

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 01 October 2011

Présentée par

Mustafa AL BAKRI

Thèse dirigée par **Marie-Christine Rousset**
et codirigée par **Manuel Atencia**

préparée au sein **Laboratoire d'Informatique de Grenoble (LIG)**
et de **EDMSTII**

Uncertainty-Sensitive Reasoning over the Web of Data

Thèse soutenue publiquement le **15 December 2014** sous réserve de
l'**autorisation à soutenir**,
devant le jury composé de :

M. Mohand-Said Hacid

Professeur Université Claude Bernard Lyon 1, Rapporteur

M. Andrea Tettamanzi

Professeur Université Nice Sophia Antipolis, Rapporteur

M. Jérôme Euzenat

Directeur de recherche INRIA Grenoble Rhône-Alpes, Examineur

Mme. Marie-Laure Mugnier

Professeur Université de Montpellier 2 , Examineur

Mme. Marie-Christine Rousset

Professeur Université Joseph Fourier (Grenoble 1), Directeur de thèse

M. Manuel Atencia

Maître de conf. Université Pierre Mendès-France (Grenoble 2), Co-Directeur de thèse



“I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A ”Semantic Web”, which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The ”intelligent agents” people have touted for ages will finally materialize”

Tim, Berners-Lee

Abstract

In this thesis we investigate several approaches that help users to find useful and trustful information in the Web of Data using the Semantic Web technologies. In this purpose, we tackle two research issues: Data Linkage in Linked Data and Trust in Semantic P2P Networks.

We model the problem of data linkage in Linked Data as a reasoning problem on possibly decentralized data. We describe a novel Import-by-Query algorithm that alternates steps of sub-query rewriting and of tailored querying the Linked Data cloud in order to import data as specific as possible for inferring or contradicting given target same-as facts. Experiments conducted on real-world datasets have demonstrated the feasibility of this approach and its usefulness in practice for data linkage and disambiguation. Furthermore, we propose an adaptation of this approach to take into account possibly uncertain data and knowledge, with a result, the inference of same-as and different-from links having some weights. In this adaptation we modeled uncertainty as probability values. Our experiments have showed that our the adapted approach scales to large data sets and produces meaningful probabilistic weights.

Concerning trust, we introduce a trust mechanism for guiding the query-answering process in Semantic P2P Networks. Peers in Semantic P2P Networks organize their information using separate ontologies and rely on alignments between their ontologies for translating queries. Trust in such a setting is subjective and estimates the probability that a peer will provide satisfactory answers for specific queries in future interactions. In order to compute trust, the mechanism exploits the information provided by alignments, along with the one that comes from peer's experiences. The calculated trust values are refined over time using Bayesian inference as more queries are sent and answers received. For the evaluation of our mechanism, we build a semantic P2P bookmarking system (TrustMe) in which we can vary different quantitative and qualitative parameters. The results show the convergence of trust, and highlight the gain in the quality of peers' answers —measured with precision and recall— when the process of query answering is guided by our trust mechanism.

Contents

Abstract	iii
Contents	iv
List of Figures	vii
List of Tables	viii
Abbreviations	ix
1 Introduction	1
1.1 Research problems addressed in this thesis	4
1.2 Contributions and Outline of the Thesis	6
1.3 Publications	7
Under Submission	7
I Reasoning over Linked-Data	8
2 Inferring same-as facts from Linked Data: an iterative import-by-query approach	9
2.1 Introduction	9
2.2 Illustrative Example	11
2.3 Related work	13
2.4 Formal background and problem statement	14
2.4.1 URIs, URLs and namespaces	14
2.4.2 RDF datasets in Linked Data	15
2.4.3 Queries over RDF datasets in Linked Data	15
2.4.4 Deductive RDF datasets	16
2.4.5 Problem statement	17
2.5 The iterative Import-by-Query Algorithm	18
2.5.1 The QESQ algorithm	19
2.5.2 Combining forward and backward chaining	21
2.6 Experiments	22
2.6.1 Experimental Goals and Set-Up	23
2.6.2 Experimental Results	24
2.7 Conclusion	25

3	Reasoning With Uncertainty For Data Linkage	28
3.1	Introduction	28
3.2	Sources of Uncertainty	29
3.3	Modeling Uncertainty in Linked Data	30
3.3.1	Illustrative Example	30
3.3.2	Formal Background	33
3.4	ProbFR: The Probabilistic forward Reasoner	35
3.5	Data complexity and approximation techniques	38
3.5.1	Maximum size of an event expression	38
3.5.2	Approximating probabilistic weights	38
3.6	Setting up the acceptance threshold	39
3.7	Experiments	40
3.7.1	Experimental Set-Up	40
3.7.2	Experimental Results	41
3.8	Extending the import by query approach to uncertainty	43
3.8.1	The Probabilistic Iterative Import-by-query	44
3.8.2	The PQESQ algorithm	45
3.9	Related Work and Conclusion	47
II	Trust in Semantic Web	49
4	Trust in Networks of Ontologies and Alignments	50
4.1	Introduction	50
4.1.1	Semantic P2P Networks	51
4.1.2	Trust in Semantic P2P Networks	51
4.1.3	Summary of our Contribution	52
4.2	Preliminaries	53
4.2.1	Ontologies and Populated Ontologies	53
4.2.2	Alignments	54
4.2.3	Peers and Acquaintance Graphs	54
4.2.4	Queries and Query Translations	55
4.3	The Trust Mechanism	55
4.3.1	Definition and Estimation of Trust	56
4.3.2	Computation and Refinement of Trust Estimation	57
	No direct experience: alignment-based trust.	58
	Direct experience: trust refinement.	58
4.3.3	Update of Populated Ontologies	60
	Provenance-based trust aggregation.	61
	Discard of instances.	62
4.3.4	Use of Trust	62
4.4	Experimentation and Evaluation	62
4.4.1	Experimental Design	63
	Generation of P2P networks	63
	Construction of the ontologies	63
	Construction of the reference populated ontologies	64
	Construction and peer assignment of initial populated ontologies	65

