DEGREE OF DOCTOR OF PHILOSOPHY IN
COMPUTER ENGINEERING AND SCIENCE

DOCTORATE SCHOOL IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXVIII Cycle

UNIVERSITY OF MODENA AND REGGIO EMILIA

INFORMATION ENGINEERING DEPARTMENT

Ph.D. DISSERTATION

Revealing the underlying structure of Linked Open Data for enabling visual querying

Candidate:
Fabio BENEDETTI
Advisor:
Prof. Sonia BERGAMASCHI
Co-Advisor:
Prof. Laura PO
The Director of the School:
Prof. Giorgio M. VITETTA

DOTTORATO DI RICERCA IN
COMPUTER ENGINEERING AND SCIENCE

SCUOLA DI DOTTORATO IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXVIII Ciclo

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI PER IL CONSEGUIMENTO DEL TITOLO DI DOTTORE DI RICERCA

Rivelare la struttura dei Linked Open Data per permettere interrogazioni visuali

Candidate:
Fabio BENEDETTI
Advisor:
Prof. Sonia BERGAMASCHI
Co-Advisor:
Prof. Laura PO
The Director of the School:
Prof. Giorgio M. VITETTA

Keywords:

Schema Extraction
Schema Summarization
SPARQL Query Generation
Classes Clustering
Document Similarity

Abstract

The Linked Data Principles ratified by Tim-Berners Lee promise that a large portion of Web Data will be usable as one big interlinked RDF (i.e. *Resource Description Framework*) database. Today, with more than one thousand of *Linked Open Data* (LOD) sources available on the Web, we are assisting to an emerging trend in publication and consumption of LOD datasets. However, the pervasive use of external resources together with a deficiency in the definition of the internal structure of a dataset causes that many LOD sources are extremely complex to understand.

The goal of this thesis is to propose tools and techniques able to reveal the underlying structure of a generic LOD dataset for promoting the consumption of this new format of data. In particular, I propose an approach for the automatic extraction of statistical and structural information from a LOD source and the creation of a set of indexes (i.e. *Statistical Indexes*) that enhance the description of the dataset. By using this structural information, I defined two models able to effectively describe the structure of a generic RDF dataset: *Schema Summary* and *Clustered Schema Summary*. The Schema Summary contains all the main classes and properties used within the datasets, whether they are taken from external vocabularies or not. The Clustered Schema Summary, suitable for large LOD datasets, provides a more high-level view of the classes and the properties used by gathering together classes that are object of multiple instantiations. All these efforts allowed the development of a tool called *LODeX* able to provide a high-level summarization of a LOD dataset and a powerful visual query interface to support users in querying/analyzing an unknown datasets.

All the techniques proposed in this thesis have been extensively evaluated and compared with the state of the art in their field: a performance evaluation of the LODeX's module delegated to the extraction of the indexes is proposed; the technique of schema summarization has been evaluated according to ontology summarization metrics; finally, LODeX itself has been evaluated inspecting its portability and usability.

In the second part of the thesis, I present a novel technique called CSA (*Context Semantic Analysis*) that exploits the information contained in a knowledge

graph for estimating the similarity between documents. This technique has been compared with other state of the art measures by using a benchmark containing documents an measures of similarity provided by human judges.

Sommario

Tim-Berners Lee, quando ha definito i Lineked Data Principles, aveva predetto che il Web dei dati sarebbe stato utilizzabile come un grande database RDF (*Resource Description Framework*). Oggigiorno, con piú di un migliaio di sorgenti *Linked Open Data* (LOD) disponibili online, stiamo assistendo ad un incremento nel volume delle sorgenti pubblicate. Tuttavia, il continuo utilizzo di risorse esterne e la comune mancanza di una definizione formale della struttura del dataset fa sí che molte sorgenti LOD siano molto complesse da interpretare ed utilizzare.

Lo scopo di questa tesi e quello di proporre tecniche e strumenti capaci di rivelare la struttura di ogni sorgente LOD, al fine di promuovere lutilizzo di questa nuova tipologia di dati. In particolare viene proposto un approccio per estrarre automaticamente informazioni strutturali e statistiche da una sorgente LOD andando a popolare un set di indici, chiamati *Statistical Indexes*. In seguito saranno proposti due modelli capaci di descrivere in modo sintetico ed efficace la struttura di un generico dataset RDF: *Schema Summary* e *Clustered Schema Summary*. Lo Schema Summary contiene le classi e le proprietá presenti nel dataset, sia che siano o meno formalmente definite al interno della sorgente. Il Clustered Schema Summary, indicato per descrivere grandi dataset, fornisce una visione di piú alto livello della struttura della sorgente clusterizzando classi che concorrono nel instanziazione multipla di stesse entitá. Tutti questi sforzi sono confluiti nello sviluppo di un tool chiamato LODeX, il quale ha lo scopo di fornire al utente un riassunto di alto livello della struttura di una sorgente LOD, fornendo anche la possibilitá di costruire query visuali in modo tale da supportare lutente nel esplorazione e lanalisi di un dataset prima sconosciuto.

Tutte le tecniche proposte in questa tesi sono state valutate e confrontate rispetto lo stato del arte nei rispettivi campi: viene proposto una valutazione delle prestazioni riguardante il modulo di LODeX dedicato al estrazione degl 'indici; le tecniche di schema summarization sono state valutate secondo metriche proposte nel campo del ontology summarization; infine, si é valutato la portabilitá e l 'usabilitá di LODeX.

Nel ultimo capitolo della tesi viene presentata una nuova tecnica per il calcolo della similaritá tra documenti, chiamata CSA (*Context Semantic Analisys*), che sfrutta le informazioni contenute in un knowledge graph. Questa tecnica é stata comparata con altri metodi di stima della similaritá tra documenti utilizzando un dataset di documenti contenente misure di similaritá espresse da esseri umani come benchmark.

Acknowledgments

I owe my deepest gratitude to my supervisor, Professor Sonia Bergamaschi, for her precious support and guide during my Ph.D.

I sincerely thank all the members of the DBGROUP at the University of Modena and Reggio Emilia and in particular my colleagues of the laboratory (in alphabetic order) Francesco Guerra, Laura Po, Giovanni Simonini and Song Zhu for creating such a great friendship in the laboratory and during the University life.

I gratefully and sincerely thank Professor Amit Sheth for the opportunity she gave me to spend a profitable period abroad at the Kno.e.sis center – Wright State University. I gratefully acknowledge to the colleagues at Kno.e.sis Riccardo, Alan, François, Pavan and Pramod for their advice and their willingness to share their bright thoughts with me, which were very fruitful for shaping up my ideas and research.

To my roommate Francesco, Biagio and Andrea for their support and for the beautiful time spent together.

My parents deserve a special thank for their invaluable support and help throughout all my studies. Marzia and Morena, thanks for being supportive and caring sisters.

Finally, I would like to thank everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

The work on this thesis was partially accomplished in the framework of a PhD program organized by the Global Grant Spinner 2013 and funded by the European Social Fund and the Emilia Romagna Region.

Contents

List of Figures

Chapter 1

Introduction

The continuous efforts provided by the Semantic Web community in developing standardized ways for defining semantically rich data, jointly to the Open access trends, drove the massive publication of datasets as Linked Open Data[BL06]. The main technologies involved in the Linked Data growth are web infrastructures, like URIs and HTTP, and Semantic Web standards, like RDF[GK04], RDFS[BG04], OWL[MVH+04] and SPARQL[HS10]. RDF, Resource Description Framework, is the key standard in the Linked Data context, indeed, it allows to represent data according to graph data model. This kind of data model allows a versatile and powerful way of describing any kind of data, but this ductility leads to complex problematics when we have to work with this kind of data given that schema information are not usually available. Indeed, the RDFS and OWL standards are not thought for defining schema constraints on RDF graph, but they are designed for defining constraints within an ontology for enabling reasoning[Wan+04]. However, a peculiar characteristic of the graph data model[Kun90] is that the structure, even if is not explicitly defined, is implicit in the data itself. The exploitation of this important concept is the foundation of this thesis which is divided in two part. The first part contains my main research topic and the second part is the result of my research effort during a 7 month period that I spent abroad at the Kno.e.sis research center (Dayton, OH) in the United States.

The first part is composed by 4 Chapters and the focus is the definition of a theoretical model, called Schema Summary, with the goal of describing the structure of a generic LOD source. This model is the foundation upon which I developed a tool called LODeX[Ben15]. This tool is able to automatically extract the Schema Summary of a generic RDF source and serve it to the User in an interactive visualization. LODeX allows also the user of building visual query upon the Schema Summary and it contains a SPARQL compiler able to translate a visual query in a SPARQL one. This part is divided as follow: Chapter 2 contains the formal definition of the Schema Summary; Chapter 3 describe the first module of LODeX, the Index Extractor, able to automatically extract indexes from datasets reachable through a SPARQL interface; Chapter 4 contains the description of LODeX and the results of several evaluations of the tool, including an usability evaluation; finally, Chapter 5, contains an extension of the Schema Summary model, called Clustered Schema Summary, able to represent the structure of huge datasets; Chapter 5 contains also an evaluation of both Schema Summary and Clustered Schema Summary according to ontology summarization metrics.

The second part contains only Chapter 6 and it describes a novel technique, called Context Semantic Analysis. This technique leverages the information contained in a knowledge base, like DBpedia[Aue+07], for representing the context of a document as a vector. Then, this vector representation can be used for estimating the inter-document similarity based on the context. I combined CSA with other techniques of similarity based on the text and I evaluated it according to scores of similarity provided by humans.

Part I

Schema Summary

Chapter 2

The Schema Summary Model

2.1 Introduction

The actual LOD Cloud contains more then one thousand of interlinked datasets and several billions of RDF triples. The production of Linked Open Data (LOD) is encouraged by the Linking Open Data community and by the Open Governament project[1]. Linked data has become the preferred channel for the dissemination of e-government data and this results also in the LOD cloud, where one fifth of the datasets are on government domain [2].

Understanding a large and unfamiliar LOD dataset becomes a key challenge when reusing it. One of the main reason is that it is often difficult to get the overall view of a large dataset. This becomes even more problematic when users have barely knowledge of Semantic Web technologies, essentials for interacting with this kind of source.

A common practice in the Linked Open Data ecosystem is to reuse external vocabularies, i.e. datasets defined in external sources, for creating a common data space. This practice has allowed the rise of reference vocabularies in specific domains: *foaf* is used to describe people in around the 69% of LOD datasets ; *dce*

[1]`http://opengovernmentdata.org/`

[2]`http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/`

(Dublin Core Metadata Element Set)[3] is used to describe metadata of resources on the web by more than half of the datasets and *skos* is used to share and link knowledge organization systems via the Semantic Web, etc[4].

The Semantic Web has provided a schema language such as RDF Schema (RDFS) and a ontology definition language as OWL which allow for adding rich semantics to the dataset. However, several LOD datasets do not make an extensive use of RDFS and OWL, that is, there is an implicit knowledge, hidden in the datasets, that is not described by RDFS or OWL triples. This behavior is primarily due to automatic translation of dataset to the RDF data model from other data model (i.e. Relational data model). Figure 2.1 exemplify this problem: here, the property *dbpedia:fax* (taken from the external source, dbpedia) links an instance of the class *organization* to its fax value. This means that an organization could have data about the fax, however, this information is not conveyed by any RDFS triples, thus it is completely absent in the intensional knowledge of the dataset.

The use of resources (classes and properties) defined externally causes that many LOD sources are extremely complex to understand, since the user can not have a precise overview of the classes and properties used within the dataset without manually explore the source through SPARQL queries. Figure 2.2 reports the number of external resources (classes/properties) used in a dataset that are not contained in the intensional knowledge (i.e. not formally defined internally). More than a half of the datasets use at least 90% of external classes and properties[5], thus, it is easy to understand why the representation of the sole internal content of a LOD source miserably fails in providing an overall representation of the dataset.

These observations are the motivations behind my work of defining a model for the creation of a representative Schema Summary for a LOD sources aiming to facilitate the understanding of an unknown source and help users in making sense of an unfamiliar dataset. In this Chapter, I first explain the distinction between

[3]`http://dublincore.org/documents/dces`

[4]These statistics are reported in `http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/`

[5]The graphs are drawn starting from the analysis conducted on 185 LOD datasets.

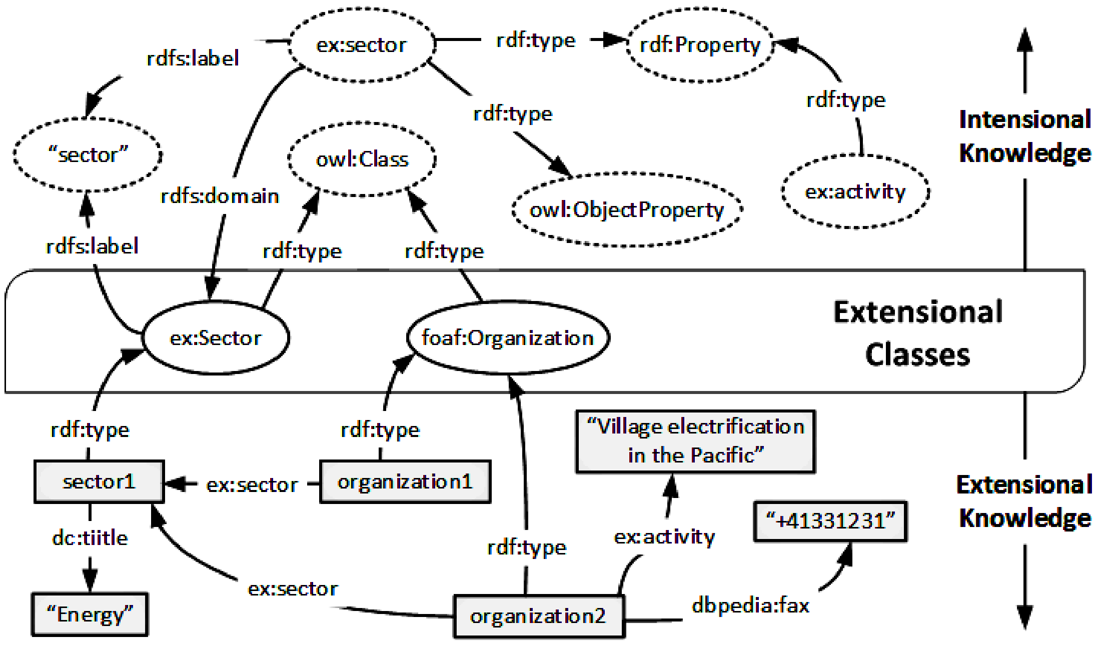


Figure 2.1: Example of content of a generic RDF graph belonging to the LOD cloud.

intensional and *extensional* knowledge, then, I present the theoretical definition of the Schema Summary, a model that conveys the implicit structure of the LOD dataset, by including the main classes and the properties used in it.

2.2 Intensional and Extensional Knowledge in LOD sources

The LOD Cloud consists of a huge number of SPARQL endpoints, each aiming to describe a knowledge base of a specific domain. The language used to describe data is RDF, while RDFS and OWL are used to represent intensional knowledge[PH04][HM03].

I can think the entire set of RDF triples partitioned between *intensional knowledge* and *extensional knowledge*. The triples belonging to the *intensional knowledge* define the terminology used in the dataset; they are expressed in RDF, however, they can be usually interpreted through RDFS or OWL and seen, with some

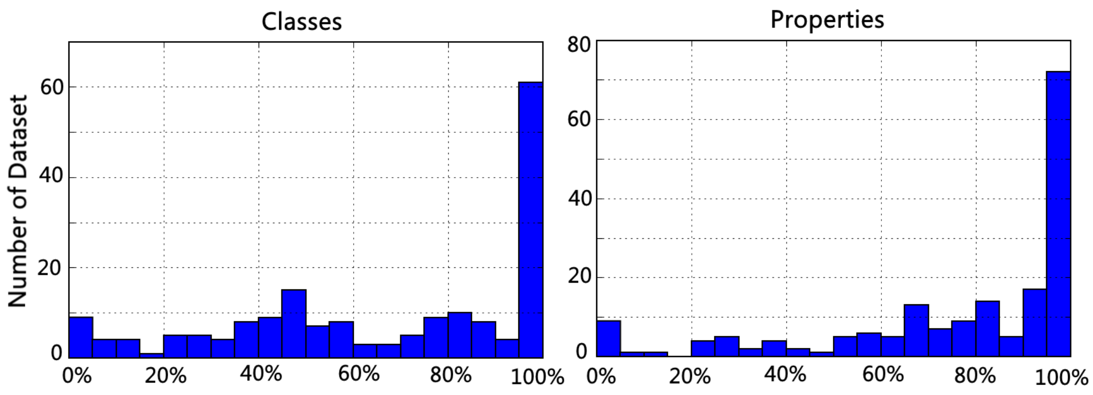


Figure 2.2: Percentage distribution of external classes/properties used in the LOD datasets.

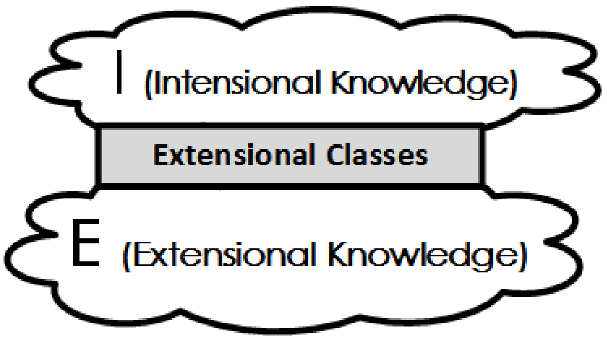


Figure 2.3: Data structure of a generic LOD dataset

restrictions, as the T-Box components of the knowledge base described in the end-point[LN94]. The *extensional knowledge* triples usually cover most of the datasets and contain the entities of the real world described in the dataset. Usually, the *extensional knowledge* is described through RDF instances which compose the A-box of the knowledge base[LN94].

These two kinds of knowledge are distinct, according to their semantic meaning, but they are joint forming a unique RDF graph. These two portion are connected by particular classes, that I call extensional classes, usually defined in *intensional knowledge* and instantiated in *extensional* one that act as a bridge among the two resources (as represented in Figure 2.3 and is shown in an example in Figure 2.3).

A well designed dataset should contain both intensional and extensional knowledge, however by analyzing a large number of endpoints I have observed that this is generally not true. Sometimes, LOD datasets are biased toward a kind of knowledge (intensional or extensional). For example, there are LOD sources containing ontologies with low number of instances, on the other hand, some datasets include only one type class (*owl:Class*) and a large number of instances that are improperly used as such. In a lot of cases, LOD datasets define only the extensional knowledge, thus they do not include the description of the used vocabulary within instances (i.e. this happens quite often when Open Data are published as RDF sources). These design issues are mainly caused by a large use of automated translation tools. For example, there are plenty of techniques for generating an OWL version of an ontology expressed with other standards as DAML+OIL[Hor+02] or RRF[Fie+04] and also the W3C consortium is spending many efforts in defining technologies able to translate data contained in Relational Databases into the RDF data model, called RDB2RDF[6][Sah+09].

2.3 Model definition

Each RDF graph is composed by a set vertices V and a set of labeled edges E. The vertices can be divided in 3 disjoint sets: the URIs U, the blank nodes B and literals L. Two vertices connected by an edge represent a statement. Each statement is stored into a <*subject*,*predicate*,*object*> triple, where *subject* $\in U \cup B$, *object* $\in V$ and *predicate* $\in E$. I can define the whole RDF graph, according to [CWL14], as a set of triples RG.

Definition 1 $RG \subseteq (U \cup B) \times E \times V$

Each triple belonging to an RDF graph defines a relation between two nodes, and the kind of relation is made explicit through the value of the property. In particular, the RDF property *rdf:type* is used to state that a certain resource is an instance of a class. I can define the set of classes (Cs).

Definition 2 $Cs = \{c | <i, rdf{:}type, c> \in RG \wedge i \in (U \cup B)\}$

[6]http://www.w3.org/2001/sw/rdb2rdf

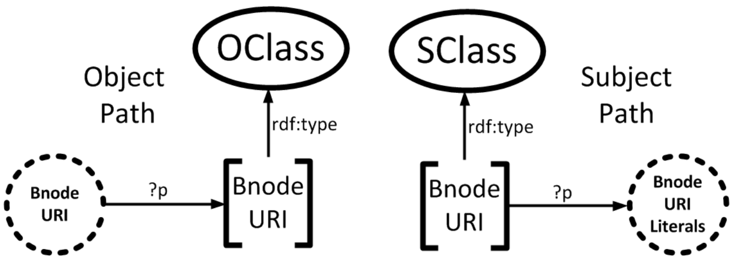


Figure 2.4: Subject and Object Path

Looking at the triples belonging to the extensional knowledge, I can find three main patterns that are responsible for defining the structure of the data contained in it. I call these patterns: *Subject Class* (Sc), *Subject Class to literal* (Scl) and *Object Class* (Oc). A graphical representation of these patterns is shown in Figure 2.4, while the formal definition is given in the following.

Definition 3 $Sc = \{(c,p) | $*<i,rdf:type,c>*$\in RG \wedge$*<i,p,u>*$\in RG \wedge i \in (U \cup B) \wedge u \in (U \cup B)\}$

Definition 4 $Scl = \{(c,p) | $*<i,rdf:type,c>*$\in RG \wedge$*<i,p,l>*$\in RG \wedge i \in (U \cup B) \wedge l \in (L)\}$

Definition 5 $Oc = \{(c,p) | $*<i,rdf:type,c>*$\in RG \wedge$*<u,p,i>*$\in RG \wedge i \in (U \cup B) \wedge u \in (U \cup B)\}$

The patterns extracted on the example of Figure 2.1 are presented in Table 2.1.

Table 2.1: Extracted patterns from the example on Figure 2.1

Name	**Values**
Cs	{ owl:Class, rdf:Prop, foaf:Organization, ex:Sector, owl:ObjectProperty }
Sc	{ (foaf:Organization,ex:sector), *(owl:Class,rdfs:label), (rdf:Property,rdfs:label), (rdfs:Property,rdfs:domain), (owl:ObjectProperty,rdfs:label), (owl:ObjectProperty,rdfs:domain)* }
Scl	{ (ex:Sector,dc:title), (foaf:Organization,ex:activity), (foaf:Organization,dbpedia:fax) }
Oc	{ (ex:Sector,ex:sector) }

The Sc, Scl and Oc patterns unveil all the classes and the properties used within the dataset even if they are not explicitly defined in the intensional knowledge. These information enrich the comprehension of the source itself and are used as input in the building of the Schema Summary.

2.3.1 Schema Summary

Definition 6 (Schema Summary) *A Schema Summary S, derived from a RDF dataset, is a pseudograph: S = <C, P, s, o, A, m,Σ_l, l, count>, where:*

- *C contains a set of c, where c is a Class of the RDF dataset. The elements of C represent the node of the pseudograph.*
- *P contains the properties between Classes of the RDF dataset. The elements of P represent the edges of the pseudograph.*
- *s: P $\rightarrow$ C is a function that assigns to each property p $\in$ P its source class c $\in$ C.*
- *o: P $\rightarrow$ C is a function that assigns to each property p $\in$ P its object class c $\in$ C.*
- *A contains the attributes of Classes of the RDF dataset.*
- *m: A $\rightarrow$ C is a function that maps each attribute a $\in$ A to the class c $\in$ C to which it refers.*
- *Σ_l is the finite alphabet of the available labels.*
- *l: $(C \cup P \cup A) \rightarrow \Sigma_l$ is a function that assigns to each class, property or attribute its label.*
- *count: $(C \cup P \cup A) \rightarrow \mathbb{N}$ is a function that assigns to each property or attribute the number of times it appears in the RDF dataset, and to each class the number of instances of the class itself.*

```
Data: $Sc, Scl, Oc, Cs, count$
Result: $S(SchemaSummary)$
S=$\emptyset$;
Filter(Sc,Scl,Oc);
forall the $sc \in Sc$ do
    if $sc.c \in Cs$ and $\exists oc \in Oc \mid sc.p = oc.p$ then
        if $sc.c \notin S.\Sigma_l$ then
            find c $\in$ Cs where c = sc.c;
            add to S.count(c) count(c);
            add c to S.C;
            add c to S.$\Sigma_l$ and create mapping in S.l;
        end
        if $oc.c \notin S.\Sigma_l$ then
            find c $\in$ Cs where c = oc.c;
            add to S.count(c) count(c);
            add c to S.C;
            add c to S.$\Sigma_l$ and create mapping in S.l;
        end
        n=MIN(count(sc),count(oc));
        p =sc.p, add p to S.P ;
        add mapping in S.s and S.o;
        add to S.count(p) n;
        add p to S.$\Sigma_l$ and create mapping in S.l;
        decrease count(sc) and count(oc) of n;
        if count(sc) or count(oc) = 0 then
            remove this element from Sc or Oc;
        end
    end
end
forall the $scl \in Scl$ do
    if $scl.c \in Cs$ then
        a = scl.p, add a to S.A;
        add mapping in S.m from S.C(scl.c) to a;
        add a to S.$\Sigma_l$ and create mapping in S.l;
        add to S.count(a) count(scl);
    end
end
```

Algorithm 1: Schema Summary generation algorithm

The *SC*, *SCl*, *OC*, *Cs* and the function $count()$, able to return the number of occurrences for each pattern, are the input of the Schema Summary generation algorithm, while the output is a a pseudograph *S*.

Before executing the core of the algorithm, it is necessary to filter some elements of the input indexes (as reported in line 2). In particular, I filter the elements that belongs to the intensional knowledge.

