision is low (20%). The use of our global matching technique is the main reason why AgreementMaker outperforms all the other systems.

Task 3. For the Music Ontology–BBC program task, BLOOMS obtains the best results, with AgreementMaker second. BLOOMS uses Wikipedia while we use WordNet, a generic background ontology. Wikipedia is very well suited for this kind of ontologies, because it covers the specific vocabulary of the ontologies being matched.

Task 4. For the Music Ontology–DBpedia matching task, and in contrast with the previous task, our results are better than those of BLOOMS in terms of F-measure. While BLOOMS achieves slightly higher recall, the precision achieved by AgreementMaker is significantly higher. Our system presents only mappings on which it is very confident, thus favoring precision, while BLOOMS clearly favors recall. The next best system, S-Match, obtains a reasonable recall (30%), albeit at the cost of very low precision (6%).

Task 5 For the Semantic Web Conference–AKT Portal matching task in the scientific publications domain, we notice again that BLOOMS favors recall while AgreementMaker favors precision. S-Match again 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 and AgreementMaker achieve very similar good results. The conference domain is the same used in the OAEI com-

TABLE VI

EXECUTION TIMES (IN SECONDS) OF THE MATCHING PROCESS (LOADING, SIMILARITY-BASED, MEDIATOR-BASED, AND TOTAL).

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

petition, on which both the systems perform well. S-Match has an 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. AgreementMaker leads in precision followed by BLOOMS and S-Match (respectively, 71% , 52%, and 56%), while BLOOMS significantly leads in recall because it is based on Wikipedia.

Table VI shows the total execution times of the AgreementMaker matching process in the seven tasks as well as the times for the different subtasks, namely, loading, mapping discovery using the similarity-based (SB) method and using 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.

4.5.2 Analysis of Matchers Effectiveness

Figure 12 shows the results achieved by our system. The *Global Matching* (GM) technique we introduced leads to the best single matcher results, because external concepts usually have a high number of subclasses. In some of the evaluation tasks, most of the mappings involve external concepts.

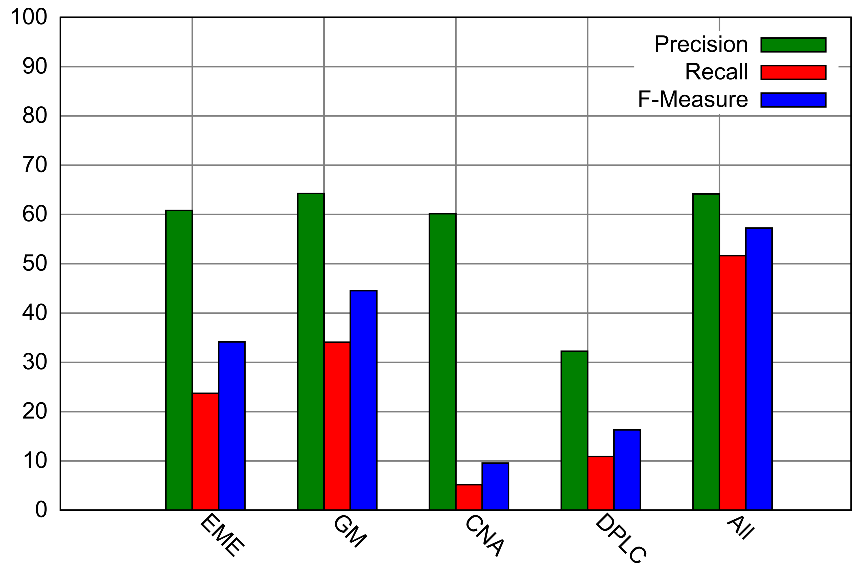


Figure 12. Analysis of the effectiveness of each matcher.

The *Equivalence Mappings Expansion* (EME) is the second best in recall because even from a small set of equality mappings a significant number of subclass relationships can be inferred. Our *Distance-based Polysemic Lexical Comparison* (DPLC) is the third best performing matcher. This is a matcher that helps in improving the overall recall, while it provides lower precision than the other methods. Only 48% of the concepts in the ontologies can be found in WordNet, and some of the mappings in the reference alignment are between concepts whose names are compound and do not appear in the WordNet ontology. For these reasons, in order to provide a significant recall, we have to sacrifice some precision. However, we note that most of the mappings discovered by this matcher are not found by other matchers, which makes this matcher an important contributor to the overall results. The *Compound Noun Analysis* (CNA) is a precise method that allows us to slightly improve the recall, while not penalizing precision. Most of the compounds are *endocentric*, but the heads extracted from the source compounds can not be frequently matched with target concepts in order to infer the subclass mappings, which keeps the recall of this matcher quite low.

The combination of all our approaches (shown as *All*) is the best overall. Our matchers are "orthogonal" in the sense that they compare different features of the ontologies, and therefore the union of the generated correspondences is better than the sets of mappings generated by the individual matchers. This is apparent in Figure 12, where the precision achieved by the overall system is close to the maximum precision of the single matchers, while the system recall is significantly higher than the recall of the individual matchers.

This phenomenon is aligned to what happens in traditional ontology matching, where the results of several matchers, which compare different ontology features (e.g., syntactic, lexical, structural), are combined by a combination matcher that significantly improves the final alignment (34).

4.5.3 Discussion of the results.

Matching LOD ontologies is different from matching ontologies in more traditional scenarios, such as the ones addressed in the OAEI competition. The ontologies are more subject to real-world characteristics such as heterogeneity and presence of noise. Therefore, part of the information that is required by traditional ontology matching tools is often not available.

Mappings involving the subclass relation become extremely important in order to integrate the datasets associated with these ontologies, since only few equivalence mappings can be established. Subclass mappings are more subjective than equivalence mappings, and this makes the creation of an agreed-upon gold standard more complicated. Moreover, the subclass relation is intrinsically many-to-many, and therefore imposes fewer constraints on the characteristics of the final alignment with respect to equivalence relation; such additional constraints are often helpful to improve the results because the selection of mappings in a one-to-one setting can be solved as an optimization problem (47).

The adoption of external lexical resources such as WordNet and Wikipedia is crucial. The use of such ontologies, and of other mediator ontologies as in the case of our system, is the reason why BLOOMS and AgreementMaker achieve better results than the other tools. It is hard to find resources covering a substantial part of the concepts and also containing hierarchies whose semantics is compliant with the subclass relation. The results show that WordNet has less coverage, but its *hypernym* relation is suitable for this task, while Wikipedia offers more coverage, but the semantics of the *subcategory* relation is less appropriate for deriving the semantics of the subclass relation, leading to lower precision.

One of the possible future developments for AgreementMaker would be the integration of the Wikipedia categories, using an approach similar to BLOOMS. BLOOMS downloads Wikipedia pages related to ontology concepts using an API and then follows the links in the page to build the categories hierarchy. This process is time consuming, and would be the bottleneck when integrated in AgreementMaker. However, an advantage of using the API is that the most up-to-date version of Wikipedia is always used. We believe that the quality of the alignments generated by AgreementMaker can be improved by using Wikipedia categories. These have not been integrated yet, because the state-of-the-art approach used by BLOOMS does not fulfill our performance requirements. Future investigation for AgreementMaker would be finding an efficient way to integrate the Wikipedia categories, so as to improve the quality of the alignments without sacrificing the time performance.

CHAPTER 5

INSTANCE MATCHING

5.1 Problem Statement

Instance Matching is the problem of deciding whether instances belonging to different data sources are referring to the same entity. It is closely related to the record linkage problem in the Databases community. However, instance matching brings new problems and requires a specific treatment.

Instances, also called individuals, are members of classes in an ontology. Sometimes the schema is not available, in which case instances can be thought in general as RDF resources. Since the RDF model is based on statements (triples) as the atomic structure for expressing knowledge, all the instances are described in a set of statements in which they appear as subject. An instance i is characterized by a URI (its unique identifier), and a set of statements $S_i = [\langle p_1, v_1 \rangle, \langle p_2, v_2 \rangle \ldots \langle p_n, v_n \rangle]$.

The instance matching problem can be defined as: given two instances i_s and i_t, belonging respectively to the ontologies S and T different ontologies, we want to learn a function $f(i_s, i_t) \rightarrow [0, 1]$ where 1 means that the two instances are referred to the same real-world object and 0 means they are two different entities.

While ontology matching refers to the problem of finding correspondences between ontological concepts, the instance matching problem determines whether two descriptions refer to the same real-world entity in a given domain. In other words, it consists in finding whether two URIs refer to the same real-world objects. An example may be finding in DBpedia a description of the same entity as `http://rdf.freebase.com/ns/en.barack_obama` (Freebase description of Barack Obama). DBpedia contains such entity and the associated URI is `http://dbpedia.org/resource/Barack_Obama`. The automatic discovery of such links allows for the integration of information from different data sources on the Web.