Construction of initial alignments	65
Simulation of query answering	65
4.4.2 Experimental Results	67
Convergence of trust	67
Gain in the quality of peer answers when using trust	69
4.5 Conclusions and Related Work	71
4.5.1 Trust	73
4.5.2 Ontology and Schema Matching	75
5 Demo: TrustMe I got what you mean	78
5.1 Introduction	78
5.2 Goals of the Demo	79
5.3 Setting Up the P2P Bookmarking Scenario	80
Taxonomy Generation.	80
Alignment Generation.	80
Network Generation.	80
Taxonomy Population.	80
5.4 TrustMe	81
6 Conclusion and Future Works	85
6.1 Data Linkage	86
6.2 Trust in Semantic P2P Networks	87
A The set of rules used in the experiments	90
B The set of uncertain rules used in the experiments	94
Bibliography	97

List of Figures

1.1	The Linked Open Data diagram. It shows datasets that have been published in Linked Data format. It is based on metadata collected and curated by contributors to the Data Hub as well as on metadata extracted from a crawl of the Linked Data web conducted in April 2014.	3
2.1	A sample of the INA RDF facts and an extract of the INA vocabulary.	11
2.2	DBpedia facts and an extract of the DBpedia vocabulary.	12
2.3	The resulted external sub-queries submitted to DBpedia and their returned answers	13
3.1	A sample of certain facts from the INA and DBpedia datasets	30
3.2	The distribution of the number of the inferred facts over the range of the probabilistic weights	42
3.3	The distribution of the facts that are confirmed to be wrong by contradiction over the range of the probabilistic weights	43
4.1	Beta density functions for different alignment relations.	59
4.2	Ontology of cinema and music.	64
4.3	Ontology of cities and music.	64
4.4	The number of peers has no significant impact on the speed of convergence. . .	69
4.5	The lower thresholds of trust are, the slower convergence of trust is.	69
4.6	The higher the F-measure an initial alignment has, the faster convergence is. . .	70
4.7	The number of peers has no significant impact on $F^t(\text{Paris})$	71
4.8	The higher thresholds of trust are, the higher $precision^t(\text{Paris})$ is.	72
4.9	The lower thresholds of trust are, the higher $recall^t(\text{Paris})$ is.	72
4.10	The higher the F-measure an initial alignment has, the higher $F^t(\text{Paris})$ is. . .	73
5.1	A snapshot of the Main screen of TrustMe	81
5.2	Snapshot showing the topology of a generated network with 16 peers.	82
5.3	Snapshot showing the added articles in Paris city class of one peer's taxonomy.	83
6.1	Summary of the contributions	85

List of Tables

1.1	The number of datasets available in Linked Open Data grouped by topical domain.	4
2.1	R1-R3 are local rules about INA properties; R4-R5 are rules involving mappings from the INA and DBpedia vocabularies; R6 translates transitivity of owl:sameAs, and R7 relates owl:sameAs and owl:differentFrom.	12
2.2	A rule for dealing with similarities between names.	23
2.3	Gain in reduced imported facts.	25
2.4	Gain in runtime.	25
3.1	A set of rules. The rule r_1 is uncertain while the other rules are certain	31
3.2	The event expression of the inferred facts and their simplified form.	32
3.3	The number of inferred facts confirmed to be correct in each sample	42
4.1	Evaluation of the matchers' alignments.	65
4.2	Parameters of our simulations.	66
4.3	Values of the parameters in the experiments of convergence of trust and gain in query answering by using trust.	68
5.1	Comparing Precision and recall averaged values for Paris homonyms.	83
A.1	The set of rules that disambiguate homonymous person inside the INA dataset	90
A.2	The set of rules that disambiguate homonymous person inside the INA dataset	91
A.3	The set of rules that disambiguate homonymous person inside the INA dataset	91
A.4	Rules that disambiguate homonyms person based on their roles in a video	92
A.5	Rules that used in the renaming techniques for the ground sameAs and differentFrom facts	92
A.6	Rules that express the semantics of owl:sameAs and owl:differentFrom	93
B.1	Rules that disambiguate homonyms person based on birthYear and deathYear	94
B.2	Rules that disambiguate homonyms person based on their occupations	95
B.3	Rules that disambiguate homonyms person based on their roles in a video	96
B.4	Rules that disambiguate homonyms person if they are related to the same video	96

Abbreviations

CSV	Comma Seperated Values
dQSQ	distributed Query Sub Query
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
INA	Institute Nationale Audiovisuele
OWL	Web Ontology Language
P2P	Peer to Peer
PQESQ	Probabilistic Query External Sub Query
QSQ	Query Sub Query
QESQ	Query External Sub Query
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	EXtensible Markup Language

Chapter 1

Introduction

“What are the latest news in a particular country?”, “Which is the best store to buy a particular stuff?”. These are examples of questions that may come in our minds and that very often we cannot answer based on the information we know. In order to reach answers to these questions we typically search for trustworthy sources of information and query them to complete our knowledge. Then, we aggregate and process what we already know with the different information that we obtained to conclude answers.