The core of the generation algorithm is situated from line 3 to line 27 of the Algorithm 1, where Sc and Oc are combined in order to discover the nodes and edges belonging to pseudograph S; in this phase the alphabet Σ_l and all the mapping functions are populated . A key role is played by the number of occurrences for the paths in Sc and Oc. In fact, thanks to this value it is possible to know how many properties coming from instances of a *Subject Class* are directed to instances of an *Object Class*. If elements of Sc and Oc correspond, through their property, their value of n are decremented. Finally (from line 28 to 35), the list of attributes (A) with the respective labels and mappings are populated using the information contained in Scl.

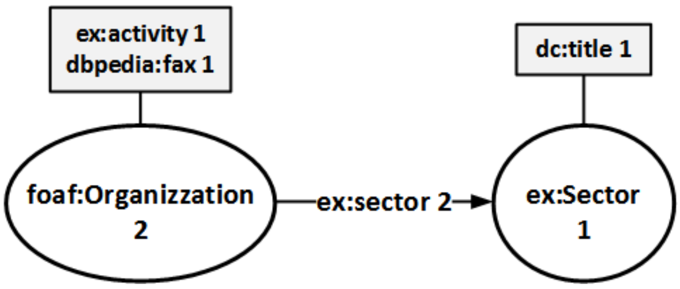


Figure 2.5: Schema Summary generated starting from the example of Figure 2.1.

Figure 2.5 depicts the Schema Summary derived from the datasets of Figure 2.1. The white circles represents the classes (C), while the attributes (A) are shown in the gray boxes. The edge depicts a property (P). Each element is equipped with a numerical value representing the number of occurrences (or the number of instances for the classes).

Chapter 3

Online Index Extraction from Linked Data Sources

3.1 Introduction

The possibility to expose any sort of data on the Web by exploiting a consolidated group of technologies of the *Semantic Web Stack*[Biz+08] is one of the main strengths of Linked Open Data. Several portals catalog LOD datasets on the Web; one of the main globally available Open Data catalogues is The Data Hub (formerly CKAN)[1]. These resources allow users to perform keyword search over the metadata associated to their list of LOD sources, but they do not provide search techniques based on structural information of these sources. As mention in[Jai+10], there is still a “lack of conceptual description of datasets". The documentation of a LOD source is produced by who published the data and, in many cases, it is incomplete or absent. Therefore, usage of LOD datasets requires a human being to identify the domain of the datasets and discriminate if they are relevant for his/her needs (usually by performing SPARQL queries).

The first step for making available the Schema Summary for each LOD dataset of the LOD cloud is define an automatic way for indexing these latter. In this Chapter, I define a set of indexes, called *Statistical Indexes*, able to structurally

[1]http://datahub.io

describe the source. The *Statistical Indexes* contain statistical information regarding both the intensional and extensional knowledge of a LOD source. The intensional knowledge contains the RDFS/OWL triples used to define a vocabulary or an ontology. The extensional knowledge is characterized by the instantiated classes, the properties between them and some graph patterns (ingoing and outgoing properties from instances of a specific class). Usually, the intensional knowledge is defined as the terminology used for characterizing the assertions, i.e. the extensional knowledge. As reported in [Got+12; AG05], the structure of an RDF source is implicitly defined by its set of triples; information about the schema resides explicitly in the instantiation of the classes and implicitly in the use of the properties among these.

The indexes can have different uses, but primarily they represent a good documentation of the dataset to which they refer. Indeed, a knowledge engineer might be interested to describe a specific environment by reusing available vocabularies or ontologies if he/she easily understands their intensional contents. Otherwise, a data scientist can explore the lists of elements characterizing the extensional knowledge (e.g. occurrence of outgoing properties from a particular type of instances) of a specific dataset to easily build the SPARQL queries he/she needs to extract the data he/she is looking for. The *Statistical Indexes* can also be used with other purposes: to support search engines (e.g DataHub) for dataset selection according to the ingoing or outgoing properties from instances of a classes ranked according to the number of occurrences or to support tools able to generate queries. In this Chapter, it is also described a software component, called *Index Extractor*, able to automatically extract the *Statistical Indexes*. The *Index Extractor* only deals with SPARQL endpoints, differently from other approaches that work with a dump of the RDF Database stored locally (e.g. [Kon+12], [Aue+12] and [CPF13]). This choice has been made in order to develop a module able to work as an online service. The *Index Extractor* takes as input just the URL of a SPARQL endpoint and produces the necessary queries for extracting the *Statistical Indexes*. One of the main problems I encounter when dealing with SPARQL endpoints is the heterogeneity on the performance of different implementations of

SPARQL endpoint. Indeed, it happens that several SPARQL aggregation queries may trigger timeout errors; the *Index Extractor* handles this problem by generating an higher number of low-complexity queries able to return the same information into smaller chunks of data. I called this mechanisms *pattern strategy*, and it will be described in Section 3.4.1.

This Chapter is structured as follows. Next Section outlines some relevant works connected to this topic or papers that have inspired the development of this tool. An overview of the *Index Extractor* and its architecture is depicted in Section 3.3. Section 3.4 details the extraction of the *Statistical Indexes*. In Section 3.5 some tests are reported and finally, conclusions and some ideas for future work are described in Section 3.6.

3.2 Related Work

In the literature, I can find several works in which a summary or a set of descriptors are extracted from a LOD source. In [CPF13], authors divide these techniques in two groups, *triples-level* and *instances-level* summaries, according to the granularity with which the sources are scanned and indexed.

The triples-level techniques inspect the content of the RDF dataset scanning each triple, and then they usually build an index containing statistical information regarding the type of these triples. SchemEx [Kon+12] is an example of work belonging to this group; here, dumps of RDF graphs are indexed for supporting the definition of queries by users. Differently from the techniques described in this Chapter, this approach does not consider the class instances, thus it is also not able to retrieve the properties among classes.

RDFstats[LW09], instead, defines a vocabulary and an algorithm able to collects statistics about the triples belonging to an RDF dataset that can be used to build histograms and document the source. The information extracted by RDFstats has a low granularity and it can not be used as documentation of a RDF dataset, because it does not contain any sort of structural information about the RDF source. Another valuable example of these works is LODStats[Aue+12].

Here, RDF graphs are scanned at triple level and in the post-processing phase, structural information such as the class and property hierarchies are discovered.

In the instance-level approach, an RDF graph is inspected by taking into account RDF, RDFS and OWL primitives and their semantics, in order to detect structural information pervasive in the source. In this group I can find two important works [Har+10; PKK12] in which the proposed instance-level summary can support efficient and federated query evaluation. Another valuable example of this group is [BL12] in which an approach that allows to enrich knowledge bases with OWL2 axioms is described. The information extracted by [BL12] overlaps the intensional knowledge that can be present within the triples of a dataset; the *Index Extractor* is able to extract the triples belonging the intensional knowledge through an ad-hoc algorithm, reducing the time and the complexity of this task.

All these techniques have been tested using a small number of datasets and they take as input the dump of an RDF graph; instead, the *Index Extractor* has been designed to be used with a wide number of SPARQL endpoints in an online environment.

The most important example of LOD indexes are the Void descriptors[Ale+09]. The Void descriptors are a W3C standard used for expressing metadata about RDF datasets. In particular, they are primarily used to describe links among different datasets rather than the structure of the dataset itself. Despite they report valuable information, their production is demanded to the publisher of the LOD dataset, thus, only the 13.82%[2] of datasets are equipped with VOID descriptors. The *Statistical Indexes* include some VOID descriptors but they contain further indexes to supply more detailed information about the structure of the dataset.

3.3 Architectural Overview

The *Index Extractor* aims to be totally automatic in the extraction and production of the indexes. Therefore, it does not require any kind of a priori knowledge about

[2]http://sparqles.okfn.org/discoverability

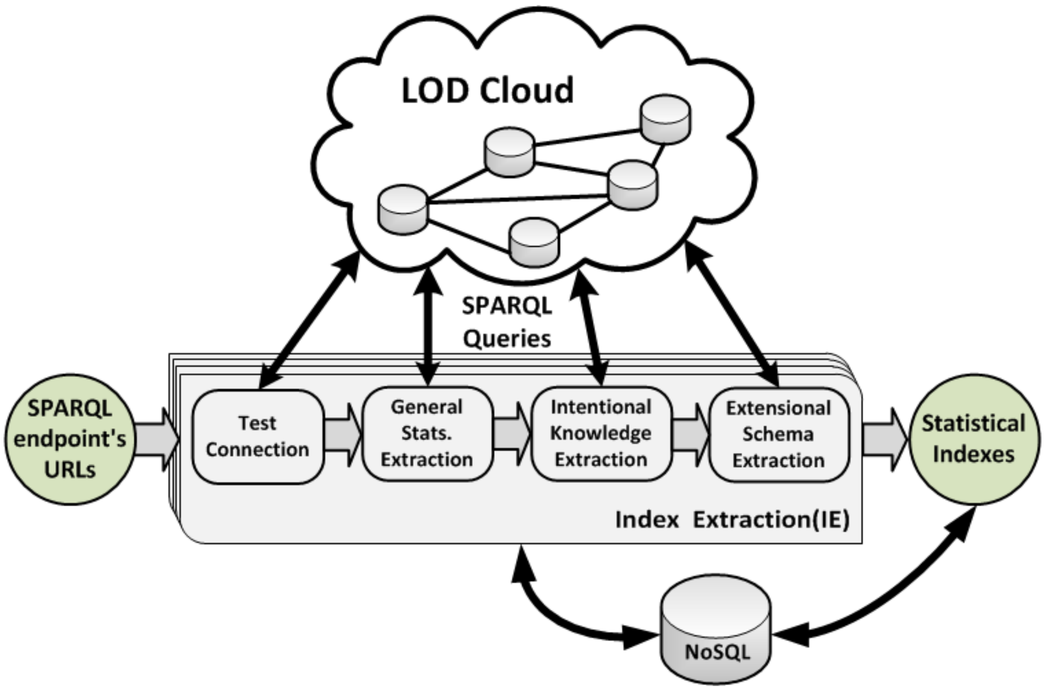


Figure 3.1: Index Extractor architecture

the dataset on which it works.

Figure 3.1 illustrates the architecture of the *Index Extractor* (IE). The component takes as input just the URL of a SPARQL endpoint and then it performs a set of steps with the goal of generating the queries, able to extract structural and statistical information about the source. The IE has been designed in order to maximize the compatibility with LOD sources and minimize the costs in terms of time and computational complexity. Two different algorithms, based on a set of SPARQL patterns, have been designed on order to extract the most relevant intensional and extensional information from heterogeneous LOD datasets. Finally, the indexes are stored into a NoSQL Database. I have chosen a NoSQL document database server, MongoDB[3][Ban11], because it allows a flexible representation of the indexes and, in particular, it can easily manage heterogeneous lists of elements.

The architecture has been designed to parallel the processing of multiple end-

[3]https://www.mongodb.org

points, thus, exploiting the idle times caused by response-time delays of single endpoints. Moving part of the computational cost of the extraction process on the endpoint can improve the performance, but it brings some drawbacks. First of all, a portion of the queries generated by IE needs some operators introduced with SPARQL 1.1[HS10], thus, an endpoint must be compatible with this standard. Another issue regards the heterogeneity of the implementation of SPARQL endpoints that affects their performance. Some endpoints are not able to answer to some queries before the timeout expires. To avoid these problems, I have limited the use of SPARQL 1.1 operators and I have defined a particular *pattern strategies* to scale the complexity of the queries.

3.4 Index Extraction Process

The *Statistical Indexes* extracted by the IE component through SPARQL queries[4] can be grouped in three categories, according to the kind of knowledge they stored: *Generic*, *Intensional*, *Extensional* (see also Table 3.1).

Table 3.1: Statistical Indexes. Legend: Cn: class name; Pn: property name; s: subject; p: property; o: object; n: number of times a path (or a property) exists; nI: number of instances

Name	Description	Structure	Category
t	Number of Triples	Integer	Generic
c	Number of Instantiated Classes	Integer	
i	Number of Instances	Integer	
Cl	Instantiated Class list	List(Cn,nI)	
Pl	Property list	List(Pn,n)	
IK	Intensional Knowledge Triples	List(s,p,o)	Intensional
Sc	Subject Class	List (s,p,n)	Extensional
Scl	Subject Class to Literal	List (s,p,n)	
Oc	Object Class	List (o,p,n)	

In the *Generic* group, all the information regarding the size and the complexity of the dataset are reported. In particular, the first three elements (**t**, **c**, **i**) give an

[4]A complete list of the query patterns is available at http://dbgroup.unimo.it/lodexQueries

insight of the dimension of the RDF graph; while, the last two components (**Cl** and **Pl**) contain information about the classes and the properties usage. The queries used to extract these values refer to those used to create the Void Descriptors[5] of a dataset[Ale+09]. As mentioned before in Section 3.2, not all the LOD sources contains VOID descriptors, therefore, I added them in my list.

The *Intensional* group contains only the **IK** index, i.e. the list of all the triples that characterize the intensional knowledge of the dataset. The queries used to extract these triples are based on a simple triple pattern (subject, predicate and object), in which the subject is iteratively replaced with the URIs representing the constraints of the ontology, this in order to traverse the RDF graph and extract the intensional knowledge.

The *Extensional* group contains the information regarding the distribution of instances within the source. The first two indexes (**Sc**, **Scl**) refer to Subject paths (the first has an URI as object and the second a literal), while the last (**Oc**) regards the Object path (see Figure 2.4). Each of these list is described by three elements: **s/o** that is a *Subject Class/Object Class*, **p** that refers to a property and **n** that represent the number of times the path is used in the dataset. A formal description of these pattern can be found in Section 2.3.

In the first step the IE component tests the connection to the endpoint (as shown in Figure 3.1) and it also checks the compatibility of the endpoint with the SPARQL 1.1 operators used in the queries. After this, the first three indexes (**t**, **c** and **i**) are extracted through simple SPARQL queries. Then, **Cl** and **Pl** are extracted using the *pattern strategies* for dealing with possible failure of the end-point. The extraction of the intensional knowledge, such as **IK**, makes use of an iterative algorithm. In the end, the **Sc**, **Scl** and **Oc** indexes exploit *pattern strategies* to increase the success rate of their extraction.

The IE component exploits different patterns through which it can produce queries of different complexity. With the introduction of the version 1.1 of SPARQL, it is possible to collect aggregated information about a specific Basic Graph Patterns (BGP) [HBF09] over an RDF graph using the operator GROUP

[5]https://code.google.com/p/void-impl/wiki/SPARQLQueriesForStatistics

BY. However, in many cases endpoints hosting large RDF datasets are not capable of providing a response before the timeout expires. I have chosen to use a restricted group of operators to minimize the complexity of the queries generated, i.e. GROUP BY, FILTER, COUNT, DISTINCT, AND. As stated in [PAG09a], the evaluation of an expression containing AND and FILTER can be solved in linear time, while the operator GROUP BY is more expensive in terms of performance. The *pattern strategy* technique is able to extract the same information resulting from complex GROUP BY queries by using a higher number of low-complexity queries and it will be described in detain in the next Section

3.4.1 Pattern Strategies

I often stumbled across errors triggered by endpoints due to performance issues. In particular, this problem occurred when extracting the Subject Path (**Sc** and **Scl**), the Object Path (**Oc**) and in few cases the classes and properties lists (**Cl** and **Pl**). The indexes matching these patterns could be extracted using just one query for pattern, but this operation has an high cost for the endpoint and, in most cases, a timeout error occurs. Hence, I have designed a *pattern strategy* able to handle this type of error and scale the complexity of the SPARQL 1.1 query.

In Figure 3.2 you can see a representation of the pattern strategy used to complete the extraction of the **Sc** index. By using the first query, it is possible to extract all the information in one go. If the endpoint is not able to answer to the first query, the strategy switches to the second step. In this case, a query for each class in **Cl** is generated, then, each successful response returns an element of **Sc**, while, if an error occurs the current class is added to the set of **ErrClass**. At the end of the second step, if some error still exists, the strategy tries to download the two items that compose each element of **Sc** (property name, and property count) separately. Thus, in the third and fourth step the queries are generated for discovering the properties related to the current class by a Subject Path; this information is temporarily stored in a list called **TmpSc**, which is taken as input by the last step, where the queries are generated for completing the partial results contained in **TmpSc** with the information regarding the number of times these paths are present in the source graph. It is worth noting that while the first and second queries exploit the GROUP BY operator, the others do not. Therefore, it is possible to compute the **Sc** index even without the GROUP BY operator.

Similar strategies (with different queries) are used to complete the extraction of **Scl**,**Oc**,**Cl** and **Pl** indexes.

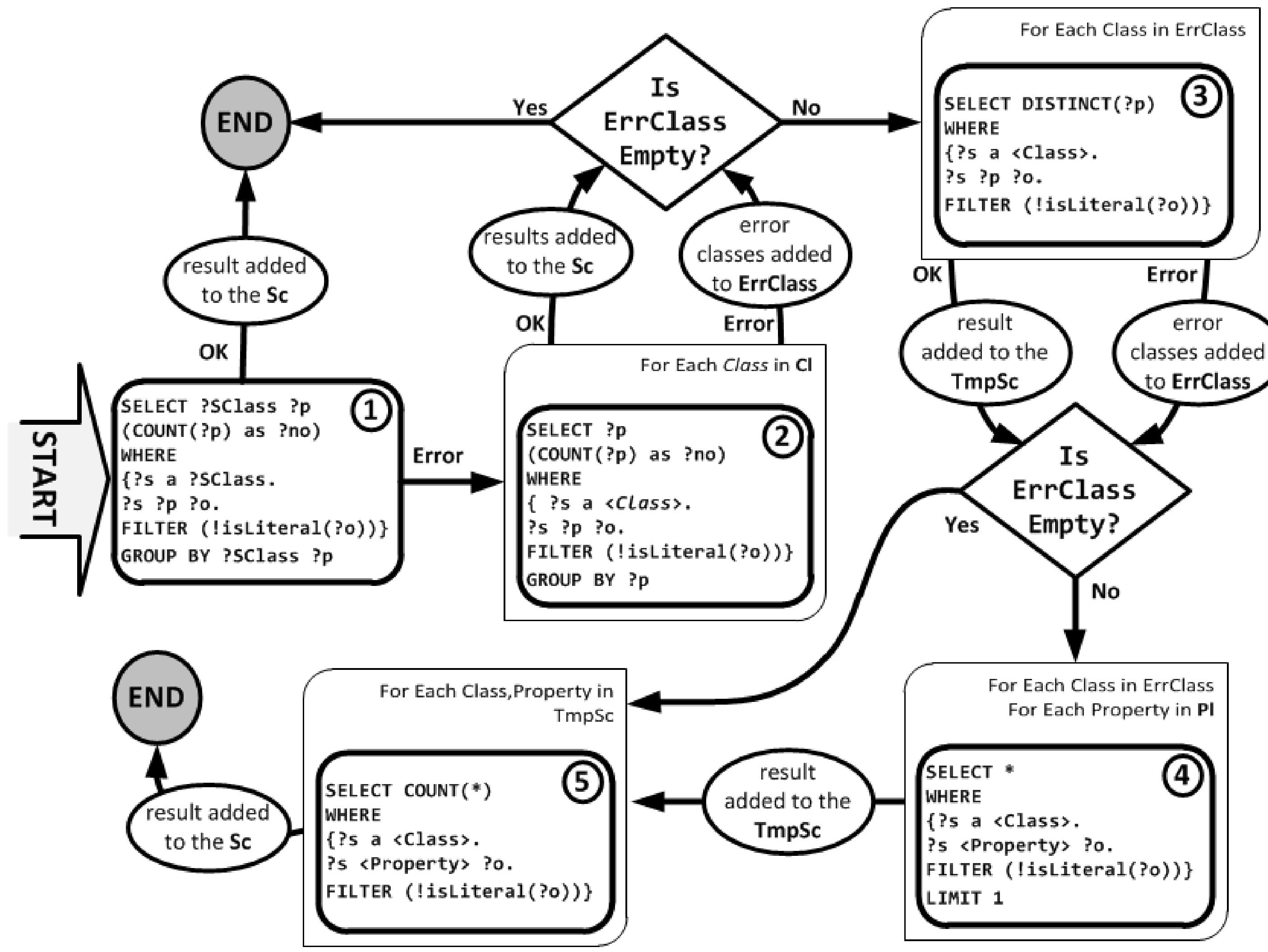
3.4.2 Intensional Knowledge Extraction Algorithm

```
Data: Cl, Pl
Result: Ik
Qn=$\emptyset$, Fn=Cl.cn $\cup$ Pl.pn;
while |Qn| < |Fn| do
    forall the node in Fn - Qn do
        results$\leftarrow$generate query for node and query the endpoint;
        add node to Qn;
        forall the r in results do
            add r to Ik;
            if r.o is not a Literal then
                add r.o to Fn;
        end
    end
end
```

Algorithm 2: Intensional Knowledge Extraction Algorithm

The intensional knowledge contained in a generic dataset usually consists in few triples within the dataset with an high information load. Therefore, it is important to pull out all these triples. To achieve this goal I designed an iterative algorithm able to traverse the RDF graph and extract the IK index (the pseudocode is presented in Algorithm 2)

An iterative algorithm is well suited for traversing any sort of graph, but I have to properly choose the starting point and the traversing conditions for making the algorithm efficient and being sure that only the triples desired are extracted. Moreover, the algorithm has to avoid trespassing in the extensional knowledge with the risk of downloading the entire graph. Fortunately, I can take advantage of the structure induced by the RDF standard. Indeed, the instantiation primitive


START
OK
result added to the Sc
END
Error
For Each Class in Cl
OK
results added to the Sc
Error
error classes added to ErrClass
Is ErrClass Empty?
Yes
END
No
For Each Class in ErrClass
OK
result added to the TmpSc
Error
error classes added to ErrClass
Is ErrClass Empty?
Yes
No
For Each Class in ErrClass
For Each Property in Pl
result added to the TmpSc
For Each Class,Property in TmpSc
result added to the Sc
END
1
2
3
4
5

```
SELECT ?SClass ?p
(COUNT(?p) as ?no)
WHERE
{?s a ?SClass.
?s ?p ?o.
FILTER (!isLiteral(?o))}
GROUP BY ?SClass ?p
```

```
SELECT ?p
(COUNT(?p) as ?no)
WHERE
{ ?s a <Class>.
?s ?p ?o.
FILTER (!isLiteral(?o))}
GROUP BY ?p
```

```
SELECT DISTINCT(?p)
WHERE
{?s a <Class>.
?s ?p ?o.
FILTER (!isLiteral(?o))}
```

```
SELECT *
WHERE
{?s a <Class>.
?s <Property> ?o.
FILTER (!isLiteral(?o))}
LIMIT 1
```

```
SELECT COUNT(*)
WHERE
{?s a <Class>.
?s <Property> ?o.
FILTER (!isLiteral(?o))}
```

Figure 3.2: Pattern strategy for extraction of the *Sc* index.

of the RDF language is a recognizable triple in which the subject can be a URI or a blank node, the predicate is *rdf:type* and the object is an URI (representing a class). Thus, I can create a group of SPARQL queries using the class list (**Cl**); each of these queries will be composed by a simple triple pattern in which I bind each of the subjects with an element of **Cl**, and use them to start traversing the portion of the graph containing the intensional knowledge. Moreover, in order to include the formal definition of properties, it is necessary to generate a set SPARQL queries also binding the list of properties (**Pl**). Indeed, the properties URIs are used as subject of a triple only in the intensional knowledge. The pseudo-code of the intensional knowledge Extraction Algorithm is shown in the beginning of this Section.

I have tested the algorithm on several datasets and it always stopped without erroneously download any triples belonging to extensional knowledge. The number of iterations can give an estimation of the ontology deepness and complexity. Usually, it stop after around 5 iterations and it reached a maximum of 22 iterations for the most complex ontologies.

3.5 Test and Performance Evaluation

The IE component has been tested on the entire set of datasets taken from *SPARQL Endpoint Status*[6], a specialized application that recursively monitors the availability of public SPARQL endpoints contained in DataHub. Table 3.2 reports the number of datasets that were examined. Here, 469 endpoints have been tested, but unfortunately only 244 were online when tests were performed (May 2014). Moreover, during the connection test phase, I checked the compliance of each endpoint with SPARQL 1.1 operators. For this reason the number of suitable endpoints decreased to 137. Since I use only a subsets of SPARQL 1.1 operators, the IE process was successfully performed on 56% of the sources (137/244). At the same time, the number of endpoints fully compatible with SPARQL 1.1 was much

[6]http://sparqles.okfn.org/

lower; they were only 14, so the 5%[7]. Also the pattern strategy has demonstrated its effectiveness by increasing the number of endpoints successfully indexed, from 33 to 107. On these datasets, I have also evaluated the behavior of the intensional knowledge Extraction Algorithm that usually stops after 5 iterations and only in few cases needs more iterations, till a maximum of 22 iterations.

In Table 3.3 statistics about the performance for the 107 datasets that have completed the extraction are shown. The average time of extraction is 6.12 minutes (the avg size of each dataset is 32 millions of triples). Thus, I have examined 3.45 billions of triples in 11 hours using a single process. I also tested a multi processes implementation of IE; by using 9 parallel processes the extraction time decreases to just 3.35 hours. This is an good result if compared to similar tools such as SchemEx that was able to analyze 2.17 billion of triples in 15 hours[Kon+12].