In ontology matching one of the most used approach is to compare every concept in the source with every concept in the target, building the so called similarity matrix. This approach requires $n * m$ comparisons, where n and m are the sizes of the source and target concept lists. Schemas are usually hundreds or thousands of concepts at most, and modern computers can handle the matching process. Instances are usually much more, as they can be thousands for every class. It is no longer possible to compare every source instance with every target instance, but some way of reducing the comparisons has to be introduced.

For instance, Freebase contains 20 million entities and DBPedia more than 14 million. In the case of DBpedia, the total number of triples (subject, predicate, object statements) exceeds 1 billion. Therefore, every instance in the source dataset cannot be compared with every instance in the target dataset. This is unlike traditional ontology matching, where an $n \times m$ similarity matrix is built, containing the results of comparing n concepts in the source ontology and m concepts in the target ontologies. That is, while schemas may contain hundreds or thousands of concepts at most, instances are usually much more numerous, therefore it is no longer feasible to compare every source instance with every target instance. Therefore it is crucial to devise a way to reduce the number of comparisons.

5.2 Record Linkage

The term *record linkage* has been introduced in the healthcare domain, when records about patients were merged together using names, addresses, birthdates, and other information. Since the 1960s, many researchers have focused on this problem, and many techniques have been developed, incorporating ideas from fields such as statistics, operations research, data mining, and machine learning. Record linkage has been surveyed in (48; 49; 50), and is also called *duplicate record detection* or *entity resolution* in the data integration field (51; 52).

The first ideas for record linkage were introduced by Howard Newcombe in (53), who introduced decision rules based on odds-ratios of frequencies. Newcombe understood that the frequencies of some string values in the database fields could be estimated among matches and non-matches, and this information should be used to compute a matching score. Also, the scores computed over different fields should be aggregated to obtain an overall score.

Newcombe's intuitions were then formalized in (54), where the first rigorous definition of record linkage was introduced. When matching two files A and B, the idea is to classify pairs in a product space $A \times B$ into M, the set of matches, and U, the set of non-matches. Fellegi and Sunter, following Newcombe's intuition considered ratios of probabilities of the form: $R = P(\gamma \in \Gamma|M)/P(\gamma \in \Gamma|U)$, where γ is an agreement pattern (e.g., sharing a particular string value in the 'name' field) in the comparison space Γ. The ratio R is known as the matching weight or score. The score is used to divide the space $A \times B$ into three disjoint sets using a decision rule:

- If $R > UPPER$, then the pair is a match.
- If $LOWER \leq R \leq UPPER$, then the pair is a possible match and needs to be reviewed by an expert.
- If $R > UPPER$, then the pair is a non-match.

where *LOWER* and *UPPER* are thresholds estimated using some known examples of matches and non-matches.

5.2.1 Record Linkage Techniques

Record linkage has been thoroughly treated in the past and many techniques and methodologies have been devised to address this problem.

Data Preparation. The first step in Record Linkage is *data preparation*, which significantly affects the quality of the overall matching process. An overview of data preparation techniques can be found in (55; 56). A very important preprocessing technique in record linkage is the *standardization* of strings such as names and addresses. It consists of replacing different spellings of words using a unified convention (e.g., the occurrences of 'Co', 'Co.', and 'Company' are unified into 'Co'). The standardization is performed using lookup tables, against which the words are compared and eventually substituted. This significantly improves the effectiveness of string matching algorithms.

String similarity metrics. A character-by-character comparison of strings is often unsatisfactory because of many reasons (i.e., typographical errors, slightly different naming conventions, different spellings). For these reasons, approximate string matching has been a major research topic in computer science. The research in this field led to the development of many string similarity metrics. These are functions that given two strings s_1 and s_2, return a score in the interval $[0, 1]$. A detailed overview of string metrics is provided in (57). One of the most successful string comparators is known as *edit distance*, which is the minimal cost of operations that have to be performed to one of the two objects in order to obtain the other. The Levenshtein edit distance (58) takes into account the number of insertions, deletions, and substitutions of characters required to transform one string into the other. There are many variations of these algorithms, and their effectiveness depends on the type of "errors" to be considered. Other examples are (59) and (60). Other approaches and token-based similarity metrics, which use cosine similarity and TF-IDF derived metrics (61). A survey on different techniques and measures that deal with this topic is (49).

Forcing 1-1 Matching. In a number of situations, the overall matching quality can be improved by making the assumption that a record in the source file can be matched with at most a record in the target file and viceversa. After the scores for every possible matching pair are computed, the problem can be formulated as a well-known optimization problem called the Assignment Problem, which consists of finding a maximum weight matching in a weighted bipartite graph.

Blocking. To address the problem of reducing the number of comparisons performed, in the record linkage literature a technique called blocking has been introduced. It consists of partitioning the datasets into disjoint subsets, and the actual comparisons are performed only between elements belonging to the same partition (62).

In record linkage there are many challenges, as deciding if records match is often computationally expensive and application specific (51). The former is because a combination of string similarity algorithms have to be used, the latter because it is difficult to provide a general solution which works well with heterogeneous datasets. For instance, the techniques used in matching scientific datasets will be different from the ones used for matching customers.

5.3 Differences between Instance Matching and Record Linkage

An instance in an ontology, is the analogous of a record in a database. For this reason, the problems of instance matching and record linkage (or duplicate detection) are closely related. Many techniques implemented in state-of-the-art instance matching tools are actually taken from the record linkage literature. However, there are some differences between records and instances, which generate the need for specific algorithms for instance matching.

First of all, the structure of relational databases and ontologies are quite different. The first are based on tables, while the second are based on graphs. Records belonging to the same table share the same structure with few possible variations. Instances instead can be very different. For example, it may happen that two instances of the same class in the same ontology have different properties defined on them, and it is difficult to choose which properties have to be used when matching with another ontology. In many record linkage applications, the fields to be compared are chosen manually, and then the research focus is on the value comparators and performance. In instance matching one of the main challenges is how to automatically select the properties whose values have to be compared.

A general characteristic of data modeling is that there are many ways of describing the same concepts or entities. Therefore, there are a number of structurally-different but semantically equivalent representations. This happens already in relational databases, where a typical example is the possibility of using a foreign key linking to another record or embedding the additional fields directly. The additional expressiveness provided by ontology languages such as OWL increases drastically the number of possible representations. For example, a concept such as red can be a subclass of the concept color or an instance of the same class. This is something that should be taken into account by instance matchers.

Another important difference between the ontologies and relational databases is that the former may contain implicit knowledge. For example, the instances of a class C are also instances of the classes that are superclasses of C. It has to be decided whether or not this fact should be considered in the matching process. In general, it is possible to run a reasoner before the matching process, so that all the implicit knowledge will be made explicit.

5.4 Evaluation

As in ontology matching, the growing interest in instance matching and the subsequent development of many matching tools have raised the need for standard benchmarks and evaluation methods. OAEI has been focusing on ontology matching for many years but starting from 2009 it started providing tracks for instance matching. The evaluation procedure and metrics are the same as the ones used for ontology matching, and they are incorporated in the ontology matching yearly evaluation event.

Since ontology matching has become a consolidated research area, we believe that the majority of the efforts will shift towards the instance level, and many more tracks and benchmarks will be addressed to the instance matching problem in the near future.

5.5 Instance Matching Techniques

In this section the instance matching techniques used by the state-of-the-art systems will be introduced. Because of the similarity between instance matching and record linkage previously explained, many techniques are taken from record linkage (e.g., string similarity metrics). The core of the matching process is the comparison between values of similar properties (attributes) using the so called *value-oriented* techniques. During this phase, a score is computed using string similarity functions, token-based similarity functions, conversion functions (e.g., transform real values into integers), statistical analysis (e.g., compute frequency of values and give more weight to the rare ones). Once all the scores are computed, these have to be combined in one single value representing the overall similarity between the two instances. This is done by a *decision system*, which takes as input the scores at a single value of granularity, and returns the final matching score. Decision systems range from simple linear combinations to complex machine learning techniques. A taxonomy of the techniques used in instance matching is shown in Figure 13.

Learning-based Techniques. Learning-based techniques consist in training a classifier to decide whether two instances refer to the same real-world entity or not. The classifier is usually

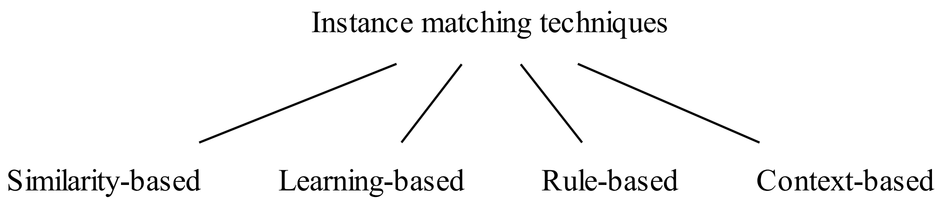


Figure 13. Classification of the instance matching approaches.

trained with some example instance pairs together with their actual classification. In this case, we are talking about *supervised-classification.* The quality of the classifier is highly influenced by the training set, which has to be carefully selected by a human. The training set has to be representative and balanced, meaning that it has to possess the same distribution as the overall data, and contain both positive and negative examples. These are strong requirements, and for this reason alternative learning methods have been proposed. An interesting example is *active learning*, in which the most ambiguous entities are presented to a domain expert who classifies them, and the system then learns from this feedback. Alternatively, when no training set is built, it is possible to use *unsupervised-learning* techniques. These methods exploit clustering techniques to group similar instances, then it is assumed that instances within the same cluster share the same class (matching or non-matching). The last approach used in the literature is the *semi-supervised learning* approach, which encompasses the combination of different learning techniques.