Searching for information over the Web: The World Wide Web has changed the way we share and search for information by breaking the barriers to publish and access documents. It is an open global information space of linked documents accessible and actively used by more than 40% ¹ of the world population. It contains more than 41 billions of different documents [45] that are linked together using Hypertext links. The contents of these documents form a mosaic of information on plenty of topics. Internet users utilize keyword-based search engines (Google, Bing, Yahoo, . . .) to reach the web documents that are likely to contain the information they are looking for.

Traditionally, data related to hypertext documents in the web are provided in different raw formats (e.g. CSV, XML or HTML tables) that miss much of their structure or semantics. Moreover, the hyperlinks between documents are not expressive to link individual entities that appear in these documents (e.g. expressing that two individuals are synonyms or homonyms). This limits the ability of keyword-based search engines to provide all the correct documents and just the correct documents that talk about a particular keyword. Also, the incredibly huge amount of data that are available on the web reveals a growing desire to facilitate the direct access to the data and not only to the documents they are related to. Up to now, keyword-based search engines cannot provide a precise answer to a particular user query of the form: who is the author

¹In 2014 according to statistic studies that are detailed in http://en.wikipedia.org/wiki/Global_Internet_usage.

of the film “Paris”? Instead, they may return millions of hypertext documents that are likely to be related to keywords that appear in the query.

Semantic Web: At the turn of the century, the Semantic Web has been proposed as an extension to the traditional web “in which information is given well-defined meaning, better enabling computers and people to work in cooperation” [16]. The emphasis is now on the interoperability across data sources both on the data (assertional) level the semantic (terminological) level as a guarantee for the Web to be significantly improved. To achieve such interoperability Semantic Web offers standard means for representing knowledge in a way that:

- Entities or resources are identified using URIs.
- The data about these entities are given in a standard machine readable format called RDF. RDF data can be serialized using many formats (e.g. RDF/XML, Turtle, N-Triples ...).
- Data are attached with a schema (or ontology) that provides the meaning of the data in a declarative way. The schema is defined using standard languages such as RDFS and the OWL. Like other formal languages, all these languages have their own defined semantics.

Linked Data: The Semantic Web community has studied for many years the theoretical and technical challenges regarding the diffusion and manipulation of structured data and their semantics. The result is Linked Data: a set of best practices for publishing and connecting structured data using the infrastructure of the Web. Tim Berners-Lee outlined four principles of Linked Data in [14]. They are paraphrased along the following lines:

- Use URIs to denote things.
- Use HTTP URIs so that these things can be referred to and looked up (“dereferenced”) by people and user agents.
- Provide useful information about the thing when its URI is dereferenced, leveraging standards such as RDF, SPARQL.
- Include links to other related things (using their URIs) when publishing data on the Web.

The Linked data has successfully attracted various data publishers including companies (e.g. BestBuy [37, 38]), organisations (e.g. BBC [43], and New York Times²), governmental entities (e.g. data.gov.uk, data.gov) or even crowd-sourced communities (e.g. DBpedia [19]). The result is a Web of Data composed of more than 1014 connected data sets³. They contain more than 32

²<http://data.nytimes.com/>

³According to the “State of the LOD Cloud 2014 webpage” <http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/>

billions of RDF triples on a veritable plethora of topics. Table 1.1 gives an overview of the topical domains of the datasets that are available and accessible in the year of writing the thesis, 2014, while Figure 1.1 gives an overview of the linkage relationships between the datasets.

Topic	Datasets	%
Government	183	18.05%
Publications	96	9.74%
Life sciences	83	8.19%
User-generated content	48	4.73%
Cross-domain	41	4.04%
Media	22	2.17%
Geographic	21	2.07%
Social web	520	51.28%
Total	1014	

TABLE 1.1: The number of datasets available in Linked Open Data grouped by topical domain.

Semantic Peer-to-Peer Networks: Peer-to-Peer (P2P) architectures are known for their scalability, since the same data can be offered by many peers and thus the accessibility to these data is increased. Also, the decentralization of these architectures increases their reliability, as the data can be replicated on many peers distributed all around the world. Semantic P2P Networks use P2P as an underlying architecture for the Semantic Web. It has two main advantages compared to other architectures like the client-server architecture. First, it provides the open environment in which peers free to publish the information they want that conforms to a chosen ontology. Second, it gives the ability to users to control their own data and ontology and to choose whom can access to their information. This is clearly a huge advantage in terms of privacy on Semantic Web.

1.1 Research problems addressed in this thesis

As we have shown before, many new technologies have been proposed to help users to publish the information they want and to reach the exact information they are looking for using the web infrastructure. However, as in many other occasions, the exercise has proven to be trickier than thought at the outset. There are still many open research issues to be solved before reaching such a real global open data and knowledge space in which it is easy to find precise and complete answers to possible complex questions. In this thesis we consider two main research problems:

- **Data linkage in Linked Data:** Linked Data has offered the standards needed to publish and access highly-interoperable data on the Web. However, the freedom it gives to the data publishers may turn the web into islands of data. For instance, data publishers are free to identify a real world object with any URI they want and to publish in Linked

Data descriptions about this object based on the chosen URI. Thus, different datasets may describe the same real world objects with entities that are identified with different URIs which are not explicitly connected together. Although Linked Data has encouraged publishers to express such links in a standard way using owl:sameAs, the huge size of data available in the Web of Data makes the task of identifying and linking to all these entities hard in practice. Entities with different URIs that describe the same real object can also appear in the same dataset. This occurs frequently in datasets that are created as collaborations of different data authors belonging to the same organizations.