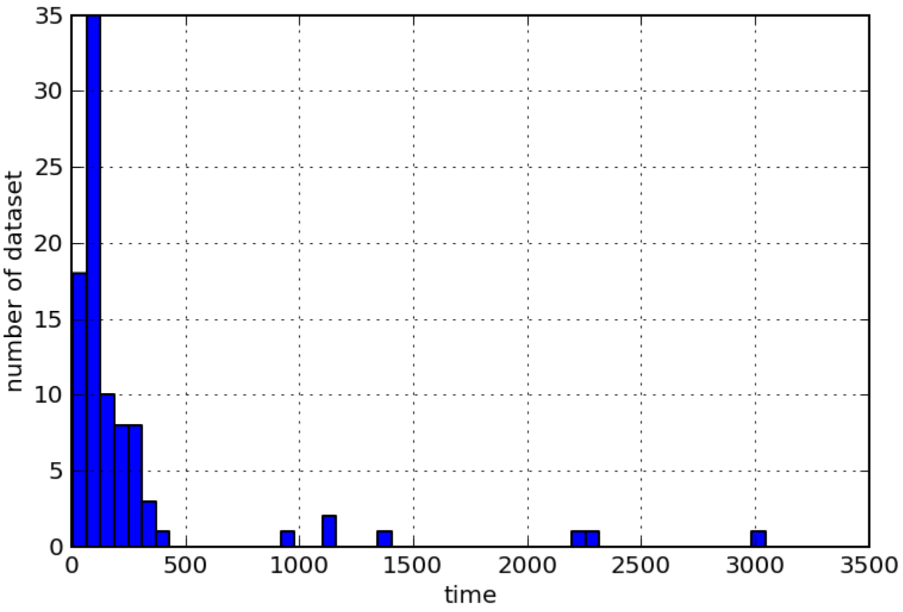


Figure 3.3: Distribution of the extraction time (s) for the datasets for which the Statistical Index was successfully extracted

Figure 3.3 reports the indexes extraction time of the 107 datasets that successfully completed the extraction. It can be seen that more than 90% of the datasets

[7]as reported in http://sparqles.okfn.org/interoperability on May 4th, 2014.

Table 3.2: Numerical information on the evaluated sources

Dataset URLs	469
Reachable datasets	244
SPARQL 1.1 compatible	137
Extraction completed	107
Extraction without Pattern Strategy	33

Table 3.3: Performance of the IE process on 107 datasets

AVG time of extraction	6,12 minutes
Total time (single process)	11,15 hours
Total time (9 processes)	3,35 hours
Total triples	3,45 billions

Table 3.4: Statistical indexes statistics over three datasets on the left. Pearson correlation between extraction time and other features on the right.

	KEGG Pathway	Dbnary	DBLP in RDF	P. correlation
Triples number	49.859.159	39.393.237	Error	0.72
Instance number	11633810	8217804	54939	0.56
Class number	32	42	6	0.44
Property number	161	171	25	0.50
Extraction Time	19 minutes	1,5 minutes	Error	1

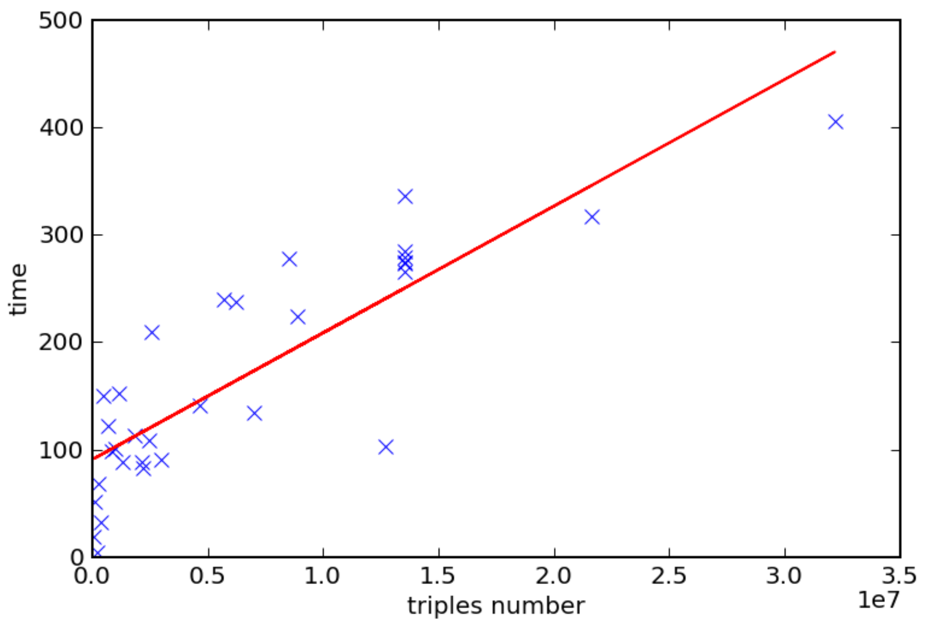


Figure 3.4: Extraction time (s) and number of triples

have completed the extraction in less than 500 seconds[8]. The regression line between the execution time and the number of triples for these datasets is draw in Figure 3.4.

The heterogeneity on the implementation of the SPARQL endpoints is one of the most critical aspects and it also dramatically affects the performances of IE. To highlight this issue, in the right part of Table 3.4, I have compared the characteristics of three datasets: KEGG Pathway (knowledge on the molecular interaction and reaction networks), Dbnary (wiktionary data for several languages) and DBLP in RDF (L3S). In terms of size and complexity the first two datasets are very similar, but the extraction time on the first dataset takes more than 10 times compared to the second. DBLP is a borderline case; although it is less complex than the first two datasets, the extraction process has not been completed.

I have also investigated which of the dataset features has the greater impact on the extraction time by using the Pearson product-moment correlation coefficient. The coefficient values are presented in the left part of Table 3.4. The number

[8]The tests ran on a portable machine (Operative System: Windows 7 - 64 bit, RAM: 6 GB, number of processors: 1, number of cores: 2).

of triples obtains the higher value of correlation. This proves that there is a high degree of linear dependence between the extraction time and the size of the dataset as previously demonstrated in Figure 3.4.

3.6 Conclusions And Future Work

Starting from the URL of a SPARQL endpoint, the IE component is able to automatically extract a set of statistical indexes describing the content of an RDF graph. In this Chapter, I presented the architecture and the algorithms composing the IE component and I proposed an evaluation over a significant number of LOD sources available on the SPARQL Endpoint Status portal. The results obtained are satisfactory both in terms of portability and performance.

Chapter 4

Visual Querying LOD sources with LODeX

4.1 Introduction

It has been eight years since Tim Berners-Lee designed the Linked Data Principles. Now the Web of Data consists of more than a thousand of datasets collecting several billion of triples[1]. The LOD dataset generation is also encouraged by the Open Access trends and its importance has been highlighted by the report[2] produced by the Open Data Barometer of the 2014: "In 2014 the G20 largest industrial economies followed up by pledging to advance open data as a tool against corruption, and the UN recognized the need for a Data Revolution to achieve global development goals". Although, the LOD cloud is growing more and more, navigation and visualization of Linked Data is still at the beginning.

Several portals, such as the well known Data Hub, catalog datasets that are available as LOD on the Web and provide keywords search methods to identify a dataset of interest. Usually, a user have to manually explore a new dataset using SPARQL queries to understand if a dataset really contains the information that he is looking for. It follows that a user with no SPARQL knowledge cannot become

[1]http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/

[2]http://barometer.opendataresearch.org/report/summary/

a consumer of the data contained in the LOD Cloud. Even for a skilled user this is not a easy task because there are no fixed modeling rules in designing the structure a LOD dataset; usually, external classes and properties are used within a dataset without formally define how they are related to the classes defined locally. Moreover, a great number of datasets is published without a real documentation that could help on revealing their structure.

In this Chapter I present LODeX a tool which is the direct implementation of the Schema Summary model and it contains the *Intdex Extractor* module. LODeX aims to solve the above issues in order to empower users without technical skills in exploring, understanding and extracting knowledge from a LOD dataset without any a-priori knowledge on the source itself. In particular, it provides: (1) an high level Schema Summary able to capture structural information of a LOD source, to enable classes and properties browsing; (2) a powerful and intuitive visual query builder, to empower users in the in-dept exploration of the instances of the source and eventually to generate a SPARQL query able to extract the piece of knowledge to which the user is concerned. The tool takes advantage of a query refinement panel and a SPARQL compiler that capture each change in the visual query and refreshes of the corresponding SPARQL query and its result.

In this Chapter, I describe LODeX and I test the portability of the tool on more than 300 datasets to demonstrate that my tool can be used with the great part of the datasets belonging the LOD cloud. Moreover, I conducted a usability evaluation in order to show the effectiveness of LODeX in representing the structure of a dataset and in supporting the user in building queries on an unknown LOD source. The results demonstrate the effectiveness of the tool and, further, highlight future lines of development.

The remainder of the Chapter is structured as follows. I discuss related work in Section 4.2. I draw the architecture and a motivation example in Section 4.3. Section 4.4 illustrates a use case scenario, while Section 4.5 reports the evaluation on portability and usability of the tool. Finally, Section 4.6 sketches the conclusion and the future lines of extension for LODeX.

	Availability Online	Scale		Querying			Result Visualization
		Whole Dataset	Instance Level	Visual Building	SPARQL Generation/Edit	By Keyword	
LOD Visualization	✓	✓					List and adv. vis.
ProLOD	*	✓					Adv. vis.
LODlive	*		✓			✓	Graph vis.
LODmilla	✓		✓			✓	Graph vis.
gFacet	*		✓	✓		✓	Graph vis.
iSPARQL	✓		✓		✓		List and adv. vis.
SPARKLIS	✓		✓			✓	List
LD Query Wizard	✓		✓	✓		✓	List and adv. vis.
LODeX	✓	✓	✓	✓	✓		List

Table 4.1: Visualization, exploration and query tools ($*$ it is provided an online demo)

4.2 Related Work

Several researchers have attempted to support users in LOD source visualization, browsing and in the definition of complex queries allowing fancy visualization of the results. Table 4.1 contains a comparison of different tools based on visualization and querying features[3].

As shown in the table, I can distinguish between two major groups: the tools that focus on providing an overall overview of the whole structure of the datasets and the tools that provide just an instance level view of the datasets and supply query functionalities.

In the first group, we can find LOD Visualization and ProLOD; tools that aim to provide to users an high level analysis of a LOD dataset. In particular, LOD Visualization is a prototype based on the Linked Data Visualization Model [BAG12], and it allows to build analysis, transformations and visualizations of Linked Data. ProLOD[Abe+14] automatically provides a group of statistical analysis regarding the content of a dataset, but it does not foresee any querying possibility.

The second group of tools are able to provide visual querying functionalities and advanced visualizations for the query results, but their focus is limited to the instance level. LD Query Wizard[Höf+14] allows to visualize an instance selected through keyword search and it uses a powerful tabular view that permit users to explore the neighborhood of this instance. LODlive and LODmilla[KB14] provide a visually appealing way to explore information associated with an instance using a graph visualization. Also, gFacet[HZL08; HEZ10] uses the same strategy of exploration (with a graph visualization), but in this case, each node is a class that contains a list of instances and the user can link new nodes (classes) as if he/she was building a visual query. SPARKLIS[Fer14] implements a fascinating approach in which a SPARQL query is composed as if the user was composing a natural language request to the dataset. ISPARQL[KBS07] allows to incrementally build a SPARQL query by extending it step by step; the main issues of this approach are that the user is required to have a good knowledge of the Semantic

[3]Among the variety of tools that handle Linked Data, I selected those able to connect to SPARQL endpoints. The comparison reported in table 4.1 is not intended to be exhaustive.

Web technologies and to understand the schema of the LOD source for defining a SPARQL query that retrieve interesting information.

As reported in [DR11], the majority of the tools for data visualization requires the user to manually explore the dataset and they are not able to provide a synthetic *schema* of the data contained in a single source. LODeX differs from the tools described above since it provides a synthetic representation of LOD source schema and the user can use it to build visual queries. However, LODeX has some limitations: it is not able to perform keyword queries, moreover there are large areas of improvement in the result visualization of the query.

4.3 Architectural Overview

LODeX consists of four distinct components, each responsible for a specific activity, named: (1) Indexes Extraction, (2) Schema Summary Generation, (3) Schema Summary Visualization, (4) Query Orchestration.

The components interact in order to: produce a visual Schema Summary (i.e. a high-level representation of the LOD source); provide it to the users; translate the visual query that a user might compose in a SPARQL query and to retrieve the results. The interaction is illustrated in Figure 4.1. For an easy reuse, all the contents extracted and processed are stored in MongoDB, a NoSQL document database (since it allows a flexible representation of the indexes).

4.3.1 Indexes Extraction

In a RDF graph the RDFS/OWL triples used to define a vocabulary or an ontology describe the intensional knowledge,while the instances and their datatype and object properties compose the extensional knowledge. In Figure 4.2 an example of the RDF graph representing a LOD source is displayed. The intensional knowledge is conveyed in the triples shown on the top of the figure, while, on the bottom, we have triples that describe three instances and compose the extensional knowledge.

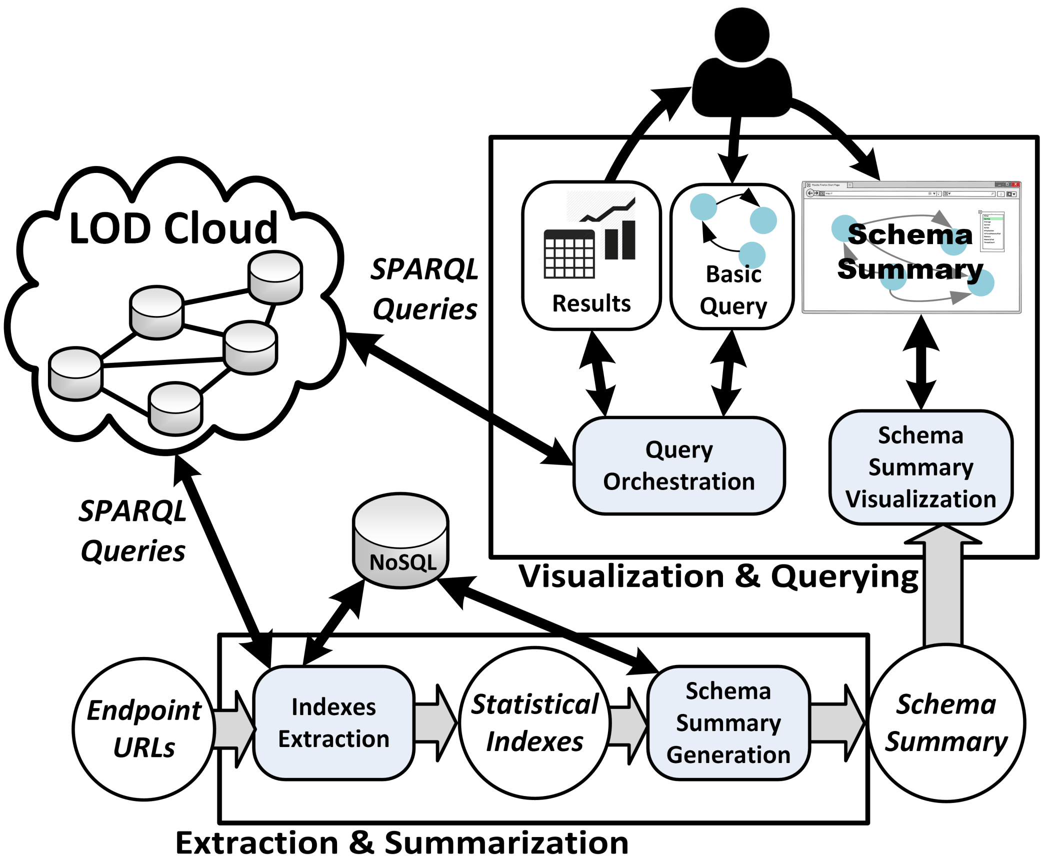


Figure 4.1: LODeX Architecture

The extraction process takes as input the URL of a SPARQL endpoint and generates a set of queries able to extract a set of indexes from the extensional knowledge (extensional group of Statistical Indexes in [BBP14b], see Chapter 3). These indexes are composed by sets of couple (c,p) where c is a class and p is a property:

- **SC** (Subject Class) contains object properties p and their domain class c.
- **SCl** (Subject Class to literal) contains datatype properties p and their domain class c.
- **OC** (Object Class) contains object property p and their range class c.

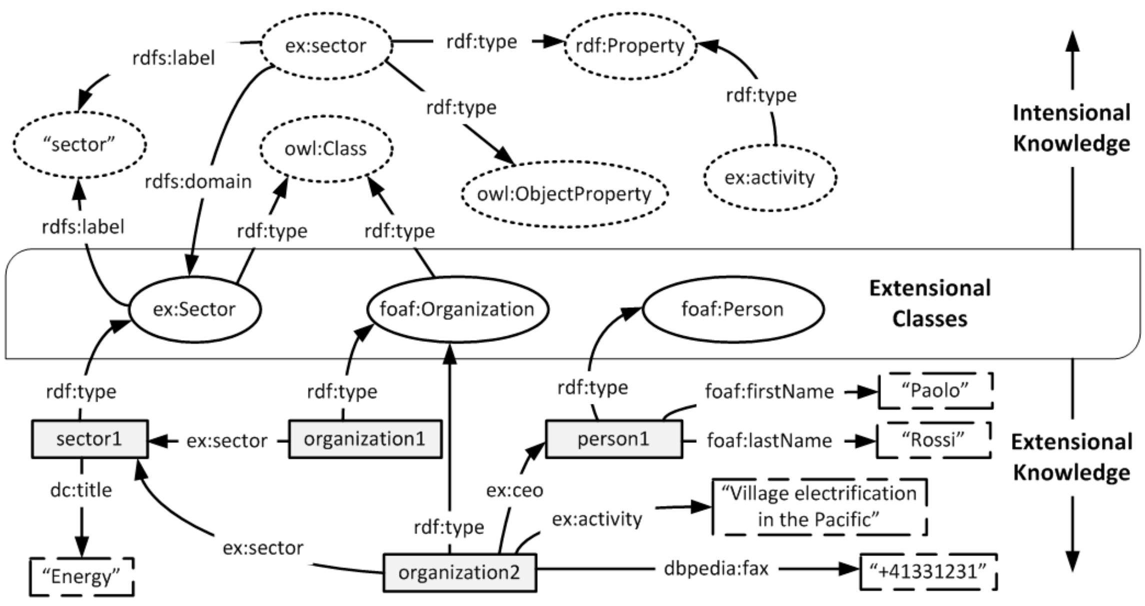


Figure 4.2: An example of the RDF Graph partitioning between intensional and extensional knowledge.

Table 4.2 lists the indexes extracted from the extensional knowledge of the example in Figure 4.2.

4.3.2 Schema Summary Generation

The Schema Summary (SS) of LOD source is created by exploiting information contained in the indexes described in the previous Section. The number of instances of each class and the number of times an index appear in a dataset are exploited in order to discover how the classes are connected in the extensional knowledge; thus, the SS is the schema of a dataset inferred from the distribution of the its instances (major details about the Schema Summary and the Schema Summary generation algorithm can be found in the Chapter 2).

In Figure 4.3 a representation of the SS of the previous example (shown in Figure 4.2) is depicted. The white circles represents classes (C), while the attributes (A) are shown in the gray boxes. Finally, the edges describe the properties (P). Each element is equipped with a numerical value representing the number of occurrences/number of instances.

Name	Values
Classes	{ ex:Sector, foaf:Person, foaf:Organization}
SC	{(foaf:Organization,ex:ceo), (foaf:Organization,ex:sector) }
SCl	{ (foaf:Person,foaf:firstName), (foaf:Person,foaf:lastName), (foaf:Organization,ex:dbpedia:fax), (ex:Sector,dc:title), (foaf:Organization,ex:activity), (foaf:Organization,dbpedia:fax) }
OC	{ (ex:Sector,ex:sector) }

Table 4.2: Classes and indexes extracted from the extensional knowledge of the source depicted in Figure 4.2

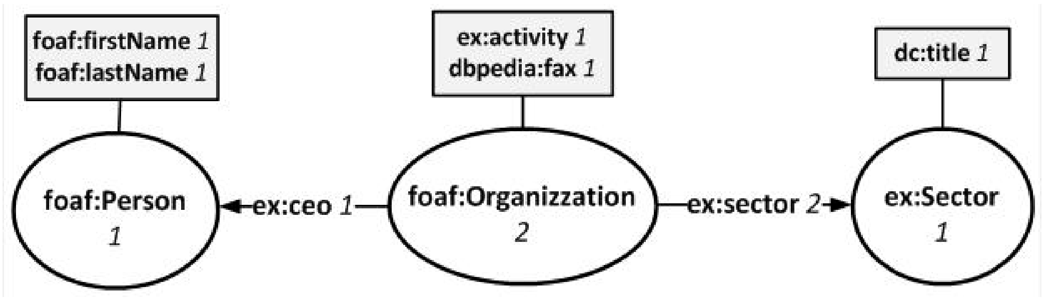
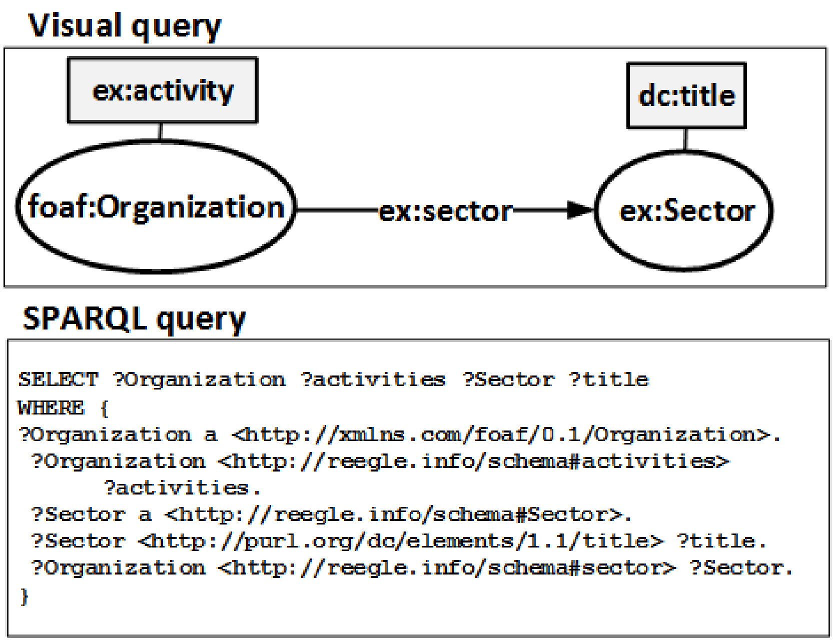


Figure 4.3: The SS of the LOD source represented in Figure 4.2.

In this context the usage of a model like the Schema Summary brings several advantages: the SS can be easily be stored and retrieved from MongoDB, storing the SS in a triplestore would have involved well known performance issues that would lead to worsening the performance of LODeX; the SS can be directly visualized in the GUI of LODeX and it makes possible the visual query building feature.

4.3.3 Schema Summary Visualization

The visualization is performed by a web application through which the user can interact for browsing the SS. The web server is implemented in Python, while



```
SELECT ?Organization ?activities ?Sector ?title
WHERE {
?Organization a <http://xmlns.com/foaf/0.1/Organization>.
 ?Organization <http://reegle.info/schema#activities>
       ?activities.
 ?Sector a <http://reegle.info/schema#Sector>.
 ?Sector <http://purl.org/dc/elements/1.1/title> ?title.
 ?Organization <http://reegle.info/schema#sector> ?Sector.
}
```

Figure 4.4: An example of a visual query created on the Schema Summary shown in Figure 4.3 and its translation in SPARQL.

the user interface uses different Javascript libraries to produce an interactive web application. In particular, I used Polymer to manage the GUI, a new library that allow to design applications according to the Material Design principles by using Web Components[4]. I used Data Driven Documents[5][BOH11] to create the interactive Schema Summary, and Sgvizler[6][Skj] to allow the querying of the remote endpoints and to display the results. The visualization of the Schema Summary has been also presented in the demo [BBP14a].

4.3.4 Query Orchestration

The Query Orchestrator manages the interaction between the GUI and the user in composing the visual query, in the generation of the SPARQL queries and in the submission of it to the remote endpoint.

[4] `http://www.w3.org/standards/techs/components`
[5] `http://d3js.org/`
[6] `http://dev.data2000.no/sgvizler/`

The classes, properties and attributes selected by the user in the visual query participate to the composition of a basic query Q. A basic query has a tree structure that overlaps the SS graph, the nodes of the tree are classes $\in C$, while the leafs can be both classes $\in C$ or attributes $\in A$. Here is its formal definition:

Definition 7 (Basic query) *A basic query Q, defined on a schema summary S, is a tuple Q= (T,m_C,m_E,m_A,o,F), where*

- *T is a directed and labeled tree composed by two kinds of vertices, T = (C',A',E'), where:*
  - *C' is a set of vertices where each vertex refers to a class in the Schema Summary*
  - *A' is a set of vertices where each vertex refers to an attribute in the Schema Summary. This kind of node can appear only as leaf of the tree T*
  - *E' is a set of edges where each edge consist of an ordered pair of vertices e=<n1,n2>, e $\in$ E', n1 $\in$ C', n2 $\in$ C' $\cup$ A'*
  - *r $\in$ C' is the root node,*
- $m_C\colon C' \rightarrow\ C$ *is a mapping function that links each vertex in C' with a vertex in C(S);*
- $m_A\colon A' \rightarrow\ A$ *is a mapping function that links each vertex in A' with an edge in A(S);*
- $m_E\colon E' \rightarrow\ P$ *is a mapping function that links each edge in E' with an edge in P(S);*
- $o\colon E' \cup A' \rightarrow\ \{true, false\}$ *is called optionality function.*
- *F contains the filtering conditions associated with the attributes of the query and it is composed by a set of tuple (a,k,v), where a is the attribute $\in$ A', k contains the filtering operator (a regular expression, or an arithmetic operator) and v is the value through which to assess the filter expression.*

Graphically, a user starts composing a basic query by selecting the first class in the SS, then, if the user selects a property for this first class, also the connected class is shown in the query panel and the edge and vertex are added to the tree. The user may also select the attributes of each class: in this case, the tree is further enriched with edges and leafs.