Similarity-based Techniques. When no training set is available, a similarity value is computed for each instance pair. Then, the final decision (match or non-match) is performed using a threshold: all the pairs whose similarity is over the threshold are designated as matches, and the others as non-matches. The scores provided in the value comparisons can be aggregated in different ways. The simplest method is to compute the average of the single scores, which means giving the same weight to every property/value. In many cases, though, some of the properties are more important than the others and this should be taken into account. Therefore, a more effective solution is to weigh every value based on some heuristics or input knowledge given manually by a domain expert.

An effective solution which does not require human intervention is the use of statistical information. The weights can be set based on frequencies. For example, a match between a very common last name such as 'Smith' should weigh less than a less common one. This would reduce the likelyhood of matching homonyms that refer to different real world entities. Similarity-based techniques have been extensively used both in the ontology matching and instance matching literature. The only drawback with such techniques is the identification of the correct threshold, which requires some human intervention.

Rule-based Techniques. Rule-based techniques make use of specific matching rules, which classify pairs as match or non-match based on the values scores previously computed. The idea is that even if the analogous of a primary key is not available, a set of uniquely identifying properties can be found and encoded in rules. These rules are usually determined by domain experts, therefore manual intervention is required. Rule-based approaches are usually very precise, but at the cost of being domain dependent.

Context-based Techniques. Context-based techniques compare not only the values contained in the pair of instances to be matched, but also the values included in related instances. This is extremely important in the context of Semantic Web and ontologies, because very often the value of a property points to another resource, which has other properties defined, in a recursive fashion.

5.6 Instance Matching Tools

A number of instance matching tools have been developed using the previously described techniques. An overview of these tools is reported in Table VII, which shows the techniques used by each system. Some of these tools were first developed as ontology matching systems, and then extended to match instances (e.g., AFlood, CODI, COMA++, DSSim, RiMOM), while others are specific for instance matching (e.g., SERIMI, LIMES, FBEM). Only a few of them provide a GUI to help users in setting up the parameters and analyzing the alignments produced (e.g., LIMES, COMA++).

As can be noted in Table II and similarly to the ontology matching state-of-the-art, most of the systems make use of similarity-based and context-based algorithms. String similarity metrics are used to compare the values of the properties, while context-based methods evaluate the structural similarity between instances. The values comparison can be significantly improved by using ad-hoc similarity functions (e.g., Zhishi.links), though they are domain-specific. To

TABLE VII

INSTANCE MATCHING TOOLS

	Techniques Used			
Tool	Similarity-based	Context-based	Rule-based	Learning-based
AFlood (15)	✓	✓		
ASMOV (17)	✓	✓		
CODI (19)	✓	✓		
COMA++ (20)	✓	✓		
DSSim (21)		✓		
FBEM (63)		✓	✓	
HMatch2.0 (64)	✓	✓		
LIMES (65)			✓	
ObjectCoref (66)				✓
RiMOM (24)	✓	✓		
SERIMI (67)	✓	✓		
Zhishi.links (68)	✓		✓	

evaluate the structural similarity, the properties and values have to be considered together by a similarity function. For this reason, tools such as FBEM and RiMOM aggregate the properties and values into a flat structure, which makes the comparisons easier and faster.

One of the toughest challenges in instance matching is how to decide which properties of the instances have to be compared. RiMOM uses schema matching and then manual refinement to determine the property alignment. Rule-based systems (e.g., LIMES and Silk) allow domain experts to specify linkage rules. This process requires a manual effort, but is the most precise and reliable. Another approach is to define a generic similarity function between RDF resources, when property mappings are not defined a priori. This was attempted by SERIMI, and is the current research direction in automatic instance matching. The only learning-based tool is ObjectCoref, which uses a semi-supervised learning approach to coreference instances. The use of both labeled and unlabeled data in the learning process allows the system to learn from a small set of labeled data.

CHAPTER 6

INSTANCE MATCHING FOR LOD

In this chapter the extension of AgreementMaker to match instances will be discussed, while the system was previously working only at the schema level. This extension has been developed and tested focusing particularly on Linked Open Data, and participated in the OAEI 2011 competition. In that occasion, the system was compared with other state-of-the-art matching tools with encouraging results, which will be shown at the end of the chapter.

6.1 Proposed Architecture

The proposed architecture for our instance matching system is shown in Figure 14. A three tier architecture has been developed, separating the key parts of the matching process into modules. The three phases are respectively Lookup, Disambiguation and Combination.

6.1.1 Lookup Phase

The Lookup phase consists in querying retrieval services to obtain some candidate instances. As introduced in Chapter 5, reducing the number of comparisons needed to provide an alignment is one of the main challenges in instance matching. The solution implemented in AgreementMaker consists in performing a look-up using the label of the instance and its type (when provided) to query against an index, which will return a reasonable number of candidate target instances. Many of the central LOD data sets offer a SPARQL endpoint, which is an online querying service that accepts queries in the SPARQL language and operates over the HTTP

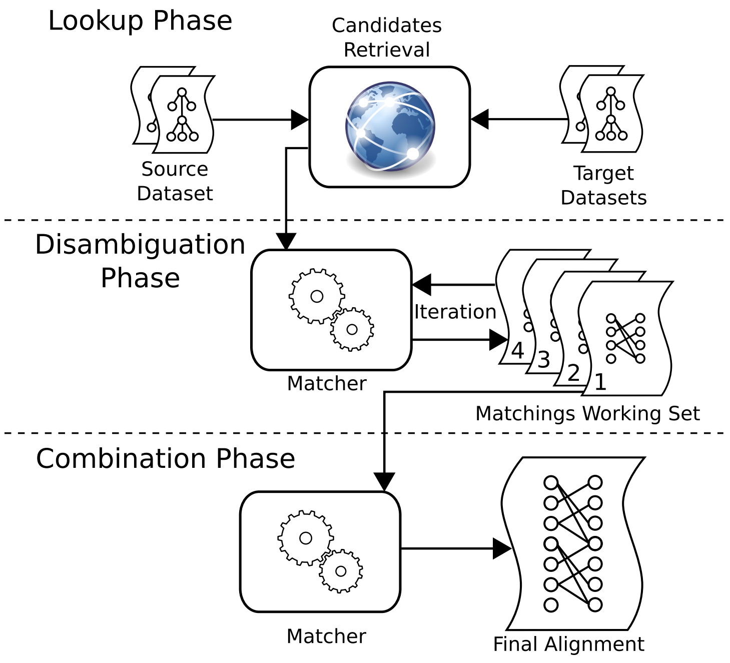


Figure 14. AgreementMaker OAEI2011 Instance Matching configuration.

protocol. In some other cases, the data can be accessed using an Application Programming Interface (API) available online. This choice is appropriate for several reasons:

- many SPARQL endpoints and APIs implement indexes, allowing for fast answers to keyword look-ups;
- the on-line version of these knowledge bases is always richer and more up to date than the versions that can be downloaded;
- multiple Knowledge Bases can be queried at the same time in a parallel fashion.

Then in the Disambiguation phase a similarity value between the source instance and the candidate instances is computed. This is achieved using different matchers that compare several features of the instances, and then their outputs are combined (Combination phase) in order to give a unique similarity value. These values are used to rank the candidates and eventually select the best one.

6.1.2 Disambiguation Phase

After the candidates have been retrieved in the lookup phase, in the disambiguation phase the system computes the similarities between the source instance and the candidate instances. In this step several different features may be taken into account and many different similarity measures may be exploited. The actual techniques implemented and used are described in detail in Section 6.2.

Following the AgreementMaker's extensibility and configurability principles, the matching techniques are separated into different classes, all extending the matcher module, implemented as an abstract Java class. In the disambiguation step all the instance matchers in AgreementMaker are run, producing similarity values for each pair of possible matching pair of instances.

6.1.3 Combination Phase

In this phase, the values returned by the matchers are used to rank the candidates and eventually select the best one. We use different matchers (forming the so called matchers stack) that compare several features about the instances to be matched, and then combine their outputs in order to give a final alignment. This process is needed because a unique decision (match/non-match) has to be taken for every pair of instances.