In this thesis we address the problem of discovering links in linked data, i.e. data linkage problem, and we answer the following questions:

- How to model the data linkage problem in the settings of Linked Data?
- How to identify the data, possibly distributed over the Web of Data, useful for computing links between two entities without importing the whole published data?
- How to combine the data available in a particular dataset together with the data imported from the Web of Data to compute a set of certain links inside this dataset?
- How to extend the proposed approach for data linkage to compute uncertain links using uncertain knowledge?
- How to determine automatically among the calculated uncertain links the correct ones?

• **Modeling and computing trust in semantic P2P networks:**

Semantic P2P networks provide a partial solution for the privacy problem while exchanging data over them thanks to the underlying P2P architecture. The P2P decentralized nature allows users to decide exactly with which peers they want to exchange data. However, the question is: how the users can choose among all available peers the ones on which they trust to access and to share data with?

Answering the above question is mandatory in semantic P2P networks as information returned by other peers can be untrue, incomplete or outdated. In this thesis we address the following research questions about trust:

- How to benefit from the Semantic Web technology to model trust in semantic P2P networks?
- How trust can be estimated for a particular peer and refined dynamically overtime as more information about that peer is available?
- How can trust be used for improving the quality in semantic P2P networks?
- How can we measure the gain of using trust in a semantic P2P network?

1.2 Contributions and Outline of the Thesis

The thesis is organized in two parts, each of them consisting of two chapters. A chapter of general conclusions and future work is included, as well as a couple of appendices.

- **Part 1** focuses on the problem of data linkage in Linked Data. It models it as a reasoning problem on possibly decentralized data.
 - **Chapter 1** focuses on proposing an approach for data linkage adapted to the decentralized nature of Linked Data. The approach allows to infer a set of certain same-as and different-from links between homonymous entities in a particular data set using the certain knowledge attached to the data. For this, it describes a novel import-by-query algorithm that alternates steps of sub-query rewriting and of tailored querying the Linked Data in order to import data as specific as possible for inferring or contradicting given target same-as facts. It shows also the experiments that are conducted on a real-world dataset. The experiments have demonstrated the feasibility of this approach and its usefulness in practice for data linkage and disambiguation.
 - **Chapter 2** extends the approach presented in Chapter 1 to take into account possibly uncertain data and knowledge, with as a result, the inference of same-as and different-from links having some probabilistic weights. It shows also how to interpret the uncertainty values calculated for the inferred links and how to benefit from logical inference to return the best results to the user.
- **Part 2** focuses on the problem of modeling and calculating trust in semantic P2P networks.
 - **Chapter 3** introduces a mechanism of trust adapted to semantic P2P networks. It presents the theoretical model of this mechanism as well as the way we used Bayesian inference to compute trust in such a setting. We report also in this chapter the experiments that we have performed to evaluate our trust mechanism.
 - **Chapter 4** shows TrustMe, a system built to demonstrate the trust mechanism proposed in Chapter 3. Specifically, it demonstrates the trust mechanism in a case of a semantic P2P bookmarking system where peers exchange URLs of articles about topics they are interested in. We highlight the gain in the quality of peers' answers, measured with precision and recall, when the process of query answering is guided by our trust mechanism. As a particular case, we show how trust overcomes homonymy. Moreover, a trust-based ranking of articles allows to distinguish the articles relevant to a category from the ones related to its homonymous categories.
- **Appendix A** provides the set of certain rules that were used during the experiments explained in Chapter 1. It gives also explanations of their meaning and origins.

- **Appendix B** gives the set of uncertain rules that were used during the experiments explained in Chapter 2. It shows their rankings and their probabilistic weights. Descriptions of the rules are provided as well.

1.3 Publications

The following publications have been derived from this thesis:

- Mustafa Al-Bakri, Manuel Atencia, Marie-Christine Rousset. Approche itérative à base de règles Datalog pour inférer des liens “same-as” à l’aide du Web des données. presented in the National Conference of Advanced Data Bases (BDA) 2014
- Manuel Atencia, Mustafa Al-Bakri, Marie-Christine Rousset: Trust in networks of ontologies and alignments. International Journal on Knowledge and Information Systems December 2013, [ISSN 0219-1377] [eISSN 0219-3116]
- Mustafa Al-Bakri, Manuel Atencia, Marie-Christine Rousset. TrustMe, I Got What You Mean! Knowledge Engineering and Knowledge Management Lecture Notes in Computer Science Volume 7603, 2012, pp 442-445

Under Submission Mustafa Al-Bakri, Manuel Atencia, Steffen Lalande, Marie-Christine Rousset. Inferring same-as facts from Linked Data: An iterative import-by-query approach

Part I

Reasoning over Linked-Data

Chapter 2

Inferring same-as facts from Linked Data: an iterative import-by-query approach

2.1 Introduction

Linked Data promotes exposing, sharing and connecting data and knowledge on the Semantic Web by using URIs, namespaces and RDF. URIs are used for referencing entities described by RDF facts, but also for referencing namespaces providing access to existing vocabularies that can be reused and shared across data sources. The adoption of the Linked Data principles has led to a Web of data of several billions of RDF triples interlinked by several millions of RDF links and connecting thousands of data sources on a wide range of domains.