The Query Orchestrator translates the basic query into a SPARQL query through a compiler. The compiler exploits an iterative algorithm that traverses the basic query tree to produce the SPARQL query. The Query Orchestrator is able to compile non cyclic SPARQL query of any length; it allows the use of these SPARQL operators: AND (.), OPTIONAL (also nested), FILTER, ORDER BY, OFFSET and LIMIT. The pseudo-code of the query compiler is presented in Algorithm 3. For simplicity, I show only the compilation of the core of the query, then, this core will be equipped with the SELECT statement, the pagination condition (LIMIT and OFFSET) and the ORDER condition, if specified. Algorithm 1 uses as input the Schema Summary S, the basic query Q and the set of filter conditions F defined in the refinement panel. F contains triples like (n, k, v), where n is an attribute on which the filter is applied, k is an operator and v is a numerical value or a string. The output (SQ) of Algorithm 3 contains a SPARQL query. I use some of the algebraic operators defined in [PAG09b] to describe the formulation of the query: the triple pattern *(s,p,o)*, in which the elements can also be parameters (in this case we use the function Par for assigning a unique parameter); the operator *AND* that represents the dot (.) in SPARQL; the operator *OPT* (OPTIONAL); the operator *FILTER* (i.e. ?parameter condition value). Moreover, I use the term *attribute* to refer to an edge that connects a vertex that is a class, and a vertex that is a base type and the term *property* to refer to an edge that connects two vertices that are classes.

The example introduced in Figure 4.4 shows a simple query built on the Schema Summary of Figure 4.3. This query has been composed by selecting the class *foaf:Organization*, its attribute *ex:activity* and the property *ex:sector*. The selection of this property automatically results in the selection of the object class *ex:Sector*, then, I also add the attribute *dc:title*. From this graphical query, the

```
Data: S, Q, F
Result: SQ
Algorithm compiler()
    SQ=($Par$(r),rdf:type,$m_C$(r));
    SQ=attributes(r,SQ);
    SQ=recursivebody(r,SQ);
    return SQ;
Function recursivebody(c,RQ)
    forall the child ch of c do
        CHQ=($Par$(ch),rdf:type,$m_C$(ch)) AND
        ($Par$(c),$m_P$(<c,ch>),$Par$(ch));
        CHQ=attributes(ch,CHQ);
        if ch is not a leaf node then recursivebody(ch,CHQ);
        if o(ch) then RQ=RQ OPT ( CHQ );
        else RQ=RQ AND CHQ;
    end
    return RQ ;
Function attributes(c,SQ)
    forall the attribute a connected with c do
        AQ=($Par$(c),$m_A$(<c,a>),$Par$(a));
        if exist F(a) then AQ=AQ AND filtCon (a);
        if o(a) then SQ=SQ OPT ( AQ );
        else SQ=SQ AND AQ;
    end
    return SQ ;
Function filtCon(a)
    FC=$\emptyset$;
    forall the fc in F related to a do
        concatenate in FC, using the AND operator, each fc condition as:
        FILTER ($Par$(a), fc.k fc.v);
    end
    return FC ;
```

Algorithm 3: SPARQL compiler algorithm

Query Orchestrator module generates the SPARQL query shown in the bottom part of the Figure 4.4.

4.4 A Use Case Scenario

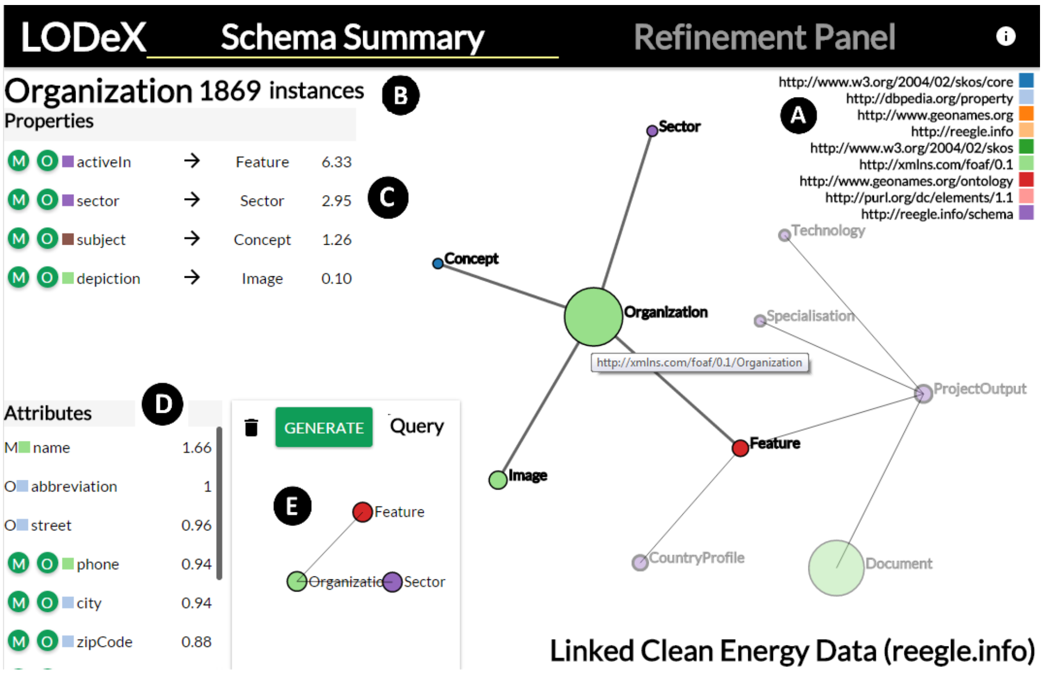
Here, I refer to a hypothetical use-case involving a company in the clean energy sector. The company has its own products and services and attempts to discover new information on renewable energy in the country where it is located. It is very likely that looking on portals like Datahub, the company detects the "Linked Clean Energy Data" dataset[7]. This dataset, composed of 60140 triples, is described as a "Comprehensive set of linked clean energy data". By using LODeX, the structure of the dataset is revealed and it can be easily browsed (see Figure 4.5)[8]. At a glance, the user can have the intuition of all the main classes (the nodes in the graph) and the connections among them (the arcs), besides the number of instances defined for each class (reflected in the dimension of the node). Focusing on the color of the nodes, a user can understand which classes are defined by the provider of the source and which others are taken from external vocabularies (in this case I can see that some of the class definitions are acquired from Foaf, Geonames.org and Skos) using the legend (Fig 4.5 Sect A). By positioning the mouse on a node, more information about the class is shown.

As depicted in Figure 4.5 (Sect B), the source collects 1869 organizations and each organization is described by some attributes (Sect D) together with the average number of times each attribute is used by an instance of the class, for example not all the instances have a zip code (0.88), whereas all of them have more than one name (1.60). Moreover, a class is linked to others by some properties (Sect C). By navigating the schema, a user might also discover that each organization is link to roughly 3 sectors, but then each sector (36 sectors in total) is linked to 151 organizations.

The user has to select a root node to start building a visual query ("Organi-

[7] `http://data.reegle.info/`

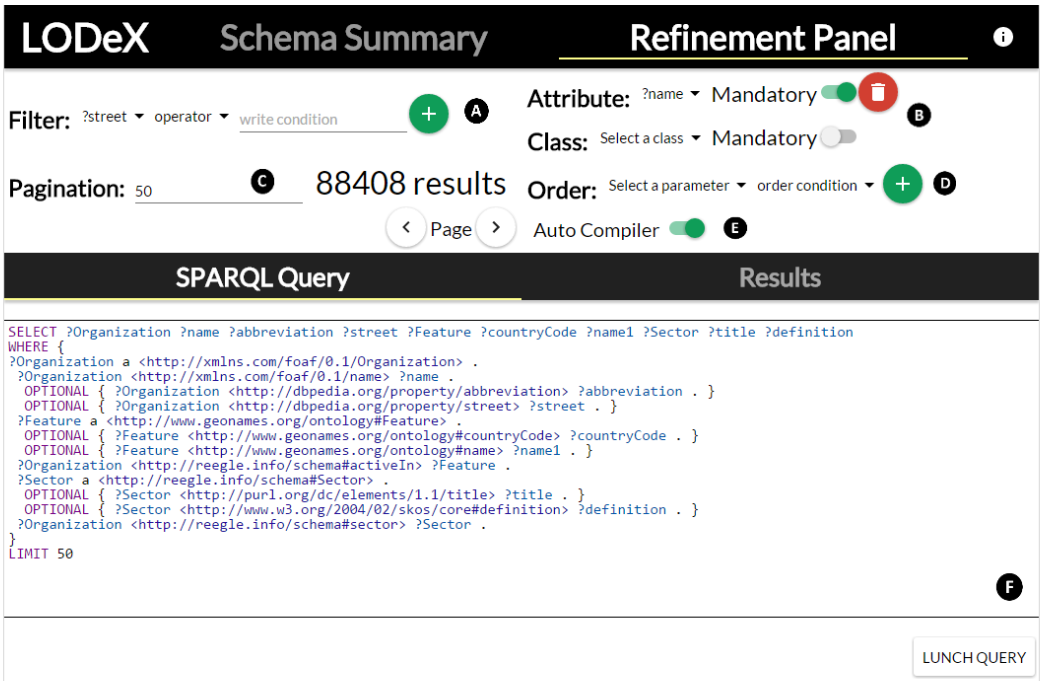
[8] The visual summary of this source is also available at `http://dbgroup.unimo.it/lodexCleanEnergy`


LODeX
Schema Summary
Refinement Panel
Organization 1869 instances
B
Properties
C
Attributes
D
GENERATE
Query
E
Feature
Organizatio
Sector
A
http://www.w3.org/2004/02/skos/core
http://dbpedia.org/property
http://www.geonames.org
http://reegle.info
http://www.w3.org/2004/02/skos
http://xmlns.com/foaf/0.1
http://www.geonames.org/ontology
http://purl.org/dc/elements/1.1
http://reegle.info/schema
Sector
Technology
Concept
Organization
Specialisation
http://xmlns.com/foaf/0.1/Organization
ProjectOutput
Feature
Image
CountryProfile
Document
Linked Clean Energy Data (reegle.info)

M O	activeIn	→	Feature	6.33
M O	sector	→	Sector	2.95
M O	subject	→	Concept	1.26
M O	depiction	→	Image	0.10

M	name	1.66
O	abbreviation	1
O	street	0.96
M O	phone	0.94
M O	city	0.94
M O	zipCode	0.88

Figure 4.5: An example of visual query on the "Linked Clean Green Energy Data" source



Filter: ?street operator write condition

Attribute: ?name Mandatory

Class: Select a class Mandatory

Pagination: 50

88408 results

Order: Select a parameter order condition

Auto Compiler

```
SELECT ?Organization ?name ?abbreviation ?street ?Feature ?countryCode ?name1 ?Sector ?title ?definition
WHERE {
?Organization a <http://xmlns.com/foaf/0.1/Organization> .
 ?Organization <http://xmlns.com/foaf/0.1/name> ?name .
  OPTIONAL { ?Organization <http://dbpedia.org/property/abbreviation> ?abbreviation . }
  OPTIONAL { ?Organization <http://dbpedia.org/property/street> ?street . }
 ?Feature a <http://www.geonames.org/ontology#Feature> .
  OPTIONAL { ?Feature <http://www.geonames.org/ontology#countryCode> ?countryCode . }
  OPTIONAL { ?Feature <http://www.geonames.org/ontology#name> ?name1 . }
 ?Organization <http://reegle.info/schema#activeIn> ?Feature .
 ?Sector a <http://reegle.info/schema#Sector> .
  OPTIONAL { ?Sector <http://purl.org/dc/elements/1.1/title> ?title . }
  OPTIONAL { ?Sector <http://www.w3.org/2004/02/skos/core#definition> ?definition . }
 ?Organization <http://reegle.info/schema#sector> ?Sector .
}
LIMIT 50
```

Figure 4.6: An example of the translation of the visual query of Figure 4.5 into the corresponding SPARQL query.

zation" in Figure 4.5). Now the user can add some attributes to the current class by clicking the buttons on the left of the attributes name (M: mandatory, O: optional). In Figure 4.5 the user select 3 optional attributes (”name", ”abbreviation" and ”street") for the class ”Organization". The user can also add other classes linked to the current class through a specific property by clicking the button on the left of a property in the property panel (Sect C). In Figure 4.5 the user added 2 mandatory classes/properties (”activeIn" ”Feature" and ”sector" "Sector"). The user can look at the visual query that he is building (Fig. 4.5 Sect E) and use it in order to focus on the different components of the query and add other attributes or properties/classes. At this point, the user can generate the query clicking the ”Generate" button which brings the user in the query refinement panel (Figure 4.6).

In the refinement panel (Fig. 4.6) the user can visualize the SPARQL query (F) that has been generated and he may manually modify it. He can also choose to visualize directly the result of the query by selecting the result tab or enable the automatic compiler (E) and modify the query by using the interface on the top (A,B,C,D) visualizing the results that change according to his refinement. After any change the query is compiled and automatically sent to the endpoint. In particular, the user can: (A) add or remove filter condition on the attributes contained in the query; (B) modify the optionality of attributes/classes or remove one of them from the query; (C) remove the pagination of the results, or modify the page size; (D) insert or remove ordering condition.

4.5 Evaluation

I propose three kinds of evaluations regarding LODeX: first, I analyze the portability of the LODeX approach; then, I evaluate the level of expressiveness of the SPARQL queries that can be generated by LODeX; finally, I provide the result of a usability evaluation performed with anonymous users. A deep evaluation of the performance of IE process can be found in [BBP14b] and in Chapter 3.

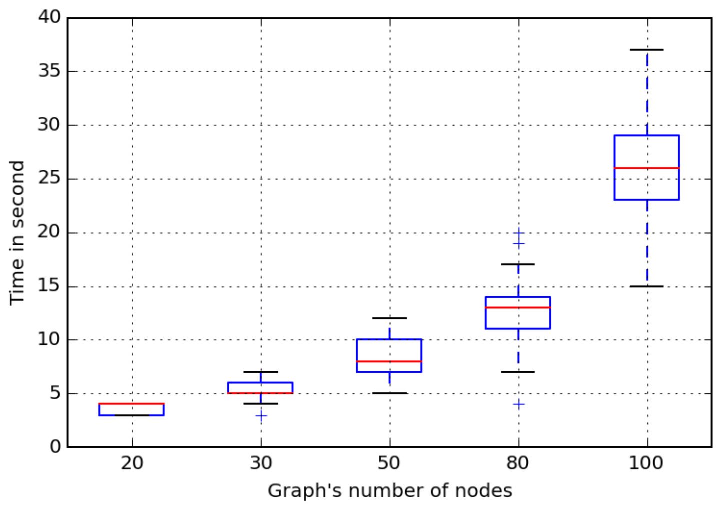


Figure 4.7: Distribution of the micro-tasks execution time grouped for graph size.

4.5.1 Portability to SPARQL endpoints

LODeX has been designed to be a tool able to work with each dataset provided of a SPARQL endpoint. Thus, I use the complete list of SPARQL endpoints contained in DataHub as test set.

Table 4.3 reports the number of datasets that were examined; 302 datasets were online when I performed the test. The IE process use a subset of SPARQL operator to extract the indexes, so, just 206 datasets were compatible. Another well known issue is the bad performance of some SPARQL endpoint, for this reason the number of endpoint for which I was able to generate the SS decrease to 185, that remains a good result because I obtained a SS from the 61% of the reachable endpoints.

Now, I am going to extend this portability evaluation to the GUI of LODeX for inspecting two aspects: success/failure of the communication with the endpoints; clarity of the graph representation of the SS[9].

[9]the results of report can be consulted online at `http://dbgroup.unimo.it/`

Test	**Nov 2014**
Reachable datasets	302
SPARQL 1.1 compatible	206
Extraction completed	185

Table 4.3: Number of Dataset used perform the portability evaluation

I executed preliminary usability test in my laboratory using 5 students to find out how many the size of the graph affects its clarity. I asked the students to individuate a specific node in graphs of different size (20, 30, 50, 80 and 100 nodes) and I measured the time taken for each task. I provided to students 25 tasks each (5 tasks for each graph size). The results are shown in the Figure 4.7, as you can see the finding time increases almost linearly when the dimension of the graph is less than 80 nodes. For this reason, I decided to not consider the datasets having more than 80 nodes. The number of these datasets is 40 and they represent the 21% of the total. Possible solutions to this issue will be discussed in as future work in Section 4.6.

Out of the remaining 145 endpoints, 7 were not online when the test was performed, 28 returned to the user interface a non-standard response. The LODeX web application makes an AJAX request to the endpoint containing the query and requiring a response encoded through *JSONP (JSON with padding)*. This kind of communication technique is used in Javascript to overcome the *same-origin policy*, commonly used to avoid cross-site scripting (XSS) attacks. Since some endpoints (28 in my case) were not able to encode a JSONP response, they replied with a non-standard response. Finally, 110 endpoints, almost the 60% of the total[10], successfully pass the test.

4.5.2 SPARQL expressiveness

To evaluate the level of expressiveness of the queries generated I inspected how many of the queries composing the BSBM benchmark[**bizer2008benchmarking**]

`detasetsLodexPortability.html`

[10]You can browse these datasets using the demo of LODeX available at: `http://dbgroup.unimo.it/lodex2`

(Berlin SPARQL Benchmark) could be generated using LODeX. These set of queries is formed by queries usually used to explore a new dataset. LODeX would be able to generate 6 of 10 queries proposed by the benchmark, a good result taking in consideration that a user without any knowledge of SPARQL could be able to generate them with LODeX. The four excluded queries contain SPARQL operator not supported by my tool: UNION, CONSTRUCT and DESCRIBE. LODeX is able to generate all the queries involving any type of JOIN and FILTER operation except for the cyclic queries. Indeed, the SPARQL compiler is able to automatically translate a basic query, the structure of which is a tree.

4.5.3 Usability Evaluation

This section summarizes the results of an evaluation performed as an online survey[11] compiled by anonymous users. Among the users involved, 22 were enrolled from IT communities and others 5 were bachelor students. I divided the survey in two distinct parts: the first aims to verify if the graph visualization of the SS is clear in representing the structure of a dataset; the second part intends to prove if the visual query panel is a powerful and adequate way for generating SPARQL queries. The survey collects the results of a sparse set of users aged between 23 and 43 years (Fig. 4.8) with different Semantic Web technologies skills (as shown in Figure 4.9). This is an ideal scenario to prove the effectiveness of the tool on users with different background knowledge.I used 3 different datasets in the survey: (D1) Bio2RDF - INOH - pathway database of model organisms[12]; (D2) Linked Open Aalto Data Service - Open data published by Aalto University[13]; (D3) Nobel Prizes - Linked Open Data about every Nobel Prize[14].

[11]The survey can be compiled at this url:`http://goo.gl/forms/FRSRWKLSq4`

[12]`http://datahub.io/dataset/bio2rdf-inoh`

[13]`http://datahub.io/dataset/linked-open-aalto-data-service`

[14]`http://datahub.io/dataset/nobelprizes`

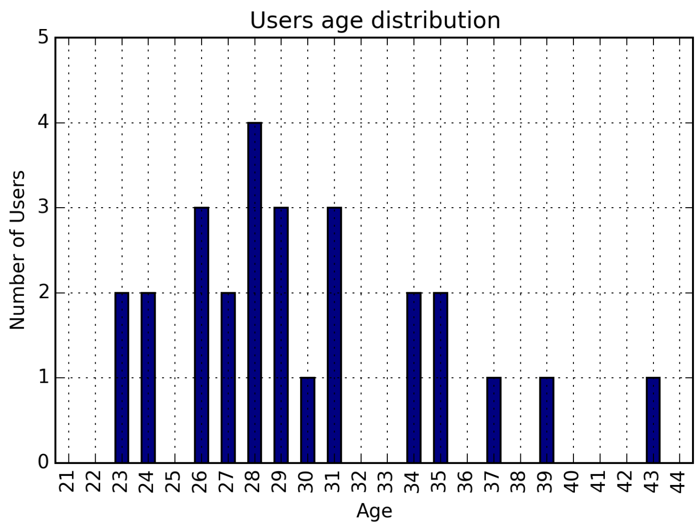


Figure 4.8: Age distribution.

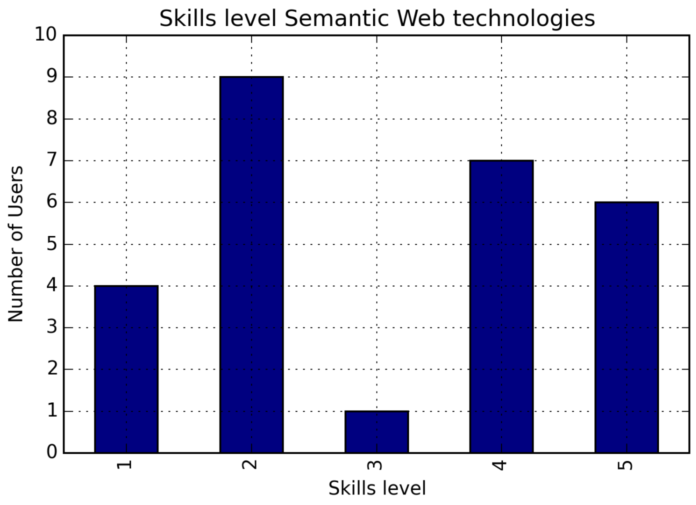


Figure 4.9: Semantic Web skill distribution.

Methodology

The survey encloses a short tutorial containing a description of the SS and a short video where the functionalities of query generation are explained[15]. Each of the two parts is composed by micro-tasks designed to evaluate the effectiveness of LODeX in addressing its two main goals.

Schema Summary Browsing Functionality - I propose two anonymous SS generated from two datasets (D1 and D2). The tasks that I asked the users to perform are listed in Table 4.4 (T1 to T4).

T1:	Find out the topic of each dataset	D1,D2
T2:	Find out the class with the largest number of instances	D1,D2
T3:	Find out the classes connected to a given class chosen by us	D2
T4:	Find out the most used attribute of a class chosen by us	D2
Q1:	Return all the different category of Nobel prizes	D3
Q2:	Return a table containing the list of winners of a Nobel prizes ordered by the name of the winner; the table has to contain the date of birth of the winner.	D3
Q3:	Find the award files related to the award of Peter W. Higgs	D3
Q4:	Find the organizations that won a Nobel prize after the 1999	D3

Table 4.4: Tasks and queries used in the LODeX evaluation and the corresponding datasets.

Query Generation Functionality - I asked to users to generate 4 different queries from natural language requests (the requests are listed in Table 4.4 from Q1 to Q4).

Finally, I asked to compile a SUS[Bro96] questionnaire and reply to a usability questionnaire. In particular, I asked to score, on a scale of 1-5, the following sen-

[15]The tutorial is accessible at `http://dbgroup.unimo.it/LODeXGuide.html`

tences: “I found the Schema Summary was easy to browse””; It permits to have an overview about the structure of a Dataset”; “The visualization of the Schema Summary is clear”. For the second part, I asked questions regarding the SPARQL query generation feature and the overall tool: "How do you evaluate your knowledge about SPARQL?"; "If you have already written SPARQL queries, how do you find using LODeX compared to manually writing SPARQL queries?"; "Any comments? What was good / bad / unexpected / difficult?".

```
SELECT ?Person ?name ?Award  ?AwardFile ?label
WHERE {
?Person a <http://xmlns.com/foaf/0.1/Person> .
 ?Person <http://xmlns.com/foaf/0.1/name> ?name .
 ?Award a <http://dbpedia.org/ontology/Award> .
 ?Award <http://data.nobelprize.org/terms/laureate> ?Person .
 ?AwardFile a <http://data.nobelprize.org/terms/AwardFile> .
 ?AwardFile <http://www.w3.org/2000/01/rdf-schema#label> ?label .
 ?Award <http://data.nobelprize.org/terms/awardFile> ?AwardFile .
 FILTER  (str(?name) = "Peter W. Higgs" ).
}
```

Listing 4.1: Query generated answering to Q3

Quantitative evaluation

I evaluate the correctness of the answers provided by users for the tasks listed in Table 4.4.

Schema Summary Browsing Functionality - The tasks belonging to this section obtain an accuracy of the 91% (Table 4.5). I asked to complete these tasks without querying the dataset, but just browsing the Schema Summary, so I obtained a high accuracy. The students that completed the survey in my laboratory were able to complete these task in less than 5 minute in average.

Query Generation Functionality - These group of tasks obtained an overall accuracy of 90%. This is a very good result because the last 3 queries are quite complex; in fact, they involve 2 or more classes with a filter or an order condition (see the correct query generated for the request Q3 is listed in Listing 4.1) . The students that completed the survey in my laboratory were able to complete these

Task	Number	n Correct	% Correct
T1	54	48	89%
T2	54	48	89%
T3	27	23	89%
T4	27	27	100%
Total	**162**	**148**	**91%**

Table 4.5: Results of the tasks listed in Table 4.4

Task	Number	n Correct	% Correct
Q1	27	27	100%
Q2	27	26	96%
Q3	27	22	81%
Q4	27	23	85%
Total	**108**	**98**	**90%**

Table 4.6: Results of the queries listed in Table 4.4

task in less than 15 minute in average. This is a very promising result, in fact, all the students enrolled had a very low knowledge of SPARQL.

Qualitative evaluation

I evaluate the SUS score obtained and the answers to the qualitative question proposed.