6.2 Matching Techniques

This section introduces the matching techniques that have been recently incorporated in AgreementMaker for matching instances. Some of them are readaptations of already available algorithms (e.g., string similarity), while others are specific for instance matching (e.g., property-value comparison). The main features we use for the comparisons are:

- Labels using a substring similarity.
- Comments and other literals using a Vector Space Model approach.
- RDF Statements considering property-value pairs.
- The score values returned by the lookup services (e.g. Freebase API, Apache Lucene score).

6.2.1 Label Instance Matcher

The first and most intuitive matcher implemented for AgreementMaker's instance matching module is the Label Instance Matcher (LIM). This matcher compares the labels of the instances, returning a score based on a string similarity metric. The *label* of an instance is a short string (e.g., from a few to several characters) representing the instance. The use of the term label has become a standard because there is a label property in RDFS, *rdfs:label*, which is widely used in ontologies and RDF data. The definition of this property is reported below.

The domain of this property (i.e., the class to which this property applies) is *Resource*, the most general class, since everything is a resource. The range (i.e., the datatype or class to which this property refers to) is *Literal*, the classes containing textual descriptions with no constraints. Even in case rdfs:label is not available, there always is an analogous property with

Listing 6.1. rdfs:label property definiton in RDFS

```
<rdf:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#label">
   <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
   <rdfs:label>label</rdfs:label>
   <rdfs:comment>A human−readable name for the subject.</rdfs:comment>
   <rdfs:domain rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

the same functionality. Examples are *foaf:name* and *skos:prefLabel*, respectively the defined in FOAF [1] and SKOS [2]. In some of the datasets the label can be an ad-hoc defined property, but the use of popular and agreed-upon properties is a preferred approach which is also encouraged by the W3C.

The matching process performed by the *LIM* is divided into three phases: Label Detection, Label Preprocessing, and Similarity Computation.

Label Detection

In this phase the matcher attempts to find the property used as label in the source and target instances. This is done using a lookup table in which the most common label properties are enumerated (e.g., rdfs:label, foaf:name, skos:prefLabel). If no such property is found, the matcher searches for a property with a name similar to the strings *label* or *name*.

[1]http://xmlns.com/foaf/spec/

[2]http://www.w3.org/2009/08/skos-reference/skos.html

Label Preprocessing

In case a label is found both for the source and target instance, the similarity between those two is computed. Before the actual similarity computation a sort of pre-processing is needed. In this phase we perform standardization and normalization of strings. The former, as in record linkage, consists of unifying different spellings of words under a unified convention (e.g., the occurrences of 'Jr', 'Jr.', and 'Junior' are unified into 'Jr.'), while the latter encompasses removal of punctuation and diacritics.

Similarity Computation

After the labels have been detected and preprocessed, the actual similarity value is computed and returned. This is done by using the string similarity metrics previously discussed. In AgreementMaker many string similarity metrics were already implemented and have been tested extensively in the field of ontology matching (e.g., Edit-distance, Jaro-Winkler). All of these metrics are exposed by *LIM* as parameters, and can be used in the matching process. A comparison between the results obtained using different string similarity metrics will be reported in the Results section.

6.2.2 Token-based Instance Matcher

The string similarity metrics work well on the labels and in general on short textual descriptions such as names, but longer text requires a different processing such as using frequencies of words. For this reason, we have implemented a matcher integrating Token-based techniques for comparing textual descriptions, called Token-based Instance Matcher (TIM).

The typical example of property that is successfully compared using token-based similarity metrics is *rdfs:comment*. Similarly to *rdfs:label*, it is a property that applies to every resource and its value is always a generic string. The comment usually contains a description of the instance longer than the one provided by a label, and since the overlapping between comments is in many cases a good similarity indicator between instances, it is used for matching. There are many other properties that can be used exploiting a token-based approach, such as the *dbpedia:abstract*, which is the abstract taken from Wikipedia, the types (sometimes there are many types associated with an instance), and the comment taken from other ontologies such as *skos:description*.

In the literature there are many Token-based similarity measures (69). There always is a sort of preprocessing in which the strings are turned into sets of tokens (also known as vectors). Stopword removal and stemming may also be performed. After this step, the sets are compared using a set similarity. The most common and used set similarities are reported in Table VIII. The similarity is maximized (i.e., equal to 1) only if the two sets share all the tokens, and is minimized (i.e., equal to 0) only if the two sets have no elements in common. Which function works better depends heavily on application and data characteristics (69).

The pseudo-code for the comparisons performed by the Token-based Instance Matcher is shown in Algorithm 1. At the beginning of the process, some relevant properties such as comments, abstracts, and types are detected in every instance and the values are aggregated in a single string called *virtual document*. These are then processed using tokenization and stopword removal. After that, the actual comparisons are performed. In this phase, every

TABLE VIII

SET SIMILARITY METRICS.

Metric	Function	Equation
Normalized Weighted Intersection	$N(s,t)$	$\frac{\|s \cap t\|}{max(\|s\|,\|t\|)}$
Jaccard Similarity	$J(s,t)$	$\frac{\|s \cap t\|}{\|s \cup t\|}$
Dice Similarity	$D(s,t)$	$\frac{2*\|s \cap t\|}{\|s\|+\|t\|}$
Cosine Similarity	$C(s,t)$	$\frac{\|s \cap t\|^2}{\|s\|*\|t\|}$

token in the source virtual document is tested against every token in the target. We designed three types of comparisons: string equality, synonymy check, and equality after stemming. The weights of the three matching conditions are parameters to the matching process, respectively called *equalityReward*, *synonymsReward*, and *stemmingReward*. Our experiments led us to set the values 1, 0.5, 0.5. All the matching scores are summed and in the end normalized using one of the set similarity measures in Table VIII. The one used by default by TIM is the Dice Similarity.

An even more accurate similarity comparison could be performed by weighing the vectors using frequencies of terms such as in the TF-IDF measure. This could slightly improve the accuracy of the TIM, but requires full access to the data to compute the statistics. The datasets we used are very large and using the endpoints makes them non-iterable. In fact, there is no way of computing those statistics unless the datasets are stored in memory or on disk. Alternatively, sampling could be performed, but this will be left as a future development.

Algorithm 1 Pseudo-code for Token Instance Matcher