Links between datasets take the form of RDF triples the subject of which is a URI reference in the namespace of one dataset, while the predicate and/or object are URIs pointing into the namespaces of other datasets. A particular case corresponds to same-as facts expressing that two URIs refer to the same real-world object. In an open environment like the Web, different information providers may publish data about URIs representing the same real-world entity. Some datasets serve as linking hubs in the Web of Data, such as the DBpedia or GeoNames.

Linked data has become a global data space accessible through query endpoints using SPARQL query language but it is not a global knowledge space yet. The reason is that, even though several existing datasets incorporate ontological OWL or RDFS statements, most of times these statements are just stored as RDF triples, but not fully exploited by automatic reasoning. To

date, semantic web tools such as Jena, TopBraid Composer or OWLIM only support light-weight inference, mainly implementing RDFS semantics.¹ However, it is up to data owners who publish their datasets on Linked Open Data to use these tools. At the moment, it is rarely the case for existing data sources (such as DBpedia, GeoNames and many other popular datasets) to offer ontology-based query answering when they are remotely queried using their SPARQL endpoints.

It is possible, though, for a data owner to build his/her own knowledge base as a local RDF dataset enriched with ontological constraints and rules on the target domain that he/she knows, and to equip it with local reasoning capabilities using Semantic Web technologies. Reasoning on schema constraints such as keys or functional properties can be useful for local data management but also for data linkage with external datasets.

Data linkage is a crucial task in Linked Data. In particular, it is very important to correctly decide whether two URIs refer to the same real-world entity. Most of the existing approaches consist in defining or learning metrics to compare entities based on similarities between the values of (some of) their properties (see [28] for a survey). The results returned by such numerical techniques are weighted owl:sameAs links, among which most of the ones having high weights are likely to be correct.

In contrast, like a few other works [40, 63], we propose a rule-based approach equipped with full reasoning to infer all certain same-as facts that are logically entailed from a given set of domain constraints and facts. Our approach is generic and declarative since we provide a (sound and complete) inference algorithm that takes as input any set of Datalog-like rules (with built-in arithmetic predicates) that enable to express in a uniform way several schema constraints (such as the ones denoted in OWL by owl:HasKey and owl:FunctionalProperty), transitivity and symmetry of owl:sameAs and domain-specific knowledge (e.g. to express composite or conditional keys).

In the decentralized setting of Linked Data, the main problem is to identify the data, possibly distributed over several datasets, useful for inferring owl:sameAs and owl:differentFrom facts. Compared to the approach reported in [40] that relies on a global import obtained by a breadth-first crawl of the Linked Data cloud, our approach performs a selective import while guaranteeing completeness for the inference of target owl:sameAs (and owl:differentFrom) facts.

Our contribution is twofold. First, we provide a novel Import-by-Query that alternates steps of sub-query rewriting and of tailored querying the Linked Data cloud to import data as specific as possible to infer owl:sameAs and owl:differentFrom facts. It is an extension of the well-known query-subquery algorithm for answering Datalog queries over deductive databases. Second, we

¹<http://jena.apache.org>, <http://www.topquadrant.com>, <http://www.ontotext.com/owlim>.

have shown the feasibility and usefulness of our approach for disambiguating person entities in the INA dataset, a real dataset with millions of RDF facts describing content of videos of French TV programs.

In section 2.2, we illustrate our approach by an example. In section 2.3, we discuss related work. In section 2.4, we provide the necessary background to formally state the problem studied in this chapter. The Import-by-Query is presented in Section 2.5. Experimental results are surveyed in Section 2.6, and Section 2.7 concludes the chapter.

2.2 Illustrative Example

The French National Audiovisual Institute (*Institut National de l'Audiovisuel* or INA) is devoted to maintain and update a repository of French radio and television audiovisual archives. INA data is currently available in the form of RDF facts. Figure 2.1 shows a small sample of these facts and an extract of the INA vocabulary.²

The first group of two RDF facts describes an entity `ina:vid1` of the class `ina:Video` whose title is “Le Petit Rapporteur”. The remainder three groups describe three person entities `ina:per1`, `ina:per2` and `ina:per3` all named “Jacques Martin”. The problem is that we do not know whether these entities represent the same or different persons. Some additional information is provided: `ina:per1` is known to be the presenter of a program recorded in the video `ina:vid1`, while `ina:per2` and `ina:per3` are known to have different dates of birth “1933-06-22” and “1921-09-25”, respectively.

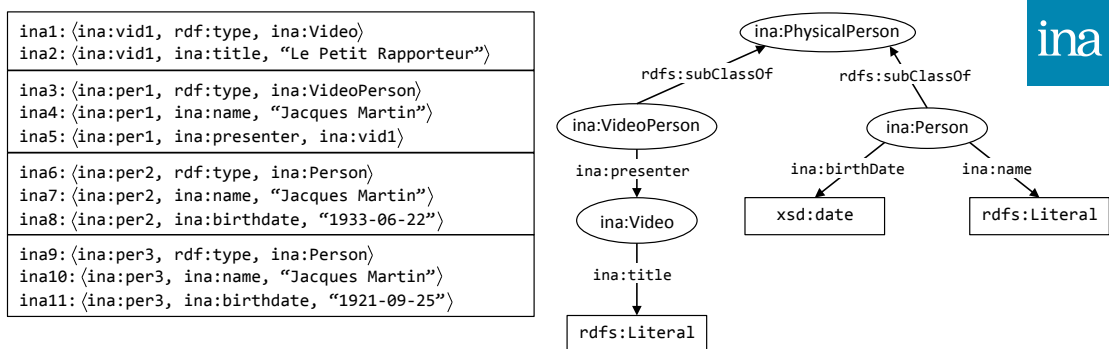


FIGURE 2.1: A sample of the INA RDF facts and an extract of the INA vocabulary.