Schema Summary Browsing Functionality - In Figure 4.13 you can have an overview of the SUS score obtained by the 27 users, the results are clustered according to the knowledge of the user of the Semantic Web technologies. The SUS overall median value is 85 and, according to [BKM09], I can classify this functionality to *Excellent*. The median values obtained distinguishing among skilled and unskilled user are rather similar (82.5 vs 87.5), so I can assume that this functionality has been appreciated by both kind of users. Moreover, I also request to rank the level of agreement to three sentences regarding the SS:

- "I found that the Schema Summary was easy to browse" (see results in Figure 4.10).

- “It (SS) permits to have an overview about the structure of a dataset” (see results in Figure 4.11).

- “The visualization of the Schema Summary is clear” (see results in Figure 4.12).

Practically, most users think that: it is easy to browse; it can work as documentation of a dataset; its visualization is clear.

Query generation functionality - This functionality uses all the features of the tool, so I can assume that the SUS scores obtained in this step represent the global SUS score of LODeX. Therefore, the distribution of the SUS score obtained for LODeX is shown in Figure 4.14 and I obtained a median SUS score of 82.5 that classifies LODeX as *Excellent*, always according to [BKM09]. Also in this case, I do not find particular differences among the median value of the score among skilled and unskilled users (82.5 vs 85). The fact that, both skilled and unskilled users equally appreciated LODeX, according to the SUS scores, demonstrates that the final user can be unaware to Semantic Web technologies to explore and query LOD sources with LODeX. That was one of the main goal of LODeX in order to increase the usage of LOD sources. Users who did not know SPARQL were able to query a dataset LOD for the first time; an user answers to this question, "If you have already written SPARQL queries, how do you find using LODeX 2.0 compared to manually writing SPARQL queries?", like this: "Just written my first SPARQL queries using LODeX. Nice". On the other hand, one skilled user answers to the question above in this way: "LODeX is cognitively less demanding". I received also some criticisms concerning some aspects of the GUI (e.g. browser rendering differences) that will be very useful for improving LODeX. Another criticism regards the graph visualization of the SS that can became complex for huge dataset and starting a query can be difficult for a user.

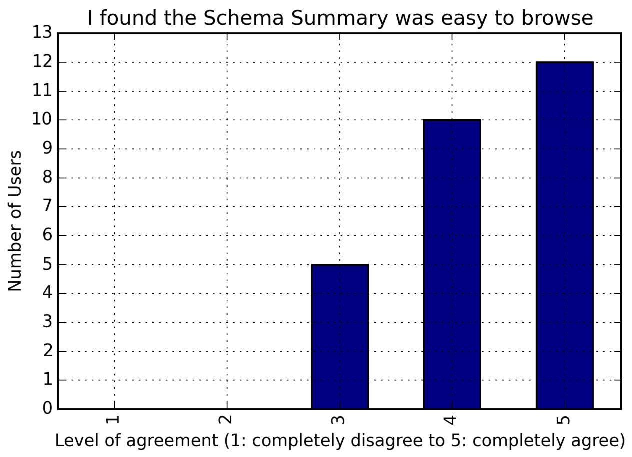


Figure 4.10: Distribution of agreement rate of users to the sentence: “I found that the Schema Summary was easy to browse”.

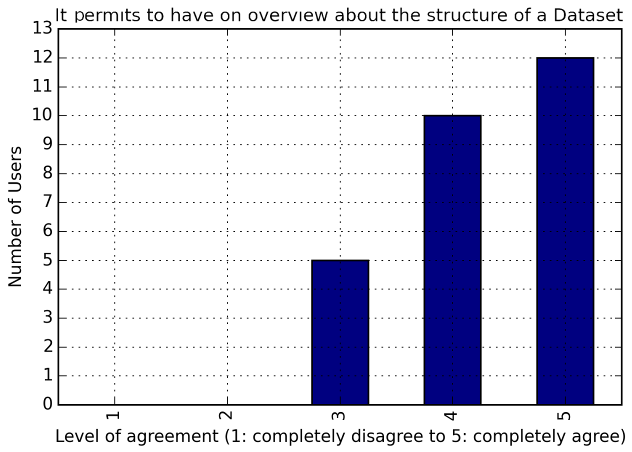


Figure 4.11: Distribution of agreement rate of users to the sentence: “It (SS) permits to have an overview about the structure of a dataset”.

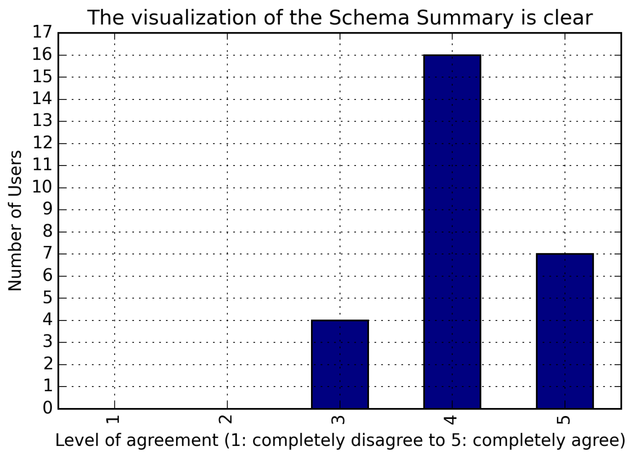


Figure 4.12: Distribution of agreement rate of users to the sentence: “The visualization of the Schema Summary is clear”.

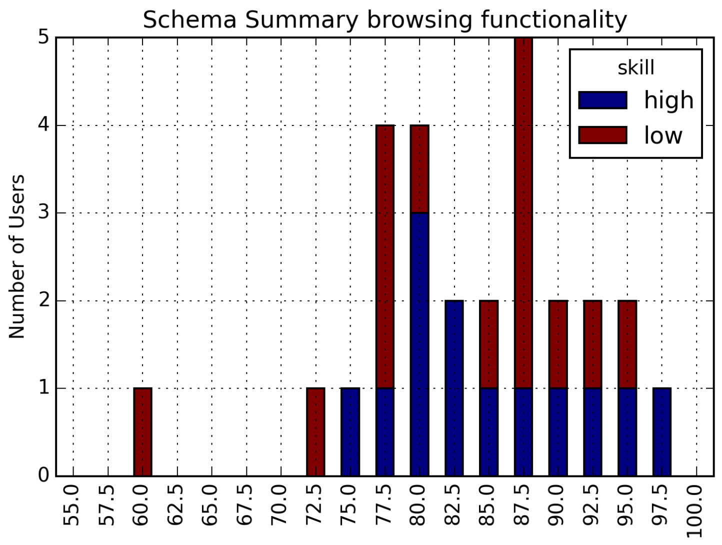


Figure 4.13: Distribution of SUS score for the Schema Summary browsing.

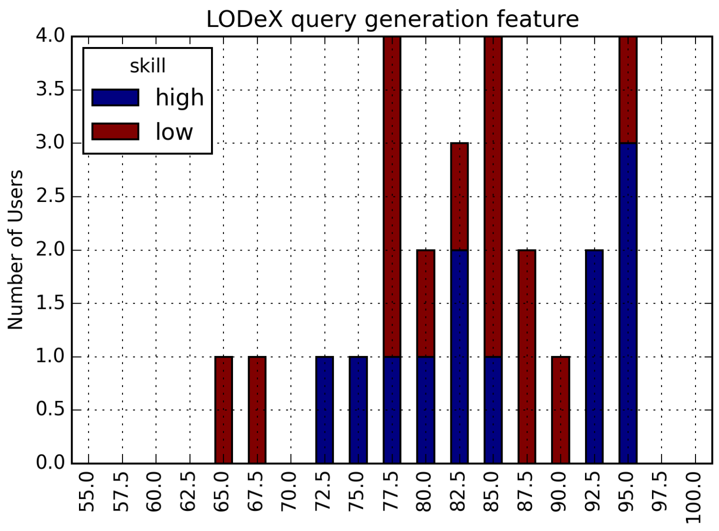


Figure 4.14: Distribution of SUS score for the query generation functionality.

4.6 Conclusion & Future Works

In this Chapter, I presented LODeX, a tool for visual exploration and querying of LOD sources. LODeX unveils the intrinsic structure of a LOD source by providing a summarized view of the dataset and allow users to visually compose/refine a query addressed to this source.

LODeX has proven to be an effective tool in facilitating users' interaction with LOD sources. Moreover, writing SPARQL queries can be a time-consuming and boring task also for experts, thus, navigating the inferred schema of a dataset and selecting classes and attributes of interest can strongly simplify the formulation of a query, making more pleasant the consumption of Linked Data. Portability tests showed that LODeX is able to process 61% of the accessible SPARQL endpoints and to render 59% of the LOD sources.The survey, conducted on 27 users, has revealed a good level of usability with a SUS classification as "Excellent". A complete demo of the tool has been also presented in the demo [BBP15a]

However, some limitations arise from the evaluation of the tool. First of all the graph visualization of the SS can become messy for huge dataset. This might affects the portability of LODeX, therefore I are currently studying different solutions to solve this drawback: for example to apply clustering techniques and group together some sets of nodes with similar characteristics or limit the number of nodes visualized to the neighborhood of the node that is the current focus of the user. The first solution allows to visualize the structure of the whole dataset, but the query building functionality might be affected. With the second option, I do not affect the query building, but I lose the possibility to represent the whole dataset. Moreover, the use of keyword search techniques could significantly improve the selection of elements of a visual query.

Chapter 5

Clustered Schema Summary

5.1 Introduction

The RDF Data Model plays a key role in the birth and continuous expansion of the Web of data since it allows to represent structured and semi-structured data. However, while the LOD cloud is still growing, we assist to a lack of techniques able to produce a meaningful, high level representation of these datasets. In Chapter 2 I presented a model, called Schema Summary, able to represent the structure of a generic LOD dataset; In this Chapter, I contribute to the LOD summarization process by defining a clustering procedure to further shrink the representative schema, crucial especially for portray huge datasets. The Schema Summary (SS) and the Clustered Schema Summary (CSS) enable summarization at different granularities. The SS conveys the implicit structure of the LOD dataset, by displaying the main classes and the properties used among them. While, the CSS provide a further "contraction" of the SS by gathering together classes which concur in the instantiation of same instances and computing the central class that best identifies each group. In this Chapter I will also evaluate both SS and CSS respect to ontology summarization metrics.

The remainder of this paper is organized as follows. Section 5.2 discusses related work. The model and the generation algorithms are presented in Section 5.3.1. Here, I formally define the Clustered Schema Summary and I present the generation algorithms able to generate an RDFS light-weight ontology that conveys the information of the SS/CSS. The evaluation of my methodology is reported in section 5.4. This section consists of some statistics on the datasets used for the tests, the definition of the employed evaluation measures of coverage and compression. Conclusions are sketched in Section 5.5.

5.2 Related Work

Several techniques of schema summarization has been applied in the last few years to different data models with the purpose of increase the usability and the comprehension of the dataset where they are applied. With the rise of publications of LOD, and Semantic Web in general, a lot of effort has been spent in applying these technique to increase the usability of RDF data.

The most important group of techniques that can be applied to LOD sources are collected under the name of Ontology Summarization. In this group, we can find a lot of works ([PMd08], [LMd10] [ZCQ07] [Wu+08]) that usually produce as output a ranked list of the most important concepts identified in the ontology. The main drawbacks of these techniques are: their summaries do not represent the structure of the source; they were applied to small OWL ontology containing just intensional content. Differently from these ontologies, the datasets belonging to the LOD cloud have a more heterogeneous content. Recently, in [Zha+09] and [Che+12], summarization techniques have been applied on vocabularies coming from the LOD cloud. The main limitation of these works is that they always rely on a schema available in RDFS format, while several datasets, lacking of intensional knowledge, remain excluded. The main difference of my method is that the summary produced takes into account the extensional content, so it can be applied on every dataset belonging to the LOD cloud.

My technique can also be compared to more general techniques working on RDF data in which a summary, or a set of descriptors, are extracted from a LOD source. In [CPF13], authors divide these techniques in two groups, *triples-level* and *instances-level* summaries, according to the granularity with which the sources are scanned and indexed.

The triples-level techniques inspect the content of the RDF dataset scanning each triple, and then usually build an index containing statistical information regarding the type of these triples. SchemEx [Kon+12] is an example of work belonging to this group; here, dumps of RDF graphs are indexed to support user query processing. Differently from my approach, this work does not consider the connection among instances, thus it is only able to give a complete characterization of classes belonging to an RDF graph.

In the instance-level approach the RDF graph is inspected taking into account RDF, RDFS and OWL primitives and their semantics, in order to detect structural information pervasive in the source. We can find two important works [Har+10] [PKK12] where the instance-level summary can support efficient and federated query evaluation. My technique can also be positioned in this group because the sources are inspected looking for the existing connection among instances of classes to infer the underlay structure of a dataset, but the main purpose is to build a summary describing the structure of the source.

5.3 Model definition

In this Section, I present the formal definition of the Clustered Schema Summary. This definition is based on the Schema Summary definition that can be found in Chapter 2, Section 2.3.

5.3.1 Clustered Schema Summary

The Clustered Schema Summary leverages multiple class instantiation which is a structural feature common to a lot RDF graphs. Multiple class instantiation (i.e. the declaration of instances as members of more than one class) is a common practice in knowledge bases, since it could offer a modeling solution for most situation were an instance may be consider as being member of two or more classes. An arbitrary number of *rdf:type* properties can be associated with a resource, so, a node of the RDF graph can be, at the same time, instance of more than a class. I call a set of classes that concur in the multiple instantiation of the same resource $i \in (U \cup B)$ *partial cluster of classes* ($PC(i)$). Each $PC(i) \subseteq C$ (C is the set of classes used in the RDF graph (RG)).

Definition 8 $PC(i) = \{c | \text{<}i,rdf{:}type,c\text{>} \in RG \wedge i \in (U \cup B)\}$

By examining all the instances in a RG graph, we can find different PC. The collection of all the PC that occur in a RG graph is called *family of PC* ($\mathfrak{C}$):

Definition 9 $\mathfrak{C} = \{PC(i) : \forall i \in (B \cup U)\}$

In $\mathfrak{C}$ is contained a particular family of sets able to generate all the other sets. I call this family, *family of super sets* $\mathfrak{S}$, and I define it as follow:

Definition 10 $\mathfrak{S} = \{SS \in \mathfrak{C} : \nexists PC \in \mathfrak{C} \wedge PC \supset SS\}$

I defined an algorithm able to extract $\mathfrak{S}$ starting from $\mathfrak{C}$. This algorithm extracts $\mathfrak{S}$ in an efficient way by taking advantage of its definition (Definition 10). The pseudo-code of the algorithm is presented below.

```
Data: $\mathfrak{C}$
Result: $\mathfrak{S}$
$\mathfrak{S} = \emptyset$;
while $\mathfrak{C} \neq \emptyset$ do
    c = set of max length $\in \mathfrak{C}$;
    add c to $\mathfrak{S}$;
    $\mathfrak{C}$ = $\mathfrak{C}$ - $powerset(c)$;
end
```

Algorithm 4: Super sets extraction algorithm

For each set $ss \in \mathfrak{S}$, a class $ca \in ss$ must be elected to represent the entire set of classes. This class is called *candidate agent of the superset* $\mathfrak{S}$. For each superset, I choose as candidate agent the class with the highest number of instances.

The set of all the candidate agents is called CA. The function $ca : CA \rightarrow \mathfrak{S}$ assigns to each candidate agent the corresponding super set. Now that some classes have been grouped together, I can produce a new Schema Summary with a reduced number of classes, that I called Clustered Schema Summary (CSS). The formal definition of the CSS is given in the following.

Definition 11 (Clustered Schema Summary) *A Clustered Schema Summary CS for a RDF dataset, derived from the Schema Summary S = <Cs, P, s, o, A, m, Σ_l, l, count>, is a pseudograph: CS = <Cs', P, s, o, A, m, Σ_l, l, count, $\mathfrak{S}$, ca>, where*

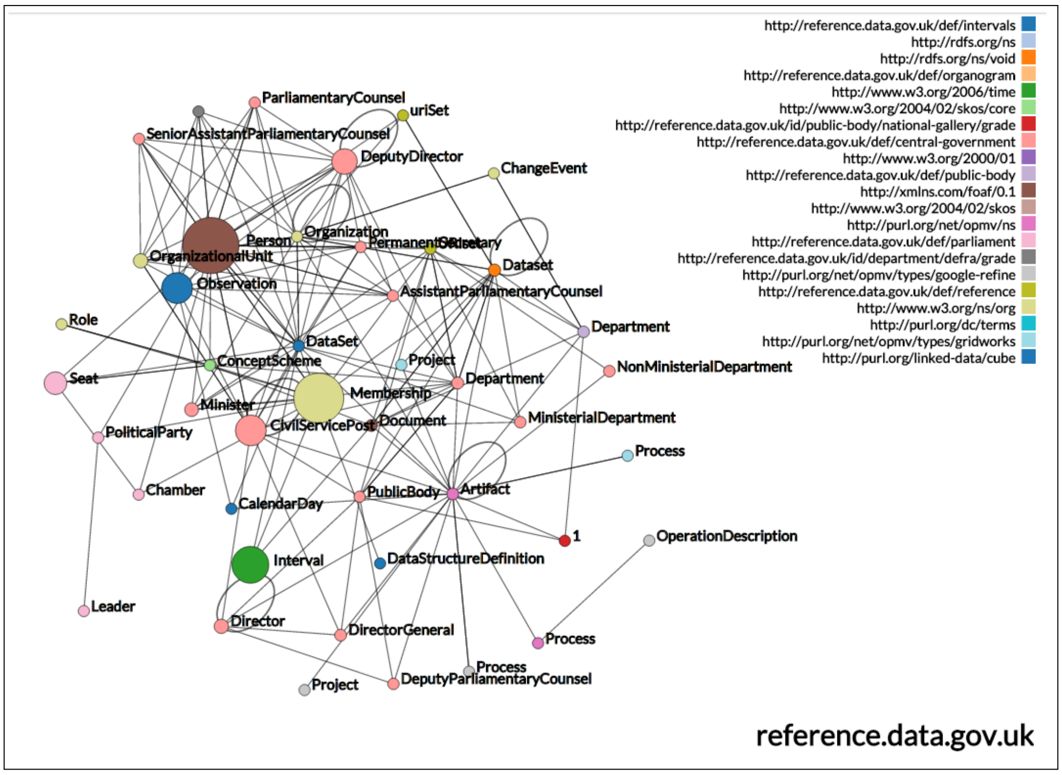
- *P, s, o, A, m,Σ_l, l, count are the same elements defined in the Schema Summary S;*
- *Cs' contains the classes represented in the CSS,* $Cs' = Cs - \{sc|sc \in SC : \forall SC \in \mathfrak{S}\} + CA$
- $\mathfrak{S}$ *is the family of superset;*
- $ca : CA \to \mathfrak{S}$ *is the candidate agent assignment function.*

5.3.2 RDFS light-weight ontology generation

I deem that a good summarization algorithm may produce an output that is compatible and comparable with the input. For this reason, I provide an algorithm able to translate the SS/CSS in an RDFS ontology[BG04] that embodies the structure of the RDF dataset.

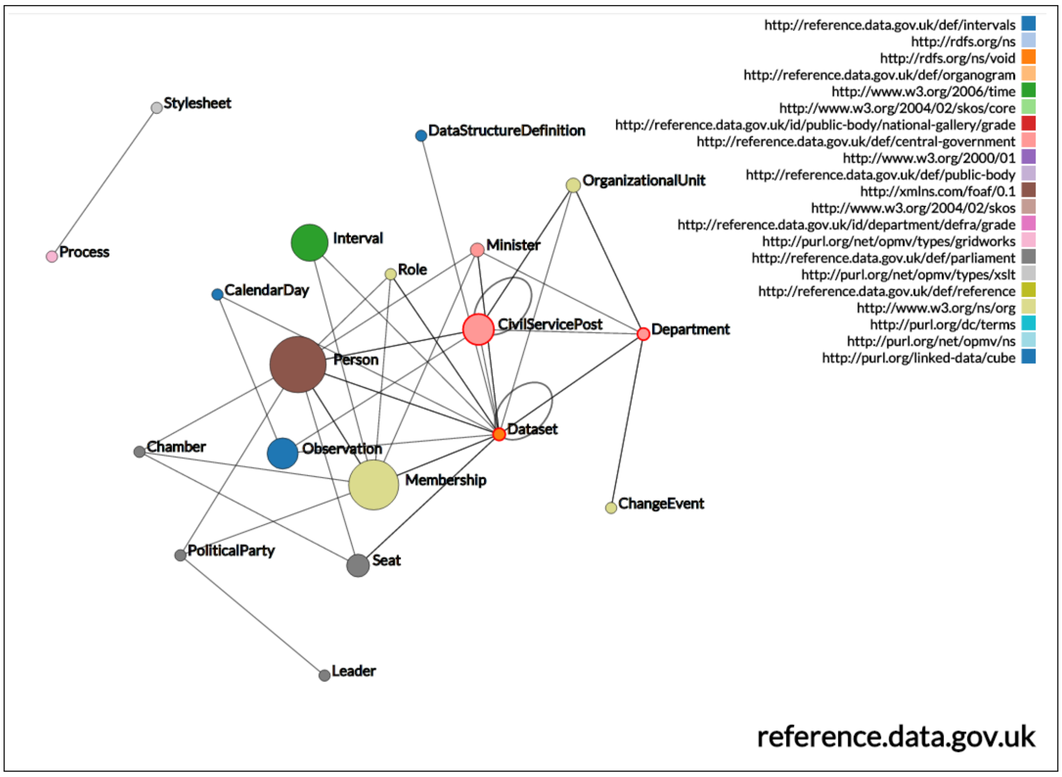
This translation is not completely lossless; by using the RDFS primitives only, it is not possible to exhibit all the information contained in the SS/CSS. In particular, I lose the number of occurrences of attributes, properties and instances (the function *count(e)*). However, an RDFS ontology describing the structure of the RDF Graph can be very useful and it can be portable to other applications.

The translation algorithm (see Algorithm 5) uses 4 primitives taken from the RDFS standard to generate the ontology: *rdfs:Class* used to define the classes; *rdfs:range*, *rdfs:domain*, *rdfs:Literal* used to define the properties and the attributes of the SS/CSS.



http://reference.data.gov.uk/def/intervals
http://rdfs.org/ns
http://rdfs.org/ns/void
http://reference.data.gov.uk/def/organogram
http://www.w3.org/2006/time
http://www.w3.org/2004/02/skos/core
http://reference.data.gov.uk/id/public-body/national-gallery/grade
http://reference.data.gov.uk/def/central-government
http://www.w3.org/2000/01
http://reference.data.gov.uk/def/public-body
http://xmlns.com/foaf/0.1
http://www.w3.org/2004/02/skos
http://purl.org/net/opmv/ns
http://reference.data.gov.uk/def/parliament
http://reference.data.gov.uk/id/department/defra/grade
http://purl.org/net/opmv/types/google-refine
http://reference.data.gov.uk/def/reference
http://www.w3.org/ns/org
http://purl.org/dc/terms
http://purl.org/net/opmv/types/gridworks
http://purl.org/linked-data/cube
ParliamentaryCounsel
uriSet
SeniorAssistantParliamentaryCounsel
DeputyDirector
ChangeEvent
Person
Organization
OrganizationalUnit
PermanentSecretary
Dataset
Observation
AssistantParliamentaryCounsel
Role
DataSet
Department
ConceptScheme
Project
NonMinisterialDepartment
Seat
Membership
Department
Minister
CivilServicePost
Document
MinisterialDepartment
PoliticalParty
Process
Chamber
CalendarDay
PublicBody
Artifact
1
OperationDescription
Interval
DataStructureDefinition
Leader
Director
DirectorGeneral
Process
Process
Project
DeputyParliamentaryCounsel
reference.data.gov.uk

Figure 5.1: Representation using LODeX of the SS generated from the "Reference data for linked UK government" dataset.



http://reference.data.gov.uk/def/intervals
http://rdfs.org/ns
http://rdfs.org/ns/void
http://reference.data.gov.uk/def/organogram
http://www.w3.org/2006/time
http://www.w3.org/2004/02/skos/core
http://reference.data.gov.uk/id/public-body/national-gallery/grade
http://reference.data.gov.uk/def/central-government
http://www.w3.org/2000/01
http://reference.data.gov.uk/def/public-body
http://xmlns.com/foaf/0.1
http://www.w3.org/2004/02/skos
http://reference.data.gov.uk/id/department/defra/grade
http://purl.org/net/opmv/types/gridworks
http://reference.data.gov.uk/def/parliament
http://purl.org/net/opmv/types/xslt
http://reference.data.gov.uk/def/reference
http://www.w3.org/ns/org
http://purl.org/dc/terms
http://purl.org/net/opmv/ns
http://purl.org/linked-data/cube
Stylesheet
DataStructureDefinition
OrganizationalUnit
Process
Interval
Minister
Role
CalendarDay
CivilServicePost
Department
Person
Dataset
Chamber
Observation
Membership
ChangeEvent
PoliticalParty
Seat
Leader
reference.data.gov.uk

Figure 5.2: Representation using LODeX of the CSS generated from the "Reference data for linked UK government" dataset.

```
Data: $S$
Result: RDFS(RDFS graph)
$RDFS = \emptyset$;
forall the $p \in S.p$ do
    RDFS.add(<S.l(S.s(p)),rdf:type,rdfs:Class>);
    RDFS.add(<S.l(S.t(p)),rdf:type,rdfs:Class>);
    RDFS.add(<S.l(p),rdfs:domain,S.l(S.s(p))>);
    RDFS.add(<S.l(p),rdfs:range,S.l(S.t(p))>);
end
forall the $a \in S.a$ do
    RDFS.add(<S.l(a),rdfs:domain,S.l(S.m(a))>);
    RDFS.add(<S.l(a),rdfs:range,rdfs:Literal>);
end
```

Algorithm 5: RDFS ontology generation algorithm.