```
1: function INSTANCESIMILARITY($source, candidate$)
2:     $S \leftarrow buildVirtualDocument(source)$            ▷ Returns a list of the property values
3:     $T \leftarrow buildVirtualDocument(target)$
4:     $S \leftarrow preProcess(source)$      ▷ Tokenization, stopword removal, and normalization
5:     $T \leftarrow preProcess(target)$
6:     $sim \leftarrow 0$
7:     for $s \in S$ do
8:         for $t \in T$ do
9:             if $s = t$ then
10:                $sim \leftarrow sim + equalityReward$
11:            else if $areSynonyms(s, t)$ then
12:                $sim \leftarrow sim + synonymsReward$
13:            else if $s.stem() = t.stem()$ then
14:                $sim \leftarrow sim + stemmingReward$
15:            end if
16:        end for
17:    end for
18:    $norm \leftarrow (S.length + T.length)/2$
19:    $sim \leftarrow sim/norm$
20:    return $sim$
21: end function
```

6.2.3 Property-Value Comparison

The first two matchers describe a comparison based on relations whose semantics is at least partially known a priori. As an example, the label is a property with the specific semantic of "a short textual description representing the instance". These matchers work because there is a shared way of using some properties, which is exactly one of the main goals of Linked Open Data and the Semantic Web. In a real-world situation, datasets can use any kind of property without following these principles, and our instance matching tool should provide relevant results also in this case. For this reason, there is a need for matching methods that work with *unknown* properties. Such methods are included in our matcher called Statements Instance Matcher (STIM), which provides techniques for comparing in general the statements belonging to two instances, namely the source and the target. The comparison is between the property-value pairs encoded in the statements.

Our implemented matcher first searches for *comparable* properties, which are the ones sharing the same URI or that possess similar names. As an example, many datasets contain geo-coordinates expressed using the standard *geo:long* and *geo:lat* properties, which can be easily detected and compared. Alternatively, matching properties can be discovered by our ontology matching algorithms or provided by a user. In the latter case, our approach would be similar to the rule-based approaches which involve a domain expert.

Once the properties to be compared are selected, the values are compared using string similarity metrics or mathemathical fuctions in case the values are real numbers. The property-value pairs to be compared are given a matching score which takes into account both the property similarity and the value similarity, as follows:

$$sim(\langle p_i, v_i\rangle, \langle p_j, v_j\rangle) = sim(p_i, p_j) \times sim(v_i, v_j) \tag{6.1}$$

The property-value scores are then normalized over the number of comparisons that have been performed so as to provide a unique overall score representing the similarity between the statements of the two instances.

6.2.4 Combination Methods

The scores computed by the previously described matchers have to be aggregated to provide a single score for each possible matching pair. This process is called *combination* and may be based on different heuristics. A first and simple method is to compute the average of the matchers, which means considering all the matchers at the same level. However, some matchers may be more effective than others, and should be given more importance when combining their results. For this reason, our combination module supports a linear weighted combination, where the weights are specified by a user. The user should understand which matchers may produce better results than the others and set the weights consequently. An improvement that was made to the linear weighted combination is the introduction of the possibility for a matcher not be included in the combination only in particular cases. It may happen that two instances have no comparable statements or one of them has no label to be compared with the other. In such cases, the matcher has to be excluded by the average or linear weighted combination. In the latter case, the weights have to be redistributed between the other matchers.

6.3 OAEI 2011 Participation

6.3.1 Track Description

The Data Interlinking track of the OAEI 2011 competition consists in recreating the links from the New York Times Data [1] to Freebase [2], DBPedia [3], and GeoNames [4]. These datasets involved are available on the Linked Open Data cloud, and are interlinked with other RDF datasets.

This track is particularly interesting and challenging, since it is a real-worlds application of instance matching and entails the following problems:

1. Datasets are very large and not easy to wholly retrieve and work with.
2. The source datasets (New York Times) have a very poor schema associated with them. Therefore, we cannot rely on traditional ontology matching to create schema level mappings.

Data Interlinking is composed by seven tasks. The source dataset is always the New York Times Data, while there are three different targets: Freebase, GeoNames, and DBPedia. Ta-

[1]http://data.nytimes.com/

[2]http://www.freebase.com/

[3]http://dbpedia.org/About

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

ble IX reports the sizes of the reference alignments, divided by matching task and entity type. Three types of entities are considered: People, Locations, and Organizations.

TABLE IX

STATISTICS ABOUT THE REFERENCE ALIGNMENT

Statistics	People	Organizations	Locations
Nr of NYT resources	9958	6088	3840
Total nr of sameAs links	14884	8003	87861
Links to Freebase	4979	3044	1920
Links to DBPedia	4977	1949	1920
Links to NYT	4979	3044	1920
Links to Geonames	0	0	1789

New York Times Data

The New York Times Data reflects the effort made by the popular newspaper New York Times [1] to semantically annotate a part of their huge collection of articles. The schema is very simple as it contains only three types of instances (i.e., People, Organizations and Locations), and there are no properties inter-relating them. Instances are provided with a label, a description page, possibly a comment, and a list of articles in which they are mentioned. In the article pages some keywords that can be used in the matching process are included.

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

DBPedia

DBpedia is the Semantic Web version of Wikipedia. It has become the center of the Linked Open Data cloud, the dataset to which most of the other datasets are linked to. Wikipedia is an excellent resource for multi-domain knowledge, since it contains a large number of heterogeneous entities.

Freebase

Freebase is a collaborative knowledge base whose data are introduced and mantained by a community of users. The dataset is totally multi-domain as it covers any kind of topic. It has a powerful search tool accessible to an API. In 2010, Freebase was acquired by Google, confirming the interest that the company has in Semantic Web technologies.

GeoNames

GeoNames is a geographical database accessible through an API. It contains a wide variety of places together with their geospatial coordinates. GeoNames has become a standard so that many other datasets use its conventions for encoding geo-coordinates.

6.3.2 Dataset Processing

As explained in 6.1, our choice for the Data Interlinking track was to use retrieval service in order to get candidate instances. The datasets involved can be queried online using SPARQL, when they provide an endpoint, or using APIs. Alternatively, dumps (i.e., the whole datasets in a downloadable format) may also be available. The two approaches may lead to different results depending on the datasets and the services exposed by who provides the data.

When dealing with large multi-domain knowledge bases like DBPedia and Freebase, memory is a bottleneck. These datasets occupy several gigabytes when they are compressed. In order to be queried, they need to be decompressed and also an index on disk is required to execute queries in reasonable times. For this reason, we decided to use the services available on the internet and to implement a caching mechanism to avoid the repetition of the same queries.

SPARQL endpoints are able to return RDF descriptions as answers, while JSON has become the standard for APIs. In the latter case, it is often possible to get the URIs from the JSON returned by the service and then access the whole descriptions with a URI lookup. Usually, APIs provide faster answers to queries, while SPARQL is slower but more flexible. What is missing in plain SPARQL is a fast approximated search. This is because it has been thought from the beginning as an exact query language, in a context where URIs are the unambiguous identifiers. In the web as it is today, it frequently happens to search for the same concept in different ways, for this a keyword lookup is needed. There are some projects integrating indexing and fast keyword query answering in SPARQL engines, and they are also used in some endpoints. An example is LARQ [1], an integration of Apache Lucene [2], an open source project implementing many information retrieval techniques, in a SPARQL engine.

[1]http://jena.sourceforge.net/ARQ/lucene-arq.html

[2]http://lucene.apache.org/java/docs/index.html

6.3.2.1 Typed and Untyped queries

We noticed that our queries can be divided in mainly two categories, namely typed and untyped. The former, as suggested by its name, asks for candidates belonging to a specific type, which is usually mapped to the type of the source instance. The latter, instead, relies on the label without asking for a specific type. The former leads to more precise candidates penalizing the recall, while the second improves the recall penalizing the precision.

Which one is better for a specific matching task depends on many factors. When the data sources share a type which is perfectly equivalent, a typed query performs better, because it would exclude the possibility of getting candidates that belong to a totally different type. In many real-world cases, though, it happens that some subsets of the classes are considered as different or even disjoint classes in one of the data sources. For example, a musical band may be considered as an organization in a data source and a distinct class in another data source. In such cases, recall can be highly penalized when using typed queries.

This situation represents the usual trade off between precision and recall. When there is more interest in precision it is better to use a typed query, while when high recall is preferred, an untyped query would be the best choice. For example, in the matching task New York Times/Freebase Organizations, a typed query would lead to an upperbound in recall (e.g., obtained by perfect disambiguating candidates) of 77.8%, while an untyped query would make an 88.3% recall result possible. In our experiments, we preferred to use untyped queries so as to avoid the recall drop-off, while the precision can still be improved with better disambiguation

Listing 6.2. Example of a Freebase query

```
http://www.freebase.com/api/service/search?
    query=barack+obama&
    type=/people/person&
    threshold=40
```

algorithms. Next, we will discuss how we accessed and processed the four data sources involved in the Data Interlinking track.

6.3.2.2 Freebase

Freebase provides an API which allows a keyword search. The query is passed through the HTTP protocol, using the parameters *query* (the actual keyword search terms), *type* (the type to be queried, in case the query is typed), and *threshold* (a parameter to limit the number of possible candidates). The results are returned in JSON, which can be easily parsed and converted to a list of instances in AgreementMaker. In Listing 6.2 is reported a Freebase typed query for the Person "Barack Obama".

6.3.2.3 DBPedia

DBPedia offers a sparql endpoint located at `http://dbpedia.org/sparql`. SPARQL endopoints are particularly interesting because many LOD datasets provide them. Listing 6.3 reported a typed query searching for the Italian town "Monza".

The keyword *construct* instructs the server to return the results as an RDF model made of statements as opposed to a classic tabular result. The subject is an instance of a particular type (`http://dbpedia.org/ontology/Place`), for which the label must contain a particular

Listing 6.3. Example of a DBpedia query

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
construct { ?p ?prop ?obj }
  WHERE {
      ?p rdf:type <http://dbpedia.org/ontology/Place> .
      ?p rdfs:label ?name .
      ?p ?prop ?obj .
      ?name bif:contains '"Monza"' .
      FILTER ( lang(?obj) = "" || lang(?obj) = "en")
  } LIMIT 1000
```

search term (in this case Monza). The object has either a non specified language or is written in English. We limit the number of statements to 1000 for performance and memory issues.

6.3.2.4 New York Times Data

The New York Times datasets are available on the website as RDF files separated by instance types. The information found in the downloadable datasets can be augmented by querying the API at the web address `http://data.nytimes.com/elements/search_api_query`. The RDF datasets contain already some queries which lead to data about the articles in which the queried entities are mentioned.

6.3.2.5 GeoNames

GeoNames provides an API which allows a keyword search. The query is passed through the HTTP protocol, using the parameters *q* (the actual keyword search terms), *type* (the format in which the results have to be returned), and *maxRows* (a parameter to limit the number of possible candidates). In this case, the results are returned in RDF, which can be easily loaded

Listing 6.4. Example of a GeoNames query

```
http://api.geonames.org/search?
    q=Monza
    type=rdf&
    maxRows=10
```

into a list of instances in AgreementMaker. In Listing 6.4 is reported a GeoNames typed query for the location “Monza”.

6.4 Results

In this section we will explain the experiments that have been performed to test the AgreementMaker's instance matching module against the datasets involved in the OAEI 2011. All the tables and figures in this section will include the precision, recall, and F-measure in several matching tasks.

6.4.1 Evaluation of String Similarity Metrics

We compared the effectiveness of the Label Instance Matcher using several string similarity metrics, and we report the most significant results in Table X. The metrics compared are Jaro-Winkler, Edit-Distance, and Q-Grams, and AM-Substring. While the first three are popular algorithms in the literature, AM-Substring is a version of the substring similarity algorithm that had been already introduced in AgreementMaker in the context of ontology matching.