In order to disambiguate the INA entities `ina:per1`, `ina:per2` and `ina:per3`, one can reason with knowledge about INA properties. For instance, if we agree that the property `ina:birthdate` is functional, it can be inferred that `ina:per2` and `ina:per3` are different because they have different birthdates. Also, we can agree that two persons having the same name and date of birth must be the same — or, in other words, that the properties `ina:name` and `ina:birthdate` form a key — and that two persons who have the same name and who have presented programs recorded in videos

²We have made some modifications in the INA vocabulary — e.g. translating French terms into English terms — for the sake of readability.

with the same title must be the same. This knowledge could be useful for deciding whether `ina:per1` refers to the same person as `ina:per2` or `ina:per3`, but some information is missing: the birthdate of `ina:per1` is not known, or whether `ina:per2` or `ina:per3` are presenters, and of which programs.

The above missing information can be completed thanks to external data coming from DBpedia. In Figure 2.2, we show DBpedia facts describing the DBpedia person entity `db:per1`, and a small extract of the DBpedia vocabulary. An alignment between the INA and DBpedia vocabularies — possibly computed by an ontology matching tool and validated by INA experts — also may state that the properties `ina:name` and `ina:birthdate` are, respectively, equivalent to the properties `foaf:name` and `foaf:birthdate` and that the composition of `ina:presenter` and `ina:title` is equivalent to `db:presenter`. In this way, it can be inferred that `db:per1` is the same as `ina:per1` because they have the same name and have presented a program with the same title; and also that `db:per1` is the same as `ina:per2` because they have the same name and date of birth. Thus, by transitivity of same-as, it can be inferred that `ina:per1` is the same as `ina:per2`, and, since `ina:per2` is different from `ina:per3`, then `ina:per1` is different from `ina:per3` too.

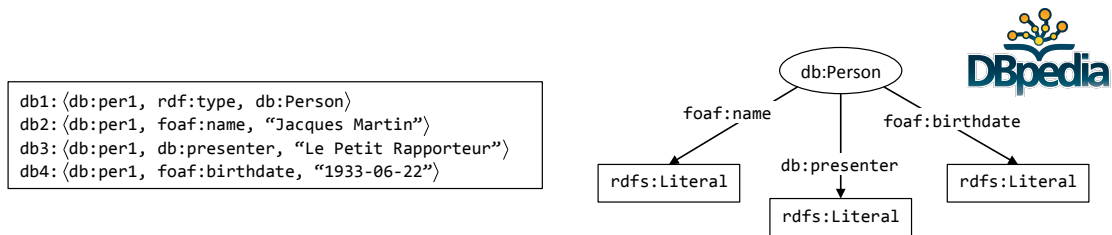


FIGURE 2.2: DBpedia facts and an extract of the DBpedia vocabulary.

Local knowledge like the knowledge about the INA properties can be expressed as rules as shown in Table 2.1. Additional knowledge like mappings between the INA and DBpedia vocabularies, and transitivity of same-as (both essential for disambiguating the INA person entities) can be translated into rules too (Table 2.1).

	IF	THEN
R1	$\langle ?x1, \text{ina:name}, ?n \rangle, \langle ?x2, \text{ina:name}, ?n \rangle, \langle ?x1, \text{ina:presenter}, ?v1 \rangle, \langle ?x2, \text{ina:presenter}, ?v2 \rangle, \langle ?v1, \text{ina:title}, ?t \rangle, \langle ?v2, \text{ina:title}, ?t \rangle$	$\langle ?x1, \text{owl:sameAs}, ?x2 \rangle$
R2	$\langle ?x1, \text{ina:name}, ?n \rangle, \langle ?x2, \text{ina:name}, ?n \rangle, \langle ?x2, \text{ina:birthdate}, ?b \rangle, \langle ?x1, \text{ina:birthdate}, ?b \rangle$	$\langle ?x1, \text{owl:sameAs}, ?x2 \rangle$
R3	$\langle ?x1, \text{ina:birthdate}, ?b1 \rangle, \langle ?x2, \text{ina:birthdate}, ?b2 \rangle, \langle ?b1, \text{notEqualTo}, ?b2 \rangle$	$\langle ?x1, \text{owl:differentFrom}, ?x2 \rangle$
R4	$\langle ?x1, \text{ina:name}, ?n \rangle, \langle ?x2, \text{foaf:name}, ?n \rangle, \langle ?x1, \text{ina:presenter}, ?v \rangle, \langle ?v, \text{ina:title}, ?t \rangle, \langle ?x2, \text{db:presenter}, ?t \rangle$	$\langle ?x1, \text{owl:sameAs}, ?x2 \rangle$
R5	$\langle ?x1, \text{ina:name}, ?n \rangle, \langle ?x2, \text{foaf:name}, ?n \rangle, \langle ?x1, \text{ina:birthdate}, ?b \rangle, \langle ?x2, \text{foaf:birthdate}, ?b \rangle$	$\langle ?x1, \text{owl:sameAs}, ?x2 \rangle$
R6	$\langle ?x1, \text{owl:sameAs}, ?x2 \rangle, \langle ?x2, \text{owl:sameAs}, ?x3 \rangle$	$\langle ?x1, \text{owl:sameAs}, ?x3 \rangle$
R7	$\langle ?x1, \text{owl:sameAs}, ?x2 \rangle, \langle ?x2, \text{owl:differentFrom}, ?x3 \rangle$	$\langle ?x1, \text{owl:differentFrom}, ?x3 \rangle$

TABLE 2.1: R1-R3 are local rules about INA properties; R4-R5 are rules involving mappings from the INA and DBpedia vocabularies; R6 translates transitivity of `owl:sameAs`, and R7 relates `owl:sameAs` and `owl:differentFrom`.