5.3.3 Example

Here, I introduce an example showing the SS and CSS built on the reference.data.gov.uk[1] dataset, i.e. a source that contains reference data for linked UK government data. The RDF dataset is composed by 59.447 triples, 9.482 instances and 50 classes. I visualize the SS[2] (see Figure 5.1) and the CSS[3] (see Figure 5.2) visualized through LODeX. I also provide the RDFS ontology generated from the SS[4] and the CSS[5]. The SS is composed by 44 nodes while the CSS is composed by 20 nodes of which 4 are candidate classes that represent a super set of classes. In this example the CSS contains less than the 50% of the SS nodes. Table 5.1 lists the clusters that have been automatically generated; it is possible to see that the class selected as candidate agent, in each cluster, correctly represents the whole set of classes. This is even more evident in the last two clusters, where the candidate agents (*CivilServicePost* and *Department*) represent a generalization of the classes contained in the set.

[1]`http://datahub.io/dataset/reference-data-gov-uk`

[2]Please use Chrome to access to this url: `http://dbgroup.unimo.it/lodex2/testCluster#!/schemaSummary/328`.

[3]Please use Chrome to access to this url: `http://dbgroup.unimo.it/lodex2/testCluster#!/cSchemaSummary/328`

[4]`http://dbgroup.unimo.it/lodex_328.rdf`

[5]`http://dbgroup.unimo.it/lodex_328c.rdf`

Table 5.1: Clusters of classes in the "Reference data for linked UK government" dataset (*uk.gov* stands for http://reference.data.gov.uk/def/)

http://purl.org/net/opmv/ns#Process
http://purl.org/net/opmv/types/google-refine#Process

http://rdfs.org/ns/void#Dataset
http://purl.org/linked-data/cube#DataSet
http://purl.org/net/opmv/ns#Artifact
uk.gov:reference/URIset
uk.gov:reference/uriSet
http://www.w3.org/2004/02/skos/core#ConceptScheme
http://xmlns.com/foaf/0.1/Document

***uk.gov*:central-government/CivilServicePost**
uk.gov:central-government/AssistantParliamentaryCounsel
uk.gov:central-government/DeputyDirector
uk.gov:central-government/DeputyParliamentaryCounsel
uk.gov:central-government/Director
uk.gov:central-government/DirectorGeneral
uk.gov:central-government/ParliamentaryCounsel
uk.gov:central-government/PermanentSecretary
uk.gov:central-government/SeniorAssistantParliamentaryCounsel
http://reference.data.gov.uk/id/public-body/national-gallery/grade/1

***uk.gov*:central-government/Department**
uk.gov:central-government/MinisterialDepartment
uk.gov:central-government/NonMinisterialDepartment
uk.gov:central-government/PublicBody
uk.gov:public-body/Department
http://www.w3.org/ns/org#Organization

5.4 Evaluation

The evaluation of ontology summarization techniques is a quite controversial topic in literature. In fact, these techniques are usually designed to summarize an ontology with a purpose, so their evaluation is focused on the achievement of this goal. Nevertheless, Li and Motta in [LM10] tried to provide a systematic view of the different evaluation measures.

I chose two of the metrics prosed in [LM10] to estimate the quality of the summarized ontology I built: corpus coverage ontology evaluation, that score the ontology appropriateness to cover the topic of the corpus, and application-driven ontology evaluation, that measure the performance of an application that uses the summary.

In the following, after a brief description of the datasets used, I delineate the two metrics.

5.4.1 Datasets

The datasets used to evaluate my methodology are taken from DataHub[6]. For the SPARQL endpoint listed in DataHub, I make use of the Index Extraction module[BBP] of LODeX[BBP15b] to extract the patterns needed to generate the SS/CSS and the corresponding RDFS ontologies.

Table 5.2: Statistics on the tests performed on the LOD datasets listed on DataHub.

Test	Value
LOD datasets	599
Reachable datasets	302
SPARQL 1.1 compatible datasets	206
SS generated	185
CSS generated	90

Table 5.2 contains information about the test set. My method has been proven to work correctly with almost all the RDF graphs that I can index. Since the

[6]http://datahub.io/

patterns are extracted through SPARQL queries, this process greatly suffers on the bad performances of the endpoints, thus the tests performed on February 2015 revealed that only 185 out of 302 datasets can be indexed (major details of this issue can be found in [BBP]). I was able to built the SS on 185 datasets and the CSS on 90 datasets that contain clusters of classes [7].

5.4.2 Corpus coverage ontology evaluation

The concept of *corpus coverage ontology evaluation* has been solely defined in [LM10], but, to the best of my knowledge, do not exist cases in the ontology summarization field in which this kind of evaluation has been used. That is primarily due to the fact that in this field most efforts have been carried out into ranking and selecting key terms from a vocabulary as a summary [Che+12][LMd10] [PMd08][Wu+08][ZLQ06].

The SS supplies a synthetic representation of the extensional knowledge, so that we can think to the concept of *coverage* as: what percentage of the instances in the source RDF graph (*corpus*) are represented though the SS/CSS. Moreover, the SS can be translated into an RDFS ontology. In that way, both the source and the output are expressed using the same language (RDF) and I can compare their sizes defining a *compression* metric.

Coverage

To evaluate the coverage of SS and CSS statistical information stored in them are crucial. Indeed, I can know how many time a particular pattern appears in the source graph. The number of triples that are represented though the SS/CSS, n_{repr}, is approximatively computed by summing the number of occurrences of each property, each attribute and the number of class instances that appear in the SS/CSS. The ratio of n_{repr} to the total number of triples of the RDF graph, n_{tot}, give us the coverage:

[7]A demo of the SS and CSS extracted is available here (please use Chrome) :`http://dbgroup.unimo.it/lodex2/testCluster`

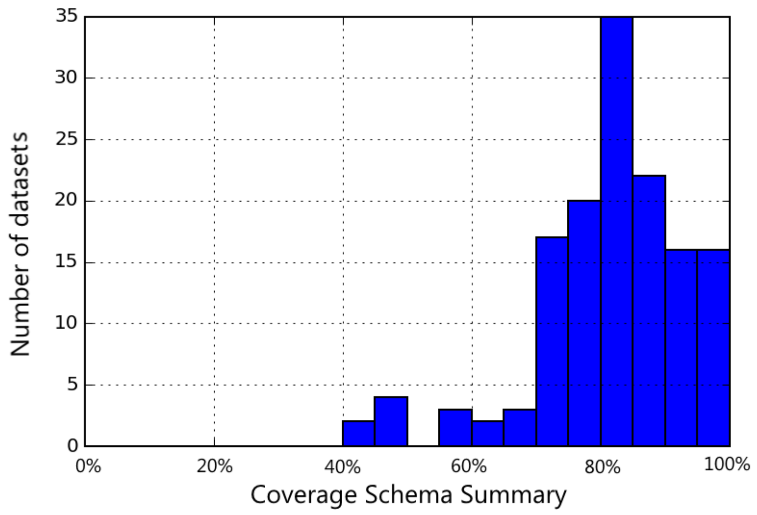


Figure 5.3: SS coverage distribution

$$coverage = n_{repr}/n_{tot}$$

In Figure 5.3, it is shown the distribution of the coverage for 140 SS (see Table 5.2)[8]. I obtained an high coverage, with an average value of 80%. An exception is given for 9 datasets that obtain a coverage under the 60%. By examining these cases, I discovered that these datasets describe an ontology in which the intensional knowledge is predominant.

Figure 5.4 reports the coverage distribution for the CSS; this time the number of datasets is 74 (see Table 5.2)[9]. Here, the average value of coverage is 54%. Given that, by using the CSS, I are not able to represent some instances of clustered classes, this was a predictable result.

Compression

The compression gives us the information about how much the SS shrinks the dimension of the original source. I choose to use the space saving measure. I extract

[8]The number of datasets is decreased, since some SPARQL endpoints does not supply the number of triples.

[9]The number of datasets is decreased, since some SPARQL endpoints does not supply the number of triples.

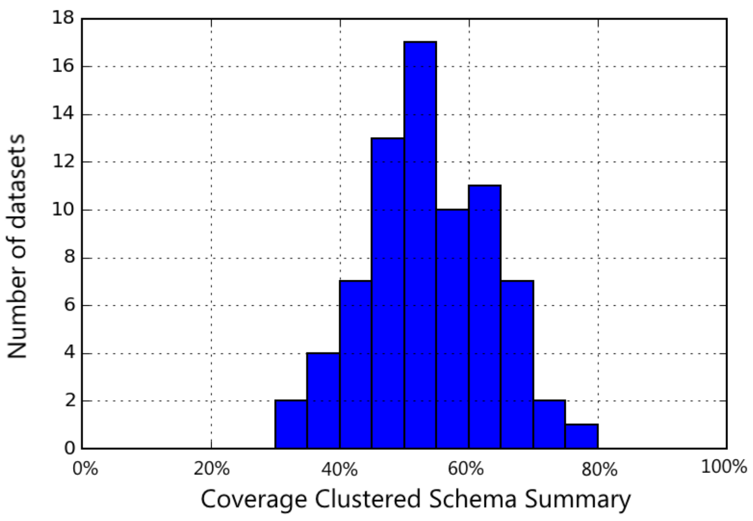


Figure 5.4: CSS coverage distribution

the number of triples in the original source that define the intensional knowledge of the source it self, i.e. the number of RDFS/OWL triples, n_{inten}. On the other hand, I calculate the number of triples of the RDFS ontology that represents the SS, n_{RDFS}. The space saving can be calculated as:

$$SpaceSaving = 1 - (n_{RDFS}/n_{inten})$$

The results of this analysis are provided in Figure 5.5. As you can see, I reach a space saving value of than 99% for more than the 90% of the datasets.

5.4.3 Application-driven ontology evaluation

This kind of evaluation, according to [LM10], leads to the definition of a “clean" environment where no other factors but the ontology influences the performances of the application [Sab+06]. My technique is directly bound to LODeX[BBP15b] which is a tool used to yield an interactive view of a SS and it aims to represent the structure of the dataset acting as a documentation of the dataset. Therefore, an evaluation of LODeX is greatly suited for this scope, indeed, LODeX is the tool built upon the Schema Summary model. I deeply described the tool in Chapter 4,

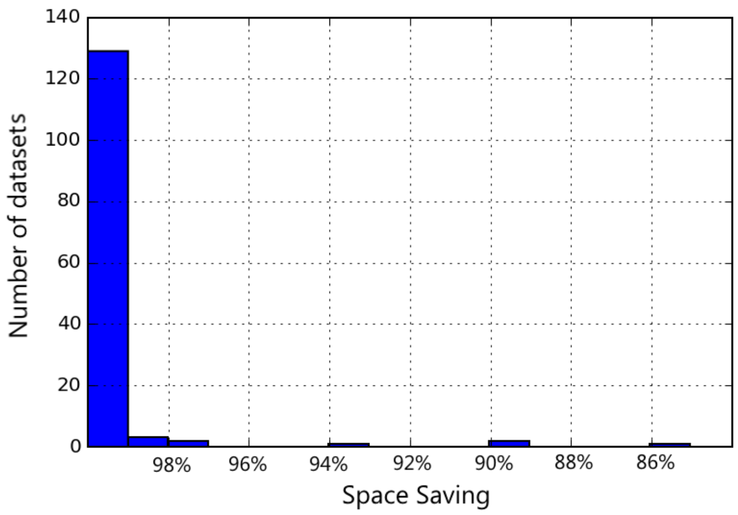


Figure 5.5: Space saving distribution.

where I provided a deep evaluation containing also a usability evaluation executed through an online survey in which LODeX obtained, as a summary result, a median SUS score of 82.5 that classifies LODeX as *Excellent*, according to [BKM09] (for major details see Section 4.5.3).

5.5 Conclusion

In this Chapter, I proposed the definition of a compress version of the Schema Summary, called Clustered Schema Summary. The SS exposes all the main classes and properties used within the datasets, either they are taken from external vocabularies or not. The CSS provides a more high level view of the classes and the properties used, it exploits multiple class instantiations to generate clusters of classes and decrease the overall size of the graph. Both SS and CSS are conceivable, and can be converted, as RDFS ontologies.

In this Chapter, I evaluated both the summaries according to ontology summarization metrics. Both SS and CSS obtained an high coverage of the LOD source on which they are applied. Moreover, the SS has proved to be a good compression method, since it reaches a compression of 99%.The LODeX application, that exploits the SS to supply users with a navigable interface, has demonstrated to be a powerful tool for both experts and unskilled user interested in the comprehension of a LOD source.

Part II

Context Semantic Analysis

Chapter 6

Context Semantic Analysis

6.1 Introduction

Recent years have seen growing number of knowledge bases that have been used in several domains and applications. Besides DBpedia [Aue+07], which is the heart of the Linked Open Data cloud [BL06], other important examples includes: Wikidata [VK14], a collaborative knowledge base which is replacing Freebase [Bol+08]; YAGO [SKW07], a huge semantic knowledge base, derived from Wikipedia, WordNet [Mil95] and GeoNames [Wic11]; Snomed CT [BD06], the best known ontology in the medical domain; and AGROVOC[Car+13], a multilingual agricultural thesaurus used recently for annotating some agricultural resources like CEREALAB[Ben+15]. The range of applications where these knowledge bases are used is heterogeneous; indeed, they provide a reliable source of knowledge that can be used for improving existing techniques, as NLP[Fri+01] or Information Retrieval[Fer+11], where has been shown that this knowledge enriched approaches can bring remarkable improvements.

In the context of documents pairwise similarity the most effective techniques are based on Vector Space Models of Semantics [TP+10]; the documents, composed by words, are represented in a vector space having words as dimensions. The way in which the components of each vector, i.e. the terms, are weighted can vary according to different strategies (term frequency, inverse document frequency, entropy and so on) and techniques of dimensionality reduction can be applied to automatically detect synonymy between words (LSA [Dum04]), and thus, improve the performance of the similarity estimation. Even if these approaches have proven their effectiveness in simulating human judgment [LPW05], they only rely on the textual information contained in the documents and they do not leverage external sources of knowledge. Consequently, they fail in detecting weak relation between concepts like in these two sentences: *"The band leaded by Mick Jagger will be playing in London next week"* and *"The Rolling Stones will open the concerts' season in Trafalgar Square"*. These two sentences contain highly related concepts and these relations can be easily found in a knowledge base like DBpedia, even if they are not contained explicitly in the text.

Usually, contextual information is deliberately omitted by an author of an article because it belongs to the common knowledge of the reader. Thus, a machine cannot find contextual information within a document, but it should leverage knowledge bases representing the machine-processable version of the common human knowledge. In this Chapter, I present a novel technique, called CSA (i.e. Context Semantic Analysis), for estimating pairwise similarity between document leveraging the information contained in a knowledge base. CSA is composed by several steps: starting from entities spotted in a document I extract a contextual graph; from this contextual graph I generate a context vector representing the document's context; I propose different weighting strategies of the context vectors by applying techniques of network analysis on the contextual graph (PageRank and Personalized PageRank) and techniques of matrix dimensionality reduction (SVD[MS07]); finally, I use these context vector for estimating the context similarity. I evaluate CSA by using the LP50 dataset [LPW05], a dataset composed by a small set of documents, where repeated similarity measures provided by humans are collected.

The following Chapter is structured as follow. Section 6.2 contains the Related Works, while Section 6.3 contains preliminaries useful for introduce the rest of Chapter. Then, CSA is described in Section 6.4 and Section 6.5 contains its evaluation respect to human judgments. Finally, the last Section contains some conclusion.

6.2 Related Work

Semantic similarity has been one the main research topic of the last few years[ZGC13]. This is due to wide range of application where it can be applied: Word Sense Disambiguation [PBP03], information retrieval [EMG11], multi-document summarization [Nas08] or for building domain specific recommendation systems [BP14][BPS14]. A lot of works on semantic similarity focused on computing semantic similarity between words and just recently efforts have been spent on the broader task of text similarity [BZG11][Agi+13]. In general, the most effective group of techniques are supervised methods that use large features sets [Bär+12] [Šar+12], however, these kind of approaches can difficultly be used in context where does not exist labeled data. Differently, the technique presented in this Chapter represents an unsupervised methods that not required any labeled data or parameter tuning for working.

CSA is, to the best of my knowledge, the first work that make use of a knowledge base for encoding the contexts of documents in a vector space and then use the context extracted for estimating inter documents similarity. My work can be linked to research works in Information Retrieval that make use of controlled vocabulary in domain specific contexts [Lan72] [BMS15]. In general, CSA can be seen as an example of an advanced Knowledge Organization System (KOS) [LZ08], because it make use of a knowledge base for representing the context of a document. Nevertheless, I do not generate an ad-hoc vocabulary but I make use of existing knowledge graphs containing general purpose knowledge.

6.3 Preliminaries

In this Section, I will introduce some concepts used later on in this Chapter.

6.3.1 Inter-Document similarity

The state of the art techniques for estimating inter-document similarity are primarily based on vector space models. A document is represented through a vector and the similarity between two documents is calculated as distance between two respective vectors.

Let C be a corpus composed of n documents, where d_i is the *i*th document. Each document d_i is composed by a sequence of terms. Let m be the number of distinct terms in C, I can represent each document d_i using the standard *bag-of-words* vector space representation:

$$\vec{d_i} = [t_{i1}, ..., t_{im}]$$

where t_{il} is the weight assigned to the term l in the document i. Now, I can define the matrix T as a matrix $n \times m$ formed by the n vectors d_i as rows. The weight of each elements of the matrix T is assigned by local and global weighting functions for making the cells content a better approximation of the corpus content. Thus, I define the wight of a generic cell as:

$$t_{ji} = L(i, j) \times G(i)$$

where $L(j, i)$ is the local weight of the term i within document j and $G(i)$ is the global weight of the term i across all documents in the corpus[NPM01]. Different strategies of weighting exist; in the following I present three of the most widely used.

Bag of words : It is the simplest way of producing a vector representation of documents and it uses only a local weight function, thus, $G(i) = 1$. The local weight $L(i, j)$ is equal to the number of time the term i appears in the document j.

Term Frequency - Inverse Document Frequency : This is the most famous kind of document vectorization strategy and it is based on term frequency as local weight function and inverse document frequency as global one. The term frequency formula is defined as follow:

Definition 12 $tf(i,j) = \frac{c(i,j)}{\sum_{k=1}^{I} c(k,j)}$

where $tf(i,j)$ is the term frequency of the term i in the document j, $c(i,j)$ is the number of times the term j appears in the document j and I is the total number of terms in the corpus.

The inverse document frequency weight is defined as follow:

Definition 13 $idf(i) = 1 + \log_2 \left(\frac{J}{df(i)} \right)$

where $idf(i)$ is the inverse document frequency of the term i, J is the number of documents in the corpus and $df(i)$ is the number of documents containing the term i.

Entropy : This kind of document vectorization uses as local weight the term frequency (see Definition 12) and the entropy as global weight. The entropy global weight is defined as follow:

Definition 14 $en(i) = 1 + \frac{\sum_{k=1}^{J} (p(i,k) \log_2 p(i,k))}{\log_2 J}$

where $p(i,j)$ is a conditional probability for the term i and the document j. It is calculated as $\frac{tf(i,j)}{gf(i)}$ where $gf(i)$ is the total number of times term i occurs in the whole collection.

Latent Semantic Analysis - LSA assumes there is a latent semantic structure in the documents it analyzes and its goal is to extract this latent semantic structure that is normally obscured by noise or variability in word usage. In order discover this structure, a single value decomposition (SVD) is applied to the matrix $n \times m$ T (the matrix representing the corpus of documents), where n is the number of documents and m is the number of terms. SVD implies selecting a dimensionality $d \leq m$ for the subspace representation, and finding the $n \times d$ orthonormal matrix U, the $d \times d$ diagonal matrix D and the $m \times d$ orthonormal matrix V that minimize the squared difference $||T - UDV^T||$. The resulting $n \times d$ matrix L is a least squares best fit to T obtained by zeroing all but the largest d coefficient of D.

The most common way of estimating the similarity of two documents is the *cosine similarity* and it is defined as follow:

Definition 15 $s_{kj} = \dfrac{\sum_i l_{ik} l_{ij}}{(\sum_i l_{ik}^2 \sum_i l_{ij}^2)^{\frac{1}{2}}}$

where l_k and l_j are the $k \times 1$ reduced rank vectors representing the k-th and j-th document.

6.3.2 PageRank and Personalized PageRank

The PageRank[Pag+99] is a famous method for ranking Web pages according to their structural importance within the graph they form (the edges are the hyper-links). The main concept is that every time an hyper-link from p_i to p_j exists, i grants importance to j, thus, the rank of j increases. Moreover, the strength of this grant depends on the importance of i. Alternatively, PageRank can also be interpreted as the result of a random walk process, also called random surfer model; in this case, the final rank of a page i represents the probability of a random walk of being on the page i after an high number of iterations.

Let G be a graph with n vertices $(v_1, ..., v_n)$, d_i be the outdegree of the node i and M be a $n \times n$ transition probability matrix defined as follow:

Definition 16 $M_{ij} = \begin{cases} \frac{1}{d_i} & \textit{if} \exists \textit{ edge from } i \textit{ to } j \\ 0 & \textit{if} \nexists \textit{ edge from } i \textit{ to } j \end{cases}$

Then, the PageRank vector R over the graph G can be calculated by resolving the following equation.

$$R = cMR + (1 - c)v$$

In the equation, c is the *dumping factor*, a scalar value between 0 and 1 and v is a $n \times 1$ vector in which each element is $\frac{1}{n}$. The first component of the equation simulates the behavior of the random surfer while the second part is needed to ensure that the random surfer does not remain trapped in some portion of the graph without outgoing edge. Indeed, it represents the probability of a surfer randomly jumping to another node of the graph. Moreover, it makes any graph fulfill the property of being aperiodic and irreducible and it ensures that the PageRank calculation converges to a unique stationary distribution. The most common values assigned to the *dumping factor* is between 0.85 to 0.95.

In the classic PageRank configuration the vector v is a stochastic normalized vector where all the values are $\frac{1}{n}$, so the random surfer has an equal probability to be teleported in any of the nodes of the graph G. In 2002 Personalized PageRank, also called Topic Sensitive PageRank, was introduced in [Hav02]. The difference is that the vector v can be non-uniform, thus, I can assign stronger probabilities to particular nodes and consequently, I can bias the computation of the rank R to be more influenced from those nodes. An extreme setting could be to concentrate all the probability mass to a single node i, in that case all random jumps would teleport the random surfer in i and, as a results, all the rank scores will be concentrated in i and in its neighbors nodes.

6.4 CSA

Context Semantic Analysis is performed on a corpus C of documents leveraging the content of a knowledge base KB. The technique is composed of three different steps:

- **Subgraph extraction**: a contextual subgraph cg_d containing the contextual information of the document d is extracted.
- **Context Vectors generation**: a context vector $\vec{cv}_d$ representing the context of the document is generated analyzing its contextual subgraph.
- **Context Similarity**: the Context Similarity is evaluated by comparing the context vectors of documents belonging to the corpus C.

6.4.1 Contextual Graph

The goal of this first step is to extract from a knowledge base KG, starting from a document d, a contextual graph cg_d containing all the contextual information that I can find in KG. A knowledge base is an RDF dataset containing general or domain specific information. Each RDF graph is composed by a set vertices V and a set of labeled edges E. The vertices can be divided in 3 disjoint sets: URIs U, blank nodes B and literals L. Two vertices connected by an edge represent a statement. Each statement is stored into a <*subject*,*predicate*,*object*> triple. I can define the whole RDF graph and consequently a knowledge base KG, according to [CWL14], as a set of triples.

Definition 17 $KG \subseteq (U \cup B) \times E \times V$

A subgraph cg_d is a particular subset of triples contained in KG. In particular, I utilize contextual graphs composed only of entities as vertices and object properties as edges. It follows that a conceptual graph is composed only of triples belonging to extensional knowledge. This choice is due to the willing of leveraging the implicit structure of a contextual graph for finding the most important entities representing the context of a document. A datatype property always leads to an isolated vertex (a literal), thus, it does not enrich a contextual graph with any useful information.

Contextual Graph Extraction

Before extracting the contextual graph of a document d, I have to identify the concepts belonging to KG which are explicitly mentioned in the document. I call this set of entities *starting entities* (SE). Named Entity Recognition[NS07] techniques allow to spot Linked Data entities within a text; I use API for finding the *starting entities* because the implementation of a novel technique of Named Entity Extraction is out of scope of this work. In particular, I use Dandelion API[Par+14] because I empirically noticed that this API returns good quality *starting entities*.

I extract a contextual graph starting from the set of *starting entities* of a document. In particular, the subgraph is composed by all the triples that connect with a path of length l at least 2 starting entities in KB.

Definition 18 $cg_d = \{< s,p,o > \mid < s,p,o > \in KG \wedge < s,p,o > \in Path(s_1, s_2) \wedge length(Path(s_1, s_2)) \leq l \wedge s_1 \in SE \wedge s_2 \in SE \wedge s_1 \neq s_2\}$

$Path(s_1, s_2)$ means that a path connecting the two entities s_1 and s_2 exists in KG.

For each couple of starting entities, according to l, I generate a set of SPARQL queries able to extract the triples belonging to any possible paths connecting those 2 entities. In Figure 6.1 you can see all the possible paths connecting two entities s_1 and s_2.