The experiments show that all the four tested metrics perform similarly. The best one is Jaro-Winkler that performs slightly better than the others, and for this reason will be used in the remaining experiments. The average difference between the best metric (Jaro-Winkler) and the

TABLE X

RESULTS ACHIEVED USING LABEL INSTANCE MATCHER USING DIFFERENT STRING SIMILARITY ALGORITHMS

	Jaro-Winkler			**Edit-Distance**			**AM-Substring**			**Q-Grams**		
Matching Task	Prec	Rec	F-m	Prec	Rec	F-m	Prec	Rec	F-m	Prec	Rec	F-m
NYT-DBpedia-Loc	0.882	0.642	**0.743**	0.745	0.667	0.704	0.742	0.658	0.697	0.748	0.668	0.706
NYT-DBpedia-Org	0.801	0.721	0.759	0.769	0.790	0.780	0.787	0.797	**0.792**	0.784	0.784	0.784
NYT-DBpedia-Peo	0.964	0.912	0.937	0.960	0.933	**0.946**	0.959	0.931	0.945	0.957	0.925	0.941
NYT-Freebase-Loc	0.871	0.842	**0.856**	0.863	0.840	0.851	0.865	0.841	0.853	0.865	0.841	0.853
NYT-Freebase-Org	0.885	0.854	**0.869**	0.854	0.827	0.840	0.867	0.839	0.853	0.869	0.843	0.856
NYT-Freebase-Peo	0.950	0.939	**0.944**	0.946	0.937	0.942	0.946	0.936	0.941	0.947	0.938	0.942
NYT-GeoNames	0.803	0.409	0.542	0.783	0.417	0.544	0.780	0.415	0.542	0.784	0.417	**0.545**
Average	0.879	0.760	**0.807**	0.846	0.773	0.801	0.849	0.774	0.803	0.851	0.774	0.804

worst (Edit-Distance) is less than one percent. This means that in this context all the popular string similarity metrics are able to capture similar differences in spelling or typographical errors, and the key to improve the results is not on the Label Instance Matcher but in other methods such as the ones implemented in the other matchers.

6.4.2 Analysis of Matchers Effectiveness

Figure 15 reports the evaluation for each matcher separately. This allows us to see the impact of our techniques on the overall results. Our tests have a baseline, which consists in creating an alignment only when the list of candidates is composed by only one instance. It is a very simple approach that in this case leads to a very high precision, with low recall. This is because most of the instances that appear in the source datasets are present also in the target datasets. The recall instead is low, due to the problem of ambiguity. In fact, it is not so frequent to find only one instance in the candidates list.

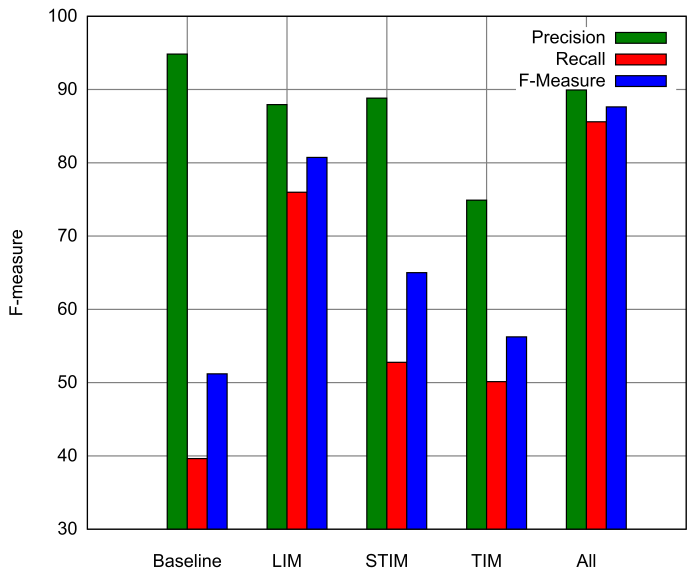


Figure 15. Analysis of the effectiveness of each matcher.

The Label Instance Matcher (LIM) improves significantly the recall, while the precision is slightly worse. The F-measure it provides is 30% higher than the baseline, which is a remarkable improvement. The Statements Instance Matcher (STIM) is very precise but finds fewer mappings than LIM. The Token-based Instance Matcher is the worst performing matcher of the three, but it still gives a 6% improvement with respect to the baseline.

The combination of the three matchers (ALL) is the overall best. As it happens in ontology matching, a combinational approach is able to get the best of different matchers, providing the best results. The overall improvement with respect to the baseline is 36%, which we consider an excellent achievement.

6.5 OAEI 2011 Results

This section reports the official results achieved by the systems competing in the Data Interlinking track of the OAEI 2011 challenge. AgreementMaker has participated in the OAEI challenge starting from 2006, where it has always been one of the best ontology matching tools. In the 2011 edition of the challenge, we entered for the first time the instance matching track.

Although many instance matching tools have been presented in the past years (e.g., see Section 5.6) and competed in the previous editions of the OAEI, in the 2011 edition only three tools presented their alignments for the instance matching track. This is in our opinion because despite the efforts in building generic tools, instance matching still requires some time-consuming preprocessing of the input data.

In Table XI the results achieved by all the systems in each of the Data Interlinking tasks are summarized. The results of AgreementMaker have been subject to further improvement after the competition, since its first version was developed in a short time. The most recent results are reported in Table XII, showing a comparison with the version participating in the competition.

All of the three tools are able to provide very good alignments, as the average over 80%. This is a very good result, showing that the interlinking problem in Linked Open Data can be

TABLE XI

RESULTS ACHIEVED BY THE SYSTEMS PARTICIPATING IN THE DATA INTERLINKING TRACK

	AgreementMaker			SERIMI			Zhishi.links		
Matching Task	Prec	Rec	F-m	Prec	Rec	F-m	Prec	Rec	F-m
NYT-DBpedia-Loc	0.79	0.61	0.69	0.69	0.67	0.68	0.92	0.91	**0.92**
NYT-DBpedia-Org	0.84	0.67	0.74	0.89	0.87	0.88	0.9	0.93	**0.91**
NYT-DBpedia-Peo	0.98	0.8	0.88	0.94	0.94	0.94	0.97	0.97	**0.97**
NYT-Freebase-Loc	0.88	0.81	0.85	0.92	0.9	**0.91**	0.9	0.86	0.88
NYT-Freebase-Org	0.87	0.74	0.8	0.92	0.89	**0.91**	0.89	0.85	0.87
NYT-Freebase-Peo	0.97	0.95	**0.96**	0.93	0.91	0.92	0.93	0.92	0.93
NYT-GeoNames	0.9	0.8	0.85	0.79	0.81	0.8	0.94	0.88	**0.91**
Average	0.890	0.769	0.824	0.869	0.856	0.863	**0.921**	**0.903**	**0.913**

in many cases solved in an automatic fashion. We consider this very relevant to the research in the field, given the high growth rate of LOD and the consequent need of matching tools.

Zhishi.links has the best results, obtained by encoding specific rules and dictionaries to solve the matching task. It uses direct access to the datasets as opposed to AgreementMaker and SERIMI, which queried the retrieval services. Zhishi.links focused on how to manage the matching process involving large datasets, and implemented a distributed algorithm which was run on a cluster of machines using MapReduce (70). AgreementMaker and SERIMI instead preferred to implement more general techniques such as property comparison instead of encoding specific linkage rules. Moreover, Zhishi.links was developed specifically for the competition, while SERIMI was used to match other datasets, and AgreementMaker has been used for several years in ontology matching.

All the systems perform best in the tasks involving entities of type Person. This is because it is the category of entities for which the ambiguity is minimal. Furthermore, the heterogeneity of the naming conventions are limited. It is different in Organizations and even more in Locations, where the ambiguity problem is substantial. For instance, there are many cities in different states sharing the same name and also many other types of entities can have similar ones.

Both AgreementMaker and SERIMI perform better in Freebase tasks than in DBPedia, because the lookup service of the former returns fewer and more precise candidates. Therefore, the disambiguation task is easier when working with Freebase data. In fact, DBpedia keyword search does not allow mistakes and spelling differences, leading to a loss in recall, while Freebase search is more flexible. All the systems provide good results in the GeoNames test, because there are some shared properties between the datasets which help in the matching process.

TABLE XII

RESULTS OBTAINED BY AgreementMaker IN THE DATA INTERLINKING TRACK OF THE OAEI 2011 CHALLENGE.