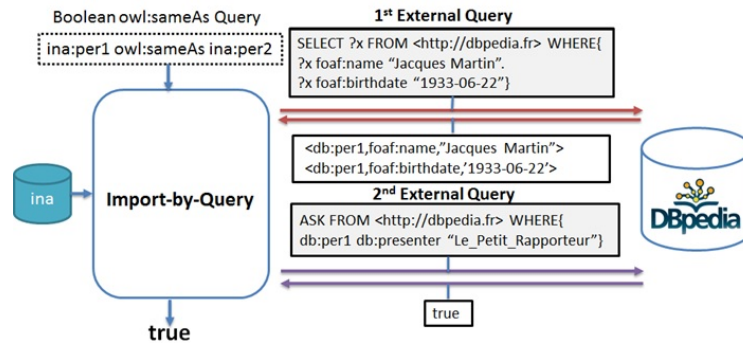


FIGURE 2.3: The resulted external sub-queries submitted to DBpedia and their returned answers

In order to avoid downloading the whole DBpedia, and, more generally, the whole Linked Open Data (something impossible in practice), our import-by-query approach (described in detail further) generates, for each target owl:sameAs fact, a sequence of external sub-queries as specific as possible so as to obtain just the missing facts. In our example, the external sub-queries generated by our algorithm for the particular query $\langle \text{ina:per1}, \text{owl:sameAs}, \text{ina:per2} \rangle$ are shown in Figure 2.3.

2.3 Related work

There exists a considerable number of systems that semi or automatically perform data interlinking (see [28] for a survey). For most of these systems, whether to link two instances or not boils down to comparing the values of the instances for all, or some of the properties they have, in such a way that the closer these values are the more likely that the instances will be linked. The quality of the results that they produce depends on the similarity and the aggregation functions that they use, and also on the threshold upon which links are asserted. The point that distinguishes the most all these numerical approaches from ours is that they are not declarative, with the inherent limitation to be difficult to extend or to adapt to new settings with new properties.

In fact, there are few existing declarative approaches, in which the comparison methods can be specified by users in the form of (XML) specifications like in Silk [69], or of rules like in LN2R [63] and Hogan et al. [40].

The Silk specifications can be seen as rules with built-in functions for both computing and aggregating degrees of similarity between property values. However, these rules are not fully exploited. In particular the possible chaining between rules is not handled by Silk, making it possibly incomplete to discover all the same-as facts that can be logically inferred. Compared to a complete forward reasoner, given a set of rules, Silk performs a single inference step and returns the same-as facts that can be inferred without rule chaining. In particular, the same-as transitivity rule, while being expressible in Silk, is not fully handled automatically: it is not

iteratively applied until a fixpoint is reached. As a result, neither Silk nor LIMES [54] would discover that `ina:p1` is the same as `ina:p2` in our illustrative scenario explained above. Closely related to Silk, the LIMES framework is conceived to optimize the number of comparisons between property values for linking two datasets. This optimization is based on estimating the similarity between a pair of values based on the similarity between each of the two values and an exemplar point in a metric space.

LN2R [63] and Hogan et al. [40] come with a complete forward-reasoner and thus guarantee to infer all the same-as facts (and all the different-from facts for LN2R) that can be logically entailed from the rules and facts that are given as input.

Contrarily to LN2R, Silk (and LIMES) and Hogan et al. consider the problem of using remote data sources to discover links. However, in contrast with our approach, the external RDF facts that are loaded to complete local data are obtained either by a global import (of a whole RDF graph of reference dataset such as DBpedia) or as an incoming data stream produced by a Linked Data crawler. This raises scalability issues that are bypassed either by light-weight incomplete reasoning (like in Silk) or by using intensive computational resources requiring clusters of machines (like in [40]). Instead, our import-by-query algorithm builds iteratively SPARQL queries for importing from external sources in Linked Data the necessary and sufficient data for resolving the link query.

A so-called import-by-query approach has been introduced in [34] for ontology reuse by specifying the limited access authorized to a remote ontology through a query interface. The associated algorithm, in contrast with ours, does not build conjunctive queries but Abox satisfiability queries used as oracles in Tableau-based reasoning.

2.4 Formal background and problem statement

We first recall the ingredients of Linked Data and we define what we call a deductive RDF dataset to capture several ontological constraints and rules for reasoning on data semantics. We end this section by formally stating the problem we consider.

2.4.1 URIs, URLs and namespaces

In Linked Data, anything (an individual, a class, a predicate, a dataset, a namespace) can be identified by a uniform resource identifier (URI as a shortcut). URIs enable interaction over the web using specific protocols such as `http`. URLs are URIs that, in addition to identifying web resources, provide the address to access them.