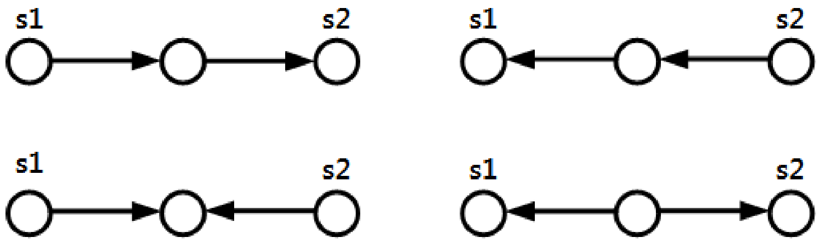


Figure 6.1: Existing paths of length $l = 2$ between s_1 and s_2.

I introduced also some heuristics for improving the quality of contextual graphs:

- Filter all the the possible cycles that can match some paths (with $l > 2$, as it is shown in Figure 6.2).
- Filter all the triples belonging to the intensional knowledge.

Example - Taking as example the two sentences used in the introduction, *"The band leaded by **Mick Jagger** will be playing in **London** next week"* and *"**The Rolling Stones** will open the concerts' season in **Trafalgar Square**"*, I can easy find two starting entities in DBpedia for each sentence: **Mick Jagger** and **London** for the first one; **The Rolling Stones** and **Trafalgar Square** for the last one. The relative contextual graphs with $l = 3$ are composed of 141 nodes for the first sentence and 4 for the second one.

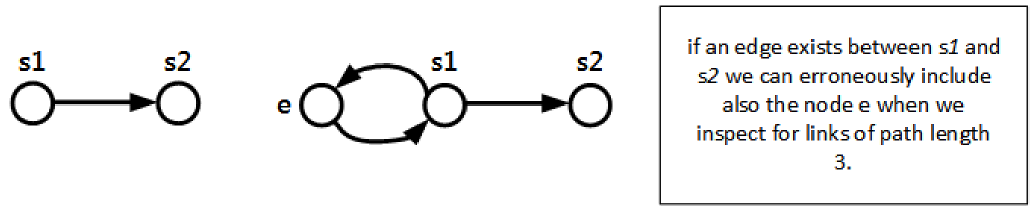

s1
s2
e
s1
s2
if an edge exists between s1 and s2 we can erroneously include also the node e when we inspect for links of path length 3.

Figure 6.2: Example of cycle that would include erroneously the node e in a contextual graph.

6.4.2 Context Vectors

In this section I present different strategies of context vector weighting given a contextual graph cg_d. Let the corpus $C = \{d_1, ..., d_n\}$ be a set of documents and $E = \{e_1, ..., e_m\}$ the set of distinct entities occurring in the contextual graphs of the documents belonging to C. The context of a document is then represented as a m-dimensional vector $\vec{cv_j}$. Let $s(d_j, e_i)$ be a generic weighting function of the entity $e_i \in E$ in the document $d_j \in C$. Then, I can define a vector representation of the context of the document d_j as follow:

Definition 19 $\vec{cv_j} = (s(d_j, e_1), ..., s(d_j, e_m))$

Thus, the context of the whole corpus of documents C can now be represented through a matrix $n \times m$ N, where each row j contains the context vector of the *jth* document.

Weighting functions

A weighting function in this context have to spread weight to entities of a generic contextual graph according to the importance of these entities within the graph. Thus, PageRank is a natural choice because the main goal of this algorithm is to rank graph nodes according their importance (major details about the algorithms can be found in Section 6.3.2). I also used the Personalized PageRank and I set up the personalization vector giving an equal probability to the starting entities; so let $\vec{p}$ be the personalization vector $(p_1, ..., p_m)$, its values are assigned as follow:

Definition 20 $p_i = \begin{cases} \frac{1}{|SE|} & \mathit{if}\, e_i \in SE \\ 0 & \mathit{if}\, e_i \notin SE \end{cases}$

where $|SE|$ is the number of starting entities ($\in SE$).

I used both PageRank and Personalized PageRank with different configuration of *dumping factor* as weight function; Table 6.1 shows the different configuration used.

Name		**Dumping Factor**
PageRank	Personalized PageRank	
r@50	pr@50	0.5
r@75	pr@75	0.75
r@85	pr@85	0.85
r@90	pr@90	0.90
r@95	pr@95	0.95

Table 6.1: PageRank and Personalized PageRank configurations

Example - In Table 6.2 are reported the context vectors extracted from DBpedia of the two sentences reported in the Introduction. In the first sentence, the starting entities were *London* and *Mick Jagger* and the contextual graph was composed of 141 nodes. In the second sentence the starting entities were *The Rolling Stones* and *Trafalgar Square*; in this case, the contextual graph was composed of just 4 nodes. As you can see, a lot of weight for both vectors is spread to the entity *London* which is target of a lot of object properties in DBpedia. Usually, the upper level concepts in a generic knowledge base are target of object properties and, thus, importance flows in them using PageRank and Personalized PageRank. Nevertheless, it is apparent that these two algorithms behave slightly differently, indeed, the PageRank tends to arrange weight in all the context graph's nodes, due to the equal probability of the random surfer to be teleported in each of them, while the Personalized PageRank focuses all the weight in the starting entities and their neighbors.

6.4.3 Context Similarity

In this section, I present how estimating the inter-document similarity based on contextual information and then I propose strategies for combining the context

Entities	**Sentence 1**		**Sentence 1**	
	pr@85	r@85	pr@85	r@85
London	**0.405**	**0.635**	0.356	0.471
The Rolling Stones	0.285	0.185	**0.226**	**0.138**
Greater London	0.235	0.105	-	-
Mick Jagger	**0.075**	**0.029**	-	-
Casino Boogie	0.000	0.002	-	-
Sweet Virginia	0.000	0.002	-	-
Torn and Frayed	0.000	0.002	-	-
Exile on Main St.	0.000	0.003	-	-
Watch That Man	0.000	0.002	-	-
Aladdin Sane	0.000	0.003	-	-
Time (David Bowie)	0.000	0.002	-	-
Polydor Records	-	-	0.192	0.254
Trafalgar Square	-	-	**0.226**	**0.138**

Table 6.2: Context Vectors r@85 and pr@85 of the two example sentences

similarity with existing techniques of inter-document similarity based on text (a deep explanation of the state of the art techniques used in this section is proposed in Section 6.3.1).

From the previous step, I obtained a matrix N $n \times m$, where each row contains the context vector of one document. I use the *cosine similarity* (see Definition 15) for calculating the context similarity between two documents. In this case, the cosine similarity of two documents will range from 0 to 1, since PageRank and Personalized PageRank can not return negative ranks. The angle between two context vectors can not be greater than 90°.

Example - Now I have all the components for estimating the context similarity of the two sentences used as example in the introduction and for which I showed the context vectors in Table 6.2. As I said, in this example common techniques of similarity based upon text are not effective, given that the two sentences do not share any common words. Using CSA I obtain a similarity score of 0.78 by using $r@85$ vectors and 0.61 by using $pr@85$. Even if this is just a toy example I obtained a remarkable results where the classic techniques of document similarity failed in detecting this latent similarity. In the next Section, will be proposed a deeper evaluation of the effectiveness of CSA.

Combining CSA with text vectorization methods

I also sift through different ways of combining CSA with existing text based techniques for estimating the similarity between documents. In particular, I used several well known techniques explained in Section 6.3.1; Table 6.3 contains all the configurations used.

Table 6.3: Different kinds of text vectorization used

Name	Weighting	n. components
bg	Bag of Words	-
tf-idf	tf-idf	-
tf-idf#1000	tf-idf	1000
tf-idf#500	tf-idf	500
tf-idf#100	tf-idf	100
tf-idf#10	tf-idf	10
tf-e	tf-entropy	-
tf-e#1000	tf-entropy	1000
tf-e#500	tf-entropy	500
tf-e#100	tf-entropy	100
tf-e#10	tf-entropy	10

The main goal here is to defines ways of combining CSA with existing techniques, insofar as they use different information for estimating the similarity: text and information taken form a knowledge base. In particular, I defined three different strategies for combining CSA with the state of the art methods.

Alpha

The first method is the simplest one and it only combines the results of the context and text based similarity; let $s_{CSA}(i, j)$ be the contextual similarity and $s_t(i, j)$ the result of the cosine similarity by using the classic methods between two documents i and j, I combine the two values as follow:

$$s = (1 - \alpha)s_t(i, j) + \alpha s_{CSA}(i, j)$$

where α is a weighting parameter utilized for combining the two similarity measures. Given that both the similarities scores are obtained as result of cosine similarity calculated on vectors that not contain negatives values, both $s_{CSA}(i,j)$ and $s_t(i,j)$ range from 0 to 1, so also s ranges from 0 to 1. In Table 6.4 all the different values of α utilized are shown.

Table 6.4: Different values of α used

Name	α **value**
alpha 0.05	0.05
alpha 0.1	0.1
alpha 0.15	0.15
alpha 0.25	0.25
alpha 0.5	0.5
alpha 0.75	0.75

Max

The second method merging is called **Max** and it is based on the intuition that text based and context similarity mostly detect independent aspect of similarity, indeed, contextual information are not explicitly contained within the text. Thus, this method produces $s(i,j)$ by selecting the maximum value between $s_{CSA}(i,j)$ and $s_t(i,j)$:

$$s = max(s_t(i,j), s_{CSA}(i,j))$$

Join

Finally, let N be the $n \times m$ matrix containing the context vectors and T be the $n \times t$ matrix containing the vectors weighted according to the text before applying any technique of dimensionality reduction. These two matrices have the same number of rows n so I can perform an horizontal concatenation creating a new matrix J of dimension $n \times (t + m)$:

$$J = [M, N]$$

The matrix J contains information about both text and context and I can calculate the similarity between two document always using the *cosine similarity*. Moreover, I can apply SVD to J for trying detect some correlation, non only between the term, but also between terms and entities.

6.5 Evaluation - Correlation with human judges

The most common and effective way for evaluating techniques of inter-document similarity is to calculate the correlation between the results computed and the similarity expressed by humans. Unfortunately, only a dataset of documents with this characteristics exists and this is primarily due to huge effort needed for creating such a dataset. The name of this dataset is LP50[1] [LPW05] and it contains 50 documents, selected from the Australian Broadcasting Corporations news mail service, evaluated by 83 University of Adelaide students (29 males and 54 females), with a mean age of 19.7 years. The most effective way for comparing a novel technique with human judges is to calculate the Pearson product-moment correlation coefficient[LL89] on the similarity results; the Pearson correlation measures the linear correlation between two variables, in my case, the similarities expressed by human judges and the ones computed. I conducted experiments using all the techniques explained in Section 6.3.1 and a summary of the results is shown in Table 6.5. As you can see, the technique that performs better is the tf-idf. The use of dimensionality reduction does not bring any improve in this experiment and this is due to the restricted number of documents involved in the experiment that does not allow to detect latent semantics in the vectorized corpus.

In this Section, two kinds of evaluation are proposed. First of all, the correlation between the different ranking strategies and the human judges will be evaluated. Secondly, the evaluation is focused in finding the best way of combining CSA with the classical methods of text vectorization.

Table 6.5: Pearson correlation of the results obtained with different weighting techniques respect to the similarity measure obtained from human judges (LP50 Dataset).

Name	Correlation	p-value
bg	0.19	2.2e-10
tf-idf	0.51	6.98e-67
tf-idf#1000	0.51	6.98e-67
tf-idf#500	0.51	6.98e-67
tf-idf#100	0.51	6.98e-67
tf-idf#10	0.33	5.99e-27
tf-e	0.43	5.46e-46
tf-e#1000	0.43	5.46e-46
tf-e#500	0.43	5.46e-46
tf-e#100	0.43	5.46e-46
tf-e#10	0.27	3.35e-18

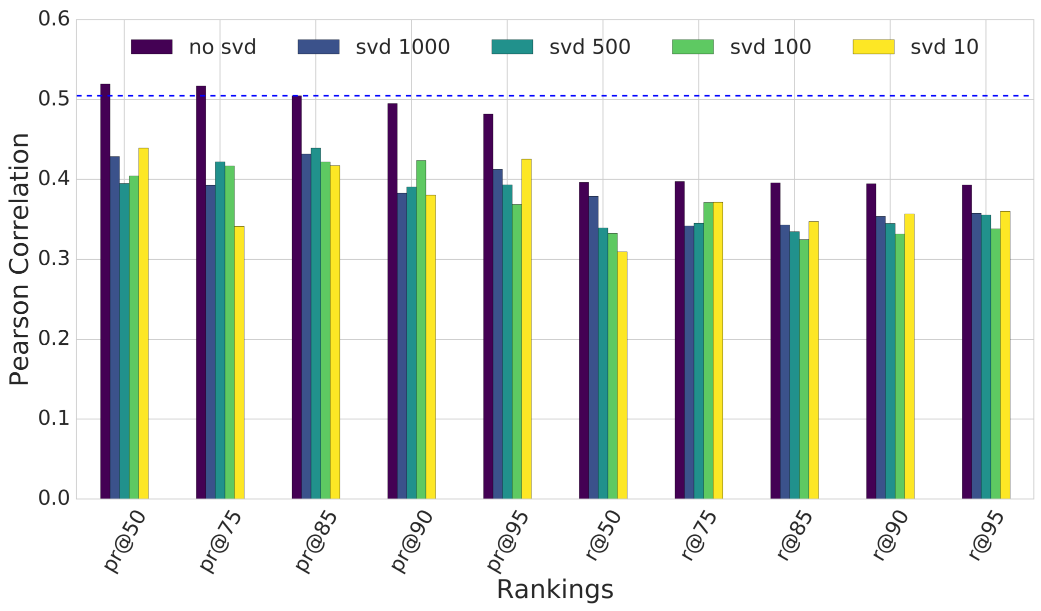


Figure 6.3: Person correlation with human judgments (LP50 Dataset) of different configuration of rankings and dimensionality reduction.

6.5.1 CSA all alone

In this Section, the different techniques of rankings described in Section 6.4.2 will be taken in exam for evaluating the effectiveness of CSA respect to human judges by using the LP50 Dataset. I used Dandelion API[Par+14] for spotting entities within the documents and then I extracted for contextual graphs by setting the path length to, 2 following a large body of evidence from previous related work [Hul+13] [NP12].

The different configurations tested are listed in Table 6.1 and the results are shown in Figure 6.3. As you can see, the Personalized PageRank outperforms the classical PageRank; the main reason is that Personalized PageRank spreads rank in the neighborhood of the starting entities and for that reason is more suitable in producing vectors that represent better the context of a document. Surprisingly, two configurations of CSA works slightly better than the best configuration of tf-idf (in Figure 6.3 the tf-idf score is displayed as a dotted blue line). Moreover, it is quite obvious that applying dimensionality reduction to the matrix containing context vectors does not bring any improvement, but this is certainly due to the low number of documents involved in the experiment and I can not draw any conclusion.

6.5.2 Merging methods

In this Section, different kind of evaluations aiming to discover the best way of merging CSA with classical methods of text vectorization are proposed by using the LP50 dataset as benchmark. The classical methods used in these experiments are listed in Table 6.5, I used all the ranking configuration of CSA shown in Table 6.1 and all the merging strategies described in Section 6.4.3.

Given the large number of possible configurations, I perform an exploratory analysis which is summarized in the box plot in Figure 6.4. The goal of this Figure is to have a glimpse of the percentage of improvement distribution varying the kind of merging and the size of dimensionality reduction applied to the CSA

[1]`https://webfiles.uci.edu/mdlee/LeePincombeWelsh.zip`

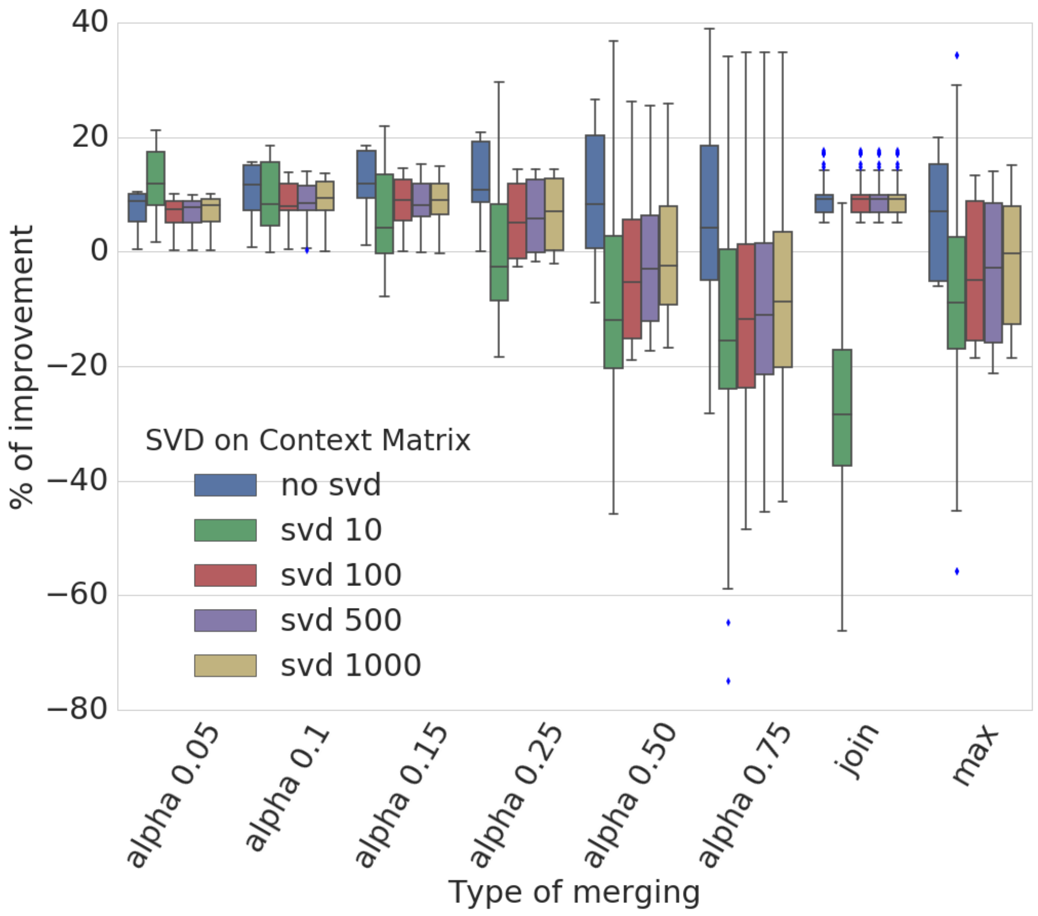


Figure 6.4: Box plot showing the percentage of improvement distributions varying merging method and SVD configuration.

matrix. It is obvious that the most stable techniques of merging are: *alpha* merging with low values of α (less than 0.25); *join* merging with no dimensionality reduction or with dimensionality reduction in a number of dimensions greater or equal than 50.

Going more in detail, the distribution of the Pearson correlation scores obtained by merging CSA with tf-idf and td-e is shown in Figure 6.5; it is evident that tf-idf works better respect to human judges than tf-e. Therefore, the following analysis will include only results involving the merging between CSA and tf-idf.

A detailed view of the different merging configuration performance, varying CSA ranking, is shown in Figure 6.6. As you can see, the Personalized PageRank

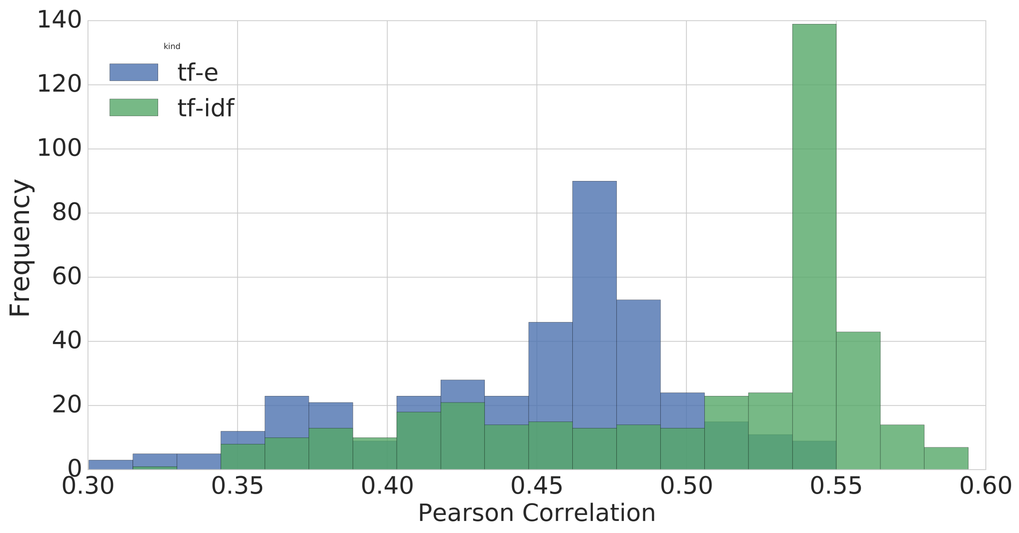


Figure 6.5: Comparison of the distributions of the Pearson correlation scores obtained by merging different configurations of CSA with tf-e and tf-idf.

always outperform the classic PageRank and the best performances are obtained by with α very low where has been applied a strong dimensionality reduction to the CSA matrix. In order to further inspect this behavior I produced heat maps of similarity matrices; a similarity matrix is a square matrix of dimension $n \times n$, where n is number of documents in the corpus and each cell of index (i, j) contains the measure of similarity among the document i and j. These heat maps are shown in Figure 6.7. The similarity matrix of human judges is shown in Figure 6.7a, while the similarity matrix computed through the tf-idf is shown in Figure 6.7b. The differences induced by SVD for *pr@50* is visualized in Figure 6.7c and 6.7d; as you can see, the dimensionality reduction causes the range of values composing the similarity matrix to be less equally distributed, so, context overlapping produces high values of similarity, thus, this behavior explains why a very low α produces improvements with a strong dimensionality reduction. Finally, the best configurations obtained are shown in Table 6.6. As it has been stated before, applying a strong dimensionality reduction combined with low values of α produces the highest improvements (from 12.410% to 14.16%) but also the not reduced version obtains good level of improvement (11.46%). The final choice relies on the number of documents involved, indeed, the whole context matrix have to be built

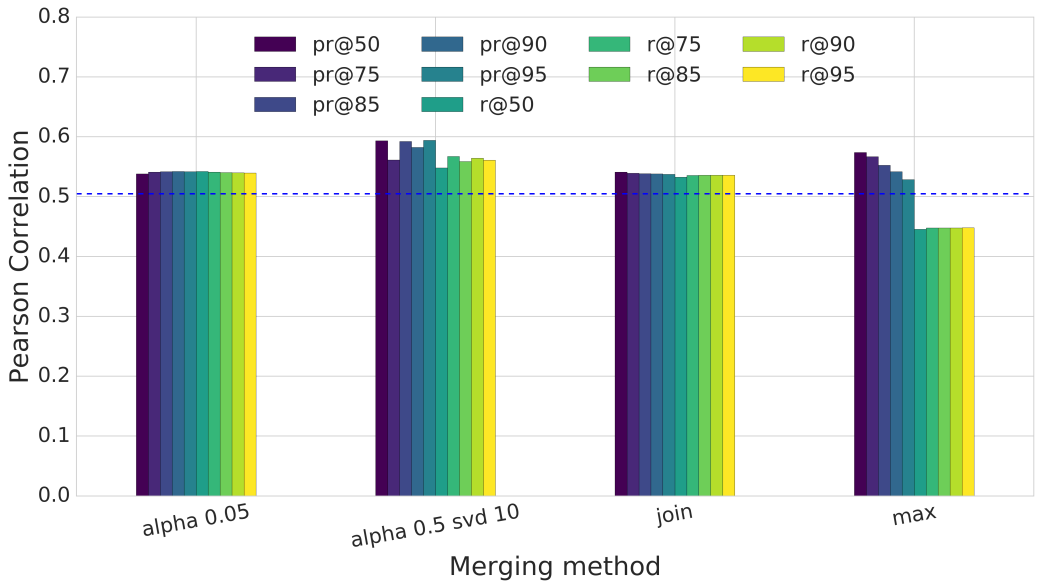
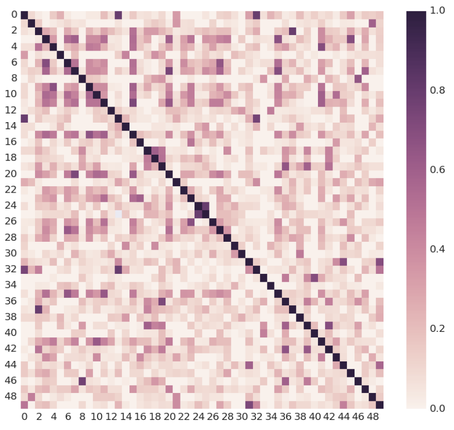


Figure 6.6: Person correlation with human judgments of the different ranking strategies of CSA merged with tf-idf grouped for type of merging.

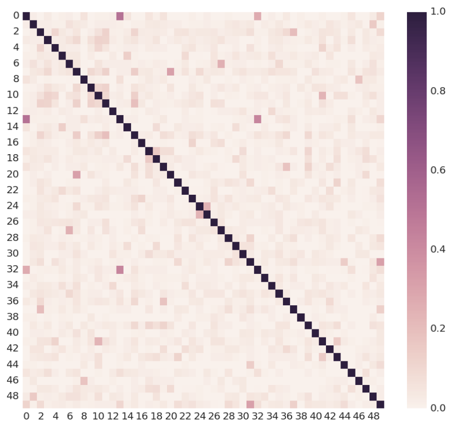
for utilizing SVD and each time new documents are added the reduction should be recomputed. Otherwise, if I implement a system that uses plain context vectors they could be easily computed and stored as document's metadata and each time a new document is added its context vector could computed independently.