	AgreementMaker			**AgreementMaker (Last)**		
Matching Task	Precision	Recall	F-Measure	Precision	Recall	F-Measure
NYT-DBpedia-Loc	0.790	0.612	0.690	0.909	0.739	**0.815**
NYT-DBpedia-Org	0.840	0.667	0.744	0.846	0.845	**0.846**
NYT-DBpedia-Peo	0.977	0.801	0.881	0.962	0.934	**0.948**
NYT-Freebase-Loc	0.884	0.811	0.846	0.874	0.846	**0.860**
NYT-Freebase-Org	0.873	0.735	0.798	0.917	0.897	**0.907**
NYT-Freebase-Peo	0.966	0.950	**0.958**	0.948	0.940	0.944
NYT-GeoNames-Loc	0.902	0.797	**0.846**	0.839	0.792	0.815
Average	0.890	0.769	0.824	**0.899**	**0.856**	**0.876**

Further improvement has been made after the competition, especially in the infrastructure and in the access to the retrieval services. The system possesses now all the capabilities described in the previous sections. Our developments led to an improvement in all of the tracks except for two, where the results are slightly worse than before, because with more candidates it increases also the possibility of finding mismatches. Our alignments are now better on average than SERIMI, while still a bit lower than Zhishi.links.

CHAPTER 7

CONCLUSIONS

We have extended AgreementMaker, one of the state-of-the-art ontology matching systems in the Semantic Web literature, following these directions:

1. Design and implementation of new ontology matching algorithms.
2. Design of a novel infrastructure for instance matching.
3. Design and implementation of instance matching algorithms.

We have designed and tested new ontology matching algorithms for discovering subclass relations, which are particularly useful when matching LOD ontologies. In particular, we have introduced the concept of Global Matching (GM), which uses subclass axioms present in several LOD ontologies to infer mappings between the two ontologies to be matched. This technique is particularly interesting because it is able to capture the patterns in the usage of shared concepts between the LOD ontologies. These patterns reflect the idea of reusability, one of the key concepts in knowledge engineering and ontologies. Then, we implemented and a novel probabilistic algorithm (DPLC) for discovering links using a mediator ontology such as WordNet. The strength of this algorithm is that it first applies word sense disambiguation techniques to filter the irrelevant concepts and then it takes into account the distance between concepts in the mediator ontology.

We have evaluated our novel approaches using the standard metrics used in the field against a set of reference alignments that have been used by the best state-of-the-art systems in matching LOD ontologies. We showed that a combinational approach which aggregates heterogeneous matchers works very well also when the mappings analyzed are mostly of type subclass. Our results are the overall best, especially in precision, while the use of multi-domain background knowledge such as Wikipedia still leads to a better recall.

The overall results show that mediator-based approaches are very promising, and therefore should be explored more. In particular, the adoption of external lexical resources such as WordNet and Wikipedia is crucial in the matching process. The use of such resources is the reason why AgreementMaker and BLOOMS achieve better results than the other tools. Future research will include experimenting other knowledge bases such as Wikipedia, DBpedia or Freebase to be used in our system as mediator ontology.

We have extended AgreementMaker with a novel instance matching infrastructure, as well as with several matching algorithms. In particular, we have defined a three-layer architecture for instance matching composed by a lookup phase, a disambiguation phase where many matchers compare the source instance with a set of candidates, and a combination phase in which the scores provided by different matchers are unified into a single output score. This architecture allows us to drastically reduce the number of comparisons made to match two datasets, which would otherwise make the matching process too computationally expensive to be performed in reasonable times by a modern computer.

In the lookup phase, we have shown how to access the data provided by different services and endpoints using the label of the instance to be matched, and how minor modifications on the queries may change significantly the quality of the candidate instances returned (e.g., typed and untyped queries). Future work would be investigate more *query expansion* techniques, which would improve the overall quality of the candidates and make the disambiguation task easier.

For the disambiguation part, we implemented three matchers: LIM, TIM, and STIM. The first two are based on the semantics of some particular relations (e.g., labels and comments), while the third one works with any type of properties without a priori knowledge. In the combination phase, we put together these methods using a linear weighted combination, with the addition of the possibility to redistribute the weights in case any of the matchers does not find comparable properties. We have evaluated our infrastructure using some central LOD datasets and proved the effectiveness of our algorithms. Our systems provides competitive results when compared with other state-of-the-art systems. As future research, we will investigate the extension of our methods to take into account statistics of the datasets involved, using sampling methods in case of non-iterable datasets. Since our infrastructure is highly extensible, other matchers will be integrated in the process as well. We will also explore new combination techniques which will adaptively set the weight in the combination based on some heuristics such as instrinsic quality measures.

CITED LITERATURE

1. Cruz, I. F., Palandri Antonelli, F., and Stroe, C.: AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. PVLDB, 2(2):1586–1589, 2009.

2. Cruz, I. F., Palmonari, M., Caimi, F., and Stroe, C.: Towards "On the Go" Matching of Linked Open Data Ontologies. In IJCAI Workshop Discovering Meaning On the Go in Large & Heterogeneous Data (LHD), pages 37–42, 2011.

3. Cruz, I. F., Stroe, C., Caimi, F., Fabiani, A., Pesquita, C., Couto, F. M., and Palmonari, M.: Using AgreementMaker to Align Ontologies for OAEI 2011. In ISWC International Workshop on Ontology Matching (OM), volume 814 of CEUR Workshop Proceedings, pages 114–121, 2011.

4. Berners-Lee, T.: Linked Data-The Story So Far. International Journal on Semantic Web and Information Systems, 5(3):1–22, 2009.

5. Heath, T. and Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology, 1(1):1–136, 2011.

6. Antoniou, G. and van Harmelen, F.: A Semantic Web Primer. MIT Press, 2004.

7. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S.: DBpedia - A Crystallization Point for the Web of Data. Journal of Web Semantics, 7(3):154–165, 2009.

8. Cyganiak, R. and Jentzsch, A.: Linking Open Data Cloud Diagram. http://lod-cloud.net/.

9. Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199–220, June 1993.

10. Euzenat, J. and Shvaiko, P.: Ontology Matching. Heidelberg (DE), Springer-Verlag, 2007.

11. Kalfoglou, Y. and Schorlemmer, M.: Ontology Mapping: the State of the Art. The Knowledge Engineering Review, 18(1):1–31, 2003.

12. Shvaiko, P. and Euzenat, J.: A Survey of Schema-Based Matching Approaches. In Journal on Data Semantics IV, volume 3730 of Lecture Notes in Computer Science, pages 146–171. Springer, 2005.

13. Noy, N. F.: Semantic Integration: A Survey Of Ontology-Based Approaches. SIGMOD Record, 33(4):65–70, 2004.

14. Castano, S., Ferrara, A., Montanelli, S., and Varese, G.: Ontology and Instance Matching. In Knowledge-Driven Multimedia Information Extraction and Ontology Evolution, pages 167–195, 2011.

15. Hanif, M. S. and Aono, M.: Anchor-Flood: Results for OAEI 2009. In ISWC International Workshop on Ontology Matching (OM), 2009.

16. David, J., Guillet, F., and Briand, H.: Matching Directories and OWL Ontologies with AROMA. In International Conference on Information and Knowledge Management (CIKM), pages 830–831, 2006.

17. Jean-Mary, Y. R., Shironoshita, E. P., and Kabuka, M. R.: ASMOV: Results for OAEI 2010. In ISWC International Workshop on Ontology Matching (OM), 2010.

18. Jain, P., Hitzler, P., Sheth, A., Verma, K., and Yeh, P. Z.: 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.

19. Noessner, J. and Niepert, M.: Codi: Combinatorial optimization for data integration: results for oaei 2010. In ISWC International Workshop on Ontology Matching (OM), 2010.

20. Aumueller, D., Do, H. H., Massmann, S., and Rahm, E.: Schema and Ontology Matching with COMA++. In SIGMOD Conference, pages 906–908, 2005.

21. Nagy, M., Vargas-Vera, M., and Stolarski, P.: DSSim Results for OAEI 2009. In ISWC International Workshop on Ontology Matching (OM), 2009.

22. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., and Halevy, A. Y.: Learning to Match Ontologies on the Semantic Web. VLDB Journal, 12(4):303–319, 2003.

23. Jiménez-Ruiz, E. and Grau, B. C.: LogMap: Logic-Based and Scalable Ontology Matching. In International Semantic Web Conference (ISWC), pages 273–288, 2011.

24. Li, J., Tang, J., Li, Y., and Luo, Q.: RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. IEEE Trans. Knowl. Data Eng., 21(8):1218–1232, 2009.

25. Lambrix, P. and Tan, H.: SAMBO - A System for Aligning and Merging Biomedical Ontologies. J. Web Sem., 4(3):196–206, 2006.

26. Giunchiglia, F., Shvaiko, P., and Yatskevich, M.: S-Match: an Algorithm and an Implementation of Semantic Matching. In The Semantic Web: Research and Applications, First European Semantic Web Symposium, pages 61–75, 2004.

27. Cruz, I., Palmonari, M., Caimi, F., and Stroe, C.: Building Linked Ontologies with High Precision Using Subclass Mapping Discovery. In Artificial Intelligence Review, 2012. Manuscript submitted for publication.