Namespaces provide a way to avoid term conflicts between several vocabularies used in different datasets. A name in a namespace is a string composed by a prefix, that is a URI of the namespace, and a local name. e.g. the name ‘`http://xmlns.com/foaf/0.1/knows`’ refers to the predicate having the local name “knows” within the namespace providing at the URL ‘`http://xmlns.com/foaf/0.1/`’ an RDF vocabulary devoted to describe social networks. This vocabulary can be reused in other datasets within Linked Data. A compact form is possible, for instance “foaf:knows”, as soon as the prefix “foaf” has been defined as a shortcut of the full URL by a prefix declaration: @PREFIX foaf: <http://xmlns.com/foaf/0.1/>

2.4.2 RDF datasets in Linked Data

An RDF dataset in Linked Data is defined by a URL u and a set F of RDF facts that are accessible as URL through a query endpoint. We will denote $ds(u)$ the set F of RDF facts that can be queried at the URL u .

An RDF fact is a triple $t = \langle s, p, o \rangle$ where the subject s is either a URI or a blank node, the predicate p is a URI, and the object o may be either a URI, a blank node or a literal. We will denote the vocabulary used in $ds(u)$ by $voc(u)$, i.e., the names of predicates used to declare triples in the dataset accessible at the URL u .

2.4.3 Queries over RDF datasets in Linked Data

Queries over Linked Data are SPARQL **conjunctive queries** entered through a given query endpoint accessible at a given URL. In this chapter, we use a simplified notation for SPARQL queries, and, without loss of generality, we consider that all variables are distinguished.

A query $q(u)$ asked to an RDF dataset identified by (and accessible at) the URL u is a conjunction of **triple patterns** denoted by $TP_1(v_1), \dots, TP_k(v_k)$ where each triple pattern $TP_i(v_i)$ is a triple $\langle s^v, p^v, o^v \rangle$ in which the subject s^v , the predicate p^v , or the object o^v can be variables: v_i is the set of variables appearing in the triple pattern. Variables are denoted by strings starting by ‘?’’. $TP_i(v_i)$ is a **ground** triple pattern if its set of variables v_i is empty (denoted by $TP_i()$). A ground triple pattern corresponds to a RDF fact. A **boolean query** is a conjunction of ground triple patterns. For instance, the first SPARQL query of Figure 2.3 in the previous section become in this simplified notation:

$q(\text{http://dbpedia.fr}): \langle \text{dbpedia:perl}, \text{dbpedia:presenter}, \text{“Le.Petit.Rapporteur”} \rangle$.

The evaluation of a query $q(u) : TP_1(v_1), \dots, TP_k(v_k)$ over the dataset $ds(u)$ consists in finding substitutions θ assigning the variables in $\bigcup_{i \in [1..k]} v_i$ to constants (i.e., identifiers or literals) such that $TP_1(\theta.v_1), \dots, TP_k(\theta.v_k)$ are RDF facts in the dataset.

The corresponding answer is equally defined as the tuple of constants assigned by θ to the variables or as the set of corresponding RDF facts $TP_1(\theta.v_1), \dots, TP_k(\theta.v_k)$ that will be denoted by $\theta.q(u)$. In the remainder of the chapter, we will adopt the latter definition. The answer set of the query $q(u)$ against the dataset $ds(u) = F$ is thus defined as:

$$Answer(q(u), F) = \bigcup_{\{\theta | \theta.q(u) \subseteq F\}} \theta.q(u)$$

For a **boolean query** $q(u)$, either the answer set is not empty and we will say that the query is evaluated to **true**, or it is empty and we will say that it evaluated to **false**.

For a query $q(u)$ to have a chance to get an answer when evaluated over the dataset $ds(u)$, it must be **compatible with** the vocabulary used in this dataset, i.e., (a) the predicates appearing in the triple patterns of $q(u)$ must belong to the set $voc(u)$ of predicates known to occur in $ds(u)$, (b) the URIs appearing as constants in the triple patterns of $q(u)$ must have u as prefix.

In accordance with SPARQL queries allowing different FROM clauses, a conjunctive query can in fact specify several entry points u_1, \dots, u_n of datasets over which to be evaluated. We will denote such a query $q(u_1, \dots, u_n)$. The above definitions of answers and compatibility can be generalized appropriately by replacing the dataset $ds(u)$ by the union $\bigcup_{i \in [1..n]} ds(u_i)$ of the specified datasets.

2.4.4 Deductive RDF datasets

In order to capture in a uniform way semantic constraints that can be declared on top of a given RDF dataset, but also possibly mappings between local predicates and external predicates within the vocabulary of other datasets, and domain knowledge provided by domain experts, we consider that RDF datasets can be enriched with Datalog rules of the form: $Cond_r \Rightarrow Conc_r$, in which the condition $Cond_r$ is a conjunction of triple patterns (i.e., a conjunctive query) and the conclusion $Conc_r$ is a triple pattern. We consider **safe** rules, i.e., such that all the variables in the conclusion are also in the condition. Datalog rules on top of RDFS facts capture most of the OWL constraints used in practice, while guaranteeing a polynomial data complexity for reasoning and query answering.

A deductive RDF dataset $dds(u)$ accessible at the URL u is thus a local knowledge base $\langle F, R \rangle$ made of a set of RDF facts F and a set R of rules. The application of rules enables to infer new facts that are logically entailed from $F \cup R$. A rule r can be applied to F if there exists a substitution θ such that $\theta.Cond_r \subseteq F$ and the result of the rule application is $F \cup \{\theta.Conc_r\}$. These new facts can in turn trigger rules and infer additional facts. The point is that from a finite

set of facts F and a finite set of safe rules R , the set of facts that can be inferred is finite and can be computed as the least fixed point of immediate consequence operator T_R defined as follows:

$$T_R(F) = F \cup \bigcup_{r \in R} \{\theta.Conc_r \mid \theta.Cond_r \subseteq F