6.6 Conclusion and Future Work

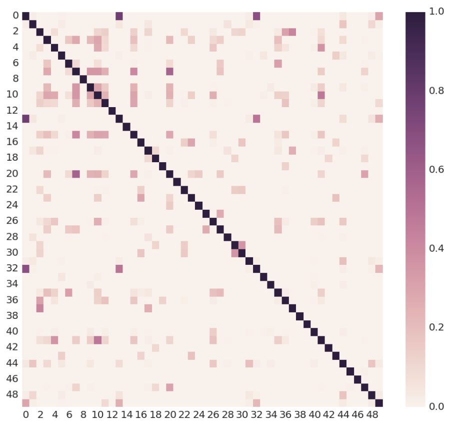
In this Chapter, I presented a novel technique, called CSA (Context Semantic Analysis) for exploiting the content of a knowledge base for representing the context of a set of documents in vectorized way and estimate the context similarity between them. A context vector can be easily stored as metadata of a document and used, when needed, for computing the context similarity with other documents. CSA has been evaluated respect to human judges by using the LP50 Dataset, a reference benchmark in the context of inter-document similarity estimation, and it obtained a Pearson correlation score greater than tf-idf, a reference technique in the field. I also inspected different strategy for combining CSA with



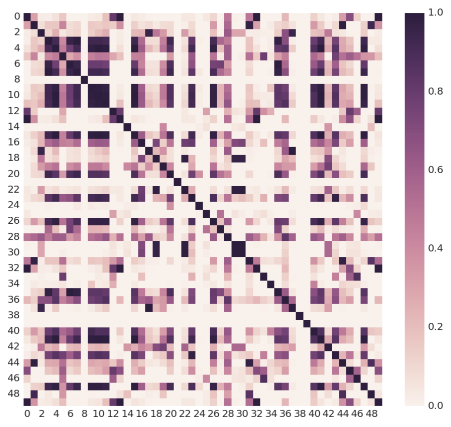
(a) human judges



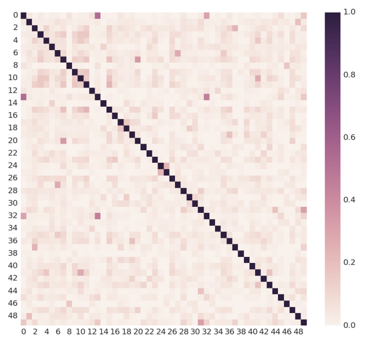
(b) tf-idf



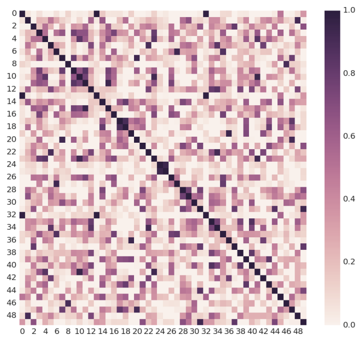
(c) pr@50



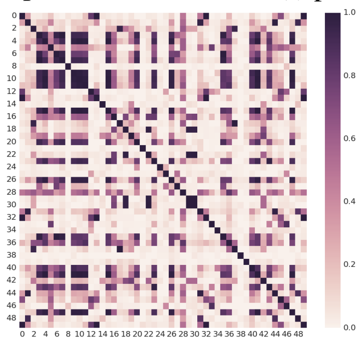
(d) pr@50 svd 10



(e) pr@50 alpha 0.05



(f) pr@50 join



(g) pr@50 max

Figure 6.7: Heat maps of similarity matrices obtained with different techniques; x and y contain the 50 document and for each cell is represented through a color the similarity score.

Table 6.6: Pearson correlation and percentage of improvement obtained by merging different configuration of CSA with tf-idf.

Rank	Merging	Pearson Corr.	% of improvement	SVD
pr@95	alpha 0.05	0.594	14.160	svd 10
pr@50	alpha 0.05	0.593	14.040	svd 10
pr@85	alpha 0.05	0.592	13.890	svd 10
pr@50	alpha 0.1	0.590	13.540	svd 10
pr@95	alpha 0.1	0.586	12.980	svd 10
pr@85	alpha 0.1	0.583	12.490	svd 10
pr@90	alpha 0.05	0.582	12.410	svd 10
pr@75	alpha 0.25	0.576	11.460	no svd
pr@50	max	0.574	11.140	no svd
pr@85	alpha 0.25	0.574	11.110	no svd
pr@50	alpha 0.25	0.573	11.010	no svd
pr@90	alpha 0.25	0.571	10.700	no svd
pr@50	alpha 0.50	0.571	10.670	no svd
pr@50	alpha 0.15	0.568	10.250	svd 10
pr@75	alpha 0.50	0.568	10.180	no svd
r@75	alpha 0.05	0.567	10.120	svd 10
pr@95	alpha 0.25	0.567	10.040	no svd
pr@75	max	0.567	10.010	no svd
pr@85	alpha 0.15	0.566	9.830	no svd
pr@75	alpha 0.15	0.565	9.790	no svd
pr@90	alpha 0.15	0.565	9.710	no svd
r@90	alpha 0.05	0.564	9.630	svd 10
pr@95	alpha 0.15	0.563	9.440	no svd
pr@75	alpha 0.05	0.561	9.170	svd 10
pr@50	alpha 0.15	0.561	9.080	no svd
r@95	alpha 0.05	0.561	9.070	svd 10
pr@95	alpha 0.15	0.561	9.050	svd 10

inter-document similarity techniques based on text, like tf-idf, reaching good percentage of improvement (more than 14%).

Even if this results are promising, It is necessary to test CSA with other knowledge bases to ensure that this technique is independent from the knowledge graph used. I'm planning to perform further tests using WikiData and UMLS.

Publications related to this thesis

[BBP] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “Online Index Extraction from Linked Open Data Sources”. In: *Proc.of the LD4IE Workshop 2014 co-located with the ISWC 2014*.

[BBP14a] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “A Visual Summary for Linked Open Data sources”. In: *International Semantic Web Conference (Posters & Demos) 2014* (2014).

[BBP14b] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “Online Index Extraction from Linked Open Data Sources”. In: *Linked Data for Information Extraction (LD4IE) Workshop held at International Semantic Web Conference* (2014).

[BBP15a] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “LODeX: A tool for Visual Querying Linked Open Data”. In: *International Semantic Web Conference (ISWC 2015 Posters & Demos)*. CEUR Workshop Proceedings. CEUR-WS.org, 2015.

[BBP15b] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “Visual Querying LOD Sources with LODeX”. In: *Proceedings of the 8th International Conference on Knowledge Capture*. K-CAP 2015. Palisades, NY, USA: ACM, 2015, 12:1–12:8. ISBN: 978-1-4503-3849-3. DOI: 10.1145/2815833.2815849. URL: http://doi.acm.org/10.1145/2815833.2815849.

[Ben+15] Domenico Beneventano, Sonia Bergamaschi, Serena Sorrentino, Maurizio Vincini, and Fabio Benedetti. “Semantic annotation of the

CEREALAB database by the AGROVOC linked dataset". In: *Ecological Informatics* 26 (2015), pp. 119–126.

Bibliography

[Abe+14] Ziawasch Abedjan, Toni Grütze, Anja Jentzsch, and Felix Naumann. "Profiling and mining RDF data with ProLOD++". In: *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*. Ed. by Isabel F. Cruz, Elena Ferrari, Yufei Tao, Elisa Bertino, and Goce Trajcevski. IEEE, 2014, pp. 1198–1201. ISBN: 978-1-4799-3480-5. DOI: `10.1109/ICDE.2014.6816740`. URL: `http://dx.doi.org/10.1109/ICDE.2014.6816740`.

[AG05] Renzo Angles and Claudio Gutierrez. "Querying RDF data from a graph database perspective". In: *The Semantic Web: Research and Applications*. Springer, 2005, pp. 346–360.

[Agi+13] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. "sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity". In: *In* SEM 2013: The Second Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics*. Citeseer. 2013.

[Ale+09] Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. "Describing Linked Datasets." In: *LDOW*. 2009.

[Aue+07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[Aue+12] Sören Auer, Jan Demter, Michael Martin, and Jens Lehmann. “LODStats–an extensible framework for high-performance dataset analytics”. In: *Knowledge Engineering and Knowledge Management*. Springer, 2012, pp. 353–362.

[BAG12] Josep Maria Brunetti, Sören Auer, and Roberto Garca. “The Linked Data Visualization Model”. In: *International Semantic Web Conference (Posters & Demos)*. 2012.

[Ban11] Kyle Banker. *MongoDB in action*. Manning Publications Co., 2011.

[BBP] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “Online Index Extraction from Linked Open Data Sources”. In: *Proc.of the LD4IE Workshop 2014 co-located with the ISWC 2014*.

[BBP14a] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “A Visual Summary for Linked Open Data sources”. In: *International Semantic Web Conference (Posters & Demos) 2014* (2014).

[BBP14b] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “Online Index Extraction from Linked Open Data Sources”. In: *Linked Data for Information Extraction (LD4IE) Workshop held at International Semantic Web Conference* (2014).

[BBP15a] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “LODeX: A tool for Visual Querying Linked Open Data”. In: *International Semantic Web Conference (ISWC 2015 Posters & Demos)*. CEUR Workshop Proceedings. CEUR-WS.org, 2015.

[BBP15b] Fabio Benedetti, Sonia Bergamaschi, and Laura Po. “Visual Querying LOD Sources with LODeX”. In: *Proceedings of the 8th International Conference on Knowledge Capture*. K-CAP 2015. Palisades, NY, USA: ACM, 2015, 12:1–12:8. ISBN: 978-1-4503-3849-3. DOI: 10.1145/2815833.2815849. URL: http://doi.acm.org/10.1145/2815833.2815849.

[BD06] L Bos and K Donnelly. "SNOMED-CT: The advanced terminology and coding system for eHealth". In: *Stud Health Technol Inform* 121 (2006), pp. 279–290.

[Ben+15] Domenico Beneventano, Sonia Bergamaschi, Serena Sorrentino, Maurizio Vincini, and Fabio Benedetti. "Semantic annotation of the CEREALAB database by the AGROVOC linked dataset". In: *Ecological Informatics* 26 (2015), pp. 119–126.

[Ben15] Fabio Benedetti. *LODeX*. Version 2.0. Aug. 1, 2015. URL: `https://github.com/linkTDP/lodex`.

[BG04] Dan Brickley and Ramanathan V Guha. "{RDF vocabulary description language 1.0: RDF schema}". In: (2004).

[Biz+08] Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. "Linked Data on the Web (LDOW2008)". In: *Proceedings of the 17th International Conference on World Wide Web*. WWW '08. Beijing, China: ACM, 2008, pp. 1265–1266. ISBN: 978-1-60558-085-2. DOI: `10.1145/1367497.1367760`. URL: `http://doi.acm.org/10.1145/1367497.1367760`.

[BKM09] Aaron Bangor, Philip Kortum, and James Miller. "Determining what individual SUS scores mean: Adding an adjective rating scale". In: *Journal of usability studies* 4.3 (2009), pp. 114–123.

[BL06] Tim Berners-Lee. *Linked data*. 2006. URL: `http://www.w3.org/DesignIssues/LinkedData.html`.

[BL12] Lorenz Bühmann and Jens Lehmann. "Universal OWL axiom enrichment for large knowledge bases". In: *Knowledge Engineering and Knowledge Management*. Springer, 2012, pp. 57–71.

[BMS15] Sonia Bergamaschi, Riccardo Martoglia, and Serena Sorrentino. "Exploiting semantics for filtering and searching knowledge in a software development context". In: *Knowledge and Information Systems* 45.2 (2015), pp. 295–318.

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. “D^3 data-driven documents”. In: *Visualization and Computer Graphics, IEEE Transactions on* 17.12 (2011), pp. 2301–2309.

[Bol+08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. “Freebase: a collaboratively created graph database for structuring human knowledge”. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM. 2008, pp. 1247–1250.

[BP14] Sonia Bergamaschi and Laura Po. “Comparing LDA and LSA Topic Models for Content-Based Movie Recommendation Systems”. In: *Web Information Systems and Technologies*. Springer, 2014, pp. 247–263.

[BPS14] Sonia Bergamaschi, Laura Po, and Serena Sorrentino. “Comparing Topic Models for a Movie Recommendation System.” In: *WEBIST (2)*. 2014, pp. 172–183.

[Bro96] John Brooke. “SUS-A quick and dirty usability scale”. In: *Usability evaluation in industry* 189.194 (1996), pp. 4–7.

[BZG11] Daniel Bär, Torsten Zesch, and Iryna Gurevych. “A Reflective View on Text Similarity.” In: *RANLP*. 2011, pp. 515–520.

[Bär+12] Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. “Ukp: Computing semantic textual similarity by combining multiple content similarity measures”. In: *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics. 2012, pp. 435–440.

[Car+13] Caterina Caracciolo, Armando Stellato, Ahsan Morshed, Gudrun Johannsen, Sachit Rajbhandari, Yves Jaques, and Johannes Keizer.

"The AGROVOC linked dataset". In: *Semantic Web* 4.3 (2013), pp. 341–348.

[Che+12] Gong Cheng, Feng Ji, Shengmei Luo, Weiyi Ge, and Yuzhong Qu. "BipRank: ranking and summarizing RDF vocabulary descriptions". In: *The Semantic Web*. Springer, 2012, pp. 226–241.

[CPF13] Klitos Christodoulou, Norman W Paton, and Alvaro AA Fernandes. "Structure inference for linked data sources using clustering". In: *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM. 2013, pp. 60–67.

[CWL14] Richard Cyganiak, David Wood, and Markus Lanthaler. "RDF 1.1 concepts and abstract syntax". In: *W3C Recommendation* 25 (2014), pp. 1–8.

[DR11] Aba-Sah Dadzie and Matthew Rowe. "Approaches to visualising linked data: A survey". In: *Semantic Web* 2.2 (2011), pp. 89–124.

[Dum04] Susan T Dumais. "Latent semantic analysis". In: *Annual review of information science and technology* 38.1 (2004), pp. 188–230.

[EMG11] Ofer Egozi, Shaul Markovitch, and Evgeniy Gabrilovich. "Concept-based information retrieval using explicit semantic analysis". In: *ACM Transactions on Information Systems (TOIS)* 29.2 (2011), p. 8.

[Fer+11] Miriam Fernández, Iván Cantador, Vanesa López, David Vallet, Pablo Castells, and Enrico Motta. "Semantically enhanced Information Retrieval: an ontology-based approach". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 9.4 (2011), pp. 434–452.

[Fer14] Sébastien Ferré. "Expressive and Scalable Query-Based Faceted Search over SPARQL Endpoints". English. In: *The Semantic Web ISWC 2014*. Ed. by Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandei, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble. Vol. 8797.

Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 438–453. ISBN: 978-3-319-11914-4. DOI: 10.1007/978-3-319-11915-1_28.

[Fie+04] M Fieschi et al. “Achieving" Source Transparency" in the UMLS® Metathesaurus®”. In: *Medinfo 2004*. Vol. 107. IOS Press. 2004, p. 371.

[Fri+01] Carol Friedman, Pauline Kra, Hong Yu, Michael Krauthammer, and Andrey Rzhetsky. “GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles”. In: *Bioinformatics* 17.suppl 1 (2001), S74–S82.

[GK04] Jeremy J. Carroll Graham Klyne. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. Feb. 2004. URL: http://www.w3.org/TR/rdf-concepts/.

[Got+12] Thomas Gottron, Malte Knauf, Stefan Scheglmann, and Ansgar Scherp. “Explicit and implicit schema information on the linked open data cloud: Joined forces or antagonists”. In: *the 11th International Semantic Web Conference*. 2012.

[Har+10] Andreas Harth, Katja Hose, Marcel Karnstedt, Axel Polleres, Kai-Uwe Sattler, and Jürgen Umbrich. “Data summaries for on-demand queries over linked data”. In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 411–420.

[Hav02] Taher H Haveliwala. “Topic-sensitive pagerank”. In: *Proceedings of the 11th international conference on World Wide Web*. ACM. 2002, pp. 517–526.

[HBF09] Olaf Hartig, Christian Bizer, and Johann-Christoph Freytag. “Executing SPARQL queries over the web of linked data”. In: *The Semantic Web-ISWC 2009*. Springer, 2009, pp. 293–309.

[HEZ10] Philipp Heim, Thomas Ertl, and Jürgen Ziegler. "Facet Graphs: Complex Semantic Querying Made Easy". In: *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I.* 2010, pp. 288–302. DOI: 10.1007/978-3-642-13486-9_20. URL: http://dx.doi.org/10.1007/978-3-642-13486-9_20.

[HM03] Volker Haarslev and Ralf Möller. "Racer: A Core Inference Engine for the Semantic Web." In: *EON*. Vol. 87. 2003.

[Hor+02] Ian Horrocks et al. "DAML+OIL: A Description Logic for the Semantic Web". In: *IEEE Data Eng. Bull.* 25.1 (2002), pp. 4–9.

[HS10] Steve Harris and Andy Seaborne. "SPARQL 1.1 query language". In: *W3C working draft* 14 (2010).

[Hul+13] Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. "Unsupervised Graph-based Topic Labelling Using Dbpedia". In: *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*. WSDM '13. Rome, Italy: ACM, 2013, pp. 465–474. ISBN: 978-1-4503-1869-3. DOI: 10.1145/2433396.2433454. URL: http://doi.acm.org/10.1145/2433396.2433454.

[HZL08] Philipp Heim, Jürgen Ziegler, and Steffen Lohmann. "gFacet: A Browser for the Web of Data". In: *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW'08) Koblenz, Germany, December 3, 2008.* 2008. URL: http://ceur-ws.org/Vol-417/paper5.pdf.

[Höf+14] Patrick Höfler, Michael Granitzer, Eduardo E. Veas, and Christin Seifert. "Linked Data Query Wizard: A Novel Interface for Accessing SPARQL Endpoints". In: *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International*

World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014. 2014. URL: `http://ceur-ws.org/Vol-1184/ldow2014_paper_06.pdf`.

[Jai+10] Prateek Jain, Pascal Hitzler, Peter Z. Yeh, Kunal Verma, and Amit P. Sheth. “Linked Data Is Merely More Data”. In: *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*. AAAI, 2010.

[KB14] Christoph Kiefer and Abraham Bernstein. “LODmilla: a Linked Data Browser for All”. In: *Proceedings of the Posters and Demos Track of 10th International Conference on Semantic Systems*. SEMANTiCS2014. Leipzig, Germany: CEUR-WS.org, 2014, pp. 31–34.

[KBS07] Christoph Kiefer, Abraham Bernstein, and Markus Stocker. “The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-based Semantic Web Tasks”. In: *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*. ISWC’07/ASWC’07. Busan, Korea: Springer-Verlag, 2007, pp. 295–309. ISBN: 3-540-76297-3, 978-3-540-76297-3. URL: `http://dl.acm.org/citation.cfm?id=1785162.1785185`.

[Kon+12] Mathias Konrath, Thomas Gottron, Steffen Staab, and Ansgar Scherp. “Schemex—efficient construction of a data catalogue by stream-based indexing of linked data”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 16 (2012), pp. 52–58.

[Kun90] Hideko S Kunii. “Graph Data Model”. In: *Graph Data Model*. Springer, 1990, pp. 7–20.

[Lan72] Frederick Wilfrid Lancaster. “Vocabulary control for information retrieval.” In: (1972).

[LL89] I Lawrence and Kuei Lin. “A concordance correlation coefficient to evaluate reproducibility”. In: *Biometrics* (1989), pp. 255–268.

[LM10] Ning Li and Enrico Motta. “Evaluations of user-driven ontology summarization”. In: *Knowledge Engineering and Management by the Masses*. Springer, 2010, pp. 544–553.

[LMd10] Ning Li, Enrico Motta, and Mathieu d’Aquin. “Ontology summarization: an analysis and an evaluation”. In: (2010). Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010), [held in conjunction with] 9th International Semantic Web Conference (ISWC2010).

[LN94] Gerhard Lakemeyer and Bernhard Nebel. *Foundations of Knowledge representation and Reasoning*. Springer, 1994.

[LPW05] M Lee, Brandon Pincombe, and Matthew Welsh. “An empirical evaluation of models of text document similarity”. In: *Cognitive Science* (2005).

[LW09] Andreas Langegger and Wolfram Woss. “RDFStats-an extensible RDF statistics generator and library”. In: *Database and Expert Systems Application, 2009. DEXA’09. 20th International Workshop on*. IEEE. 2009, pp. 79–83.

[LZ08] Marcia LEI ZENG. “Knowledge organization systems (KOS)”. In: *Knowledge organization* 35.2-3 (2008), pp. 160–182.

[Mil95] George A Miller. “WordNet: a lexical database for English”. In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[MS07] Andriy Mnih and Ruslan Salakhutdinov. “Probabilistic matrix factorization”. In: *Advances in neural information processing systems*. 2007, pp. 1257–1264.

[MVH+04] Deborah L McGuinness, Frank Van Harmelen, et al. “OWL web ontology language overview”. In: *W3C recommendation* 10.2004-03 (2004), p. 10.

[Nas08] Vivi Nastase. “Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2008, pp. 763–772.

[NP12] Roberto Navigli and Simone Paolo Ponzetto. “BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network”. In: *Artificial Intelligence* 193 (2012), pp. 217–250.

[NPM01] Preslav Nakov, Antonia Popova, and Plamen Mateev. “Weight functions impact on LSA performance”. In: *EuroConference RANLP* (2001), pp. 187–193.

[NS07] David Nadeau and Satoshi Sekine. “A survey of named entity recognition and classification”. In: *Lingvisticae Investigationes* 30.1 (2007), pp. 3–26.

[Pag+99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. “The PageRank citation ranking: bringing order to the Web.” In: (1999).

[PAG09a] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. “Semantics and complexity of SPARQL”. In: *ACM Transactions on Database Systems* 34.3 (Aug. 2009), pp. 1–45. ISSN: 03625915. DOI: `10.1145/1567274.1567278`. URL: `http://portal.acm.org/citation.cfm?doid=1567274.1567278`.

[PAG09b] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. “Semantics and complexity of SPARQL”. In: *ACM Transactions on Database Systems (TODS)* 34.3 (2009), p. 16.

[Par+14] Stefano Parmesan, Ugo Scaiella, Michele Barbera, and Tatiana Tarasova. “Dandelion: from raw data to dataGEMs for developers”. In: *Proceedings of the ISWC Developers Workshop*. 2014, pp. 1–6.

[PBP03] Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. “Using measures of semantic relatedness for word sense disambiguation”. In: *Computational linguistics and intelligent text processing*. Springer, 2003, pp. 241–257.

[PH04] Zhiming Pan and I Horrocks. *Description Logics: reasoning support for the Semantic Web*. University of Manchester, 2004.

[PKK12] Fabian Prasser, Alfons Kemper, and Klaus A Kuhn. “Efficient distributed query processing for autonomous RDF databases”. In: *Proceedings of the 15th International Conference on Extending Database Technology*. ACM. 2012, pp. 372–383.

[PMd08] Silvio Peroni, Enrico Motta, and Mathieu dAquin. “Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures”. In: *The Semantic Web*. Springer, 2008, pp. 242–256.

[Sab+06] Marta Sabou, Vanessa Lopez, Enrico Motta, and Victoria Uren. “Ontology selection: Ontology evaluation on the real semantic web”. In: (2006).

[Sah+09] Satya S Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, and Ahmed Ezzat. “A survey of current approaches for mapping of relational databases to RDF”. In: *W3C RDB2RDF Incubator Group Report* (2009), pp. 113–130.

[Skj] Martin G Skjæveland. “Sgvizler: A javascript wrapper for easy visualization of sparql result sets”. In: *ESWC 2012*.

[SKW07] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. “Yago: a core of semantic knowledge”. In: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, pp. 697–706.

[TP+10] Peter D Turney, Patrick Pantel, et al. “From frequency to meaning: Vector space models of semantics”. In: *Journal of artificial intelligence research* 37.1 (2010), pp. 141–188.

[VK14] Denny Vrandečić and Markus Krötzsch. “Wikidata: a free collaborative knowledgebase”. In: *Communications of the ACM* 57.10 (2014), pp. 78–85.

[Wan+04] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. “Ontology based context modeling and reasoning using OWL”. In: *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*. Ieee. 2004, pp. 18–22.

[Wic11] Marc Wick. “GeoNames”. In: *Symposium on Space-Time Integration in Geography and GIScience*. Vol. 201. 1. 2011.

[Wu+08] Gang Wu, Juanzi Li, Ling Feng, and Kehong Wang. “Identifying potentially important concepts and relations in an ontology”. In: *The Semantic Web-ISWC 2008*. Springer, 2008, pp. 33–49.

[ZCQ07] Xiang Zhang, Gong Cheng, and Yuzhong Qu. “Ontology summarization based on rdf sentence graph”. In: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, pp. 707–716.

[ZGC13] Ziqi Zhang, Anna Lisa Gentile, and Fabio Ciravegna. “Recent advances in methods of lexical semantic relatedness–a survey”. In: *Natural Language Engineering* 19.04 (2013), pp. 411–479.

[Zha+09] Xiang Zhang, Gong Cheng, Wei-Yi Ge, and Yu-Zhong Qu. “Summarizing vocabularies in the global semantic web”. In: *Journal of Computer Science and Technology* 24.1 (2009), pp. 165–174.

[ZLQ06] Xiang Zhang, Hongda Li, and Yuzhong Qu. “Finding important vocabulary within ontology”. In: *The Semantic Web–ASWC 2006*. Springer, 2006, pp. 106–112.

[Šar+12] Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. “Takelab: Systems for measuring semantic text similarity”. In: *Proceedings of the First Joint Conference on Lexical* *and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics. 2012, pp. 441–448.

Software

[Ben15] Fabio Benedetti. *LODeX*. Version 2.0. Aug. 1, 2015. URL: https://github.com/linkTDP/lodex.