28. Bizer, C., Heath, T., and Berners-Lee, T.: Linked Data—The Story So Far. International Journal on Semantic Web and Information Systems (IJSWIS), 5(3):1–22, 2009.

29. eds. S. Staab and R. Studer Handbook on Ontologies. International Handbooks on Information Systems. Springer, 2004.

30. Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C.: DBpedia Spotlight: Shedding Light on the Web of Documents. In International Conference on Semantic Systems (I-Semantics), 2011.

31. Euzenat, J., Ferrara, A., Hollink, L., Isaac, A., Joslyn, C., Malaisé, V., Meilicke, C., Nikolov, A., Pane, J., Sabou, M., Scharffe, F., Shvaiko, P., Spiliopoulos, V., Stuckenschmidt, H., Sváb-Zamazal, O., Svátek, V., dos Santos, C. T., Vouros, G. A., and Wang, S.: Results of the Ontology Alignment Evaluation Initiative 2009. In ISWC International Workshop on Ontology Matching (OM), volume 551 of CEUR Workshop Proceedings, pages 73–126, 2009.

32. Euzenat, J., Ferrara, A., Meilicke, C., Pane, J., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., Svátek, V., and dos Santos, C. T.: Results of the Ontology Alignment Evaluation Initiative 2010. In ISWC International Workshop on Ontology Matching (OM), volume 689 of CEUR Workshop Proceedings, pages 85–117, 2010.

33. Euzenat, J., Ferrara, A., van Hage, W. R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., and dos Santos, C. T.: Results of the Ontology Alignment Evaluation Initiative 2011. In ISWC International Workshop on Ontology Matching (OM), volume 814 of CEUR Workshop Proceedings, pages 85–113, 2011.

34. Cruz, I. F., Palandri Antonelli, F., Stroe, C., Keles, U., and Maduko, A.: Using AgreementMaker to Align Ontologies for OAEI 2009: Overview, Results, and Outlook. In ISWC International Workshop on Ontology Matching (OM), volume 551 of CEUR Workshop Proceedings, 2009.

35. Cruz, I. F., Stroe, C., Caci, M., Caimi, F., Palmonari, M., Palandri Antonelli, F., and Keles, U. C.: Using AgreementMaker to Align Ontologies for OAEI 2010. In ISWC International Workshop on Ontology Matching (OM), volume 689 of CEUR Workshop Proceedings, pages 118–125, 2010.

36. Volz, J., Bizer, C., Gaedke, M., and Kobilarov, G.: 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.

37. Nikolov, A., Uren, V., Motta, E., and Roeck, A.: 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.

38. Parundekar, R., Knoblock, C., and Ambite, J. L.: 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.

39. Li, J., Tang, J., Li, Y., and Luo, Q.: RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. IEEE Transactions on Data and Knowledge Engineering, 21(8):1218–1232, 2009.

40. Giunchiglia, F., Yatskevich, M., and Shvaiko, P.: Semantic Matching: Algorithms and Implementation. In Journal on Data Semantics IX, volume 4601 of Lecture Notes in Computer Science, pages 1–38. Springer, 2007.

41. Damova, M., Kiryakov, A., Simov, K. I., and Petrov, S.: Mapping the Central LOD Ontologies to PROTON Upper-level Ontology. In ISWC International Workshop on Ontology Matching (OM), volume 689 of CEUR Workshop Proceedings, pages 61–72, 2010.

42. Sabou, M., d'Aquin, M., and Motta, E.: 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.

43. Sorrentino, S., Bergamaschi, S., Gawinecki, M., and Po, L.: Schema Label Normalization for Improving Schema Matching. Data & Knowledge Engineering, 69(12):1254–1273, 2010.

44. Po, L. and Sorrentino, S.: Automatic Generation of Probabilistic Relationships for Improving Schema Matching. Information Systems, 36(2):192–208, 2011.

45. Plag, I.: Word-formation in English. Cambridge University Press, 2003.

46. Williams, E.: On the Notions "Lexically Related" and "Head of a Word". Linguistic Inquiry, 12(2):245–274, 1981.

47. Cruz, I. F., Palandri Antonelli, F., and Stroe, C.: 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, pages 49–60, 2009.

48. Newcombe, H. B.: Handbook of Record Linkage: Methods for Health and Statistical Studies, Administration, and Business. Oxford University Press, Inc., 1988.

49. Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S.: Duplicate Record Detection: A Survey. 19(1):1–16, 2007.

50. Winkler, W.: Overview of Record Linkage and Current Research Directions. Current, (2006-2), 2006.

51. Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S., and Widom, J.: Swoosh: A Generic Approach to Entity Resolution. The VLDB Journal, 18(1):255–276, 2009.

52. Bhattacharya, I. and Getoor, L.: Collective Entity Resolution in Relational Data. ACM Transactions on Knowledge Discovery from Data (TKDD), 1(1):5, 2007.

53. Newcombe, H., Kennedy, J., Axford, S., and James, A.: Automatic Linkage of Vital Records. 1959.

54. Fellegi, I. and Sunter, A.: A Theory for Record Linkage. Journal of the American Statistical Association, pages 1183–1210, 1969.

55. Kimball, R. and Caserta, J.: The Data Warehouse Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. John Wiley & Sons, 2004.

56. Zhang, S., Zhang, C., and Yang, Q.: Data Preparation for Data Mining. Applied Artificial Intelligence, 17(5–6):375–381, 2003.

57. Navarro, G.: A Guided Tour to Approximate String Matching. ACM Computing Surveys (CSUR), 33(1):31–88, 2001.

58. Ristad, E. S. and Yianilos, P. N.: Learning String-Edit Distance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(5):522–532, 1998.

59. Jaro, M. A.: UNIMATCH: A Record Linkage System: User's Manual. Technical report, US Bureau of the Census, Washington, DC, 1976.

60. Ullmann, J. R.: A Binary N-gram Technique for Automatic Correction of Substitution, Deletion, Insertion and Reversal Errors in Words. The Computer Journal, 20(2):141–147, 1977.

61. Cohen, W. W.: Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. In ACM SIGMOD International Conference on Management of Data, pages 201–212, 1998.

62. Draisbach, U. and Naumann, F.: A Generalization of Blocking and Windowing Algorithms for Duplicate Detection. In International Conference on Data and Knowledge Engineering (ICDKE), pages 18–24, 2011.

63. Stoermer, H. and Rassadko, N.: Results of OKKAM Feature based Entity Matching Algorithm for Instance Matching Contest of OAEI 2009. In ISWC International Workshop on Ontology Matching (OM), 2009.

64. Castano, S., Ferrara, A., Lorusso, D., and Montanelli, S.: The HMatch 2.0 Suite for Ontology Matchmaking. In SWAP, eds. G. Semeraro, E. D. Sciascio, C. Morbidoni, and H. Stoermer, volume 314 of CEUR Workshop Proceedings. CEUR-WS.org, 2007.

65. Ngomo, A.-C. N. and Auer, S.: LIMES A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In IJCAI, ed. T. Walsh, pages 2312–2317. IJCAI/AAAI, 2011.

66. Hu, W., Chen, J., Cheng, G., and Qu, Y.: ObjectCoref & Falcon-AO: results for OAEI 2010. In ISWC International Workshop on Ontology Matching (OM), 2010.

67. Araújo, S., Hidders, J., Schwabe, D., and de Vries, A. P.: SERIMI-Resource Description Similarity, RDF Instance Matching and Interlinking. CoRR, abs/1107.1104, 2011.

68. Niu, X., Rong, S., Zhang, Y., and Wang, H.: Zhishi.links Results for OAEI 2011. In ISWC International Workshop on Ontology Matching (OM), volume 814 of CEUR Workshop Proceedings, pages 220–227, 2011.

69. Hadjieleftheriou, M. and Srivastava, D.: Weighted Set-Based String Similarity. IEEE Data Engineering Bulletin, 33(1):25–36, 2010.

70. Dean, J. and Ghemawat, S.: MapReduce: a Flexible Data Processing Tool. Communications of the ACM, 53(1):72–77, 2010.

VITA

Education

- B.S. Computer Engineering, Politecnico di Milano, Milan, Italy, 2009
- M.S. Computer Engineering, Politecnico di Milano, Milan, Italy, 2012

Publications

- Isabel F. Cruz, Matteo Palmonari, Federico Caimi, Cosmin Stroe: Towards ”On the Go” Matching of Linked Open Data Ontologies. LDH 2011: 37-42
- Isabel F. Cruz, Cosmin Stroe, Federico Caimi, Alessio Fabiani, Catia Pesquita, Francisco M. Couto, Matteo Palmonari: Using AgreementMaker to align ontologies for OAEI 2011. In ISWC International Workshop on Ontology Matching (OM), volume 814 of CEUR Workshop Proceedings, 2011.
- Isabel F. Cruz, Cosmin Stroe, Michele Caci, Federico Caimi, Matteo Palmonari, Flavio Palandri Antonelli, 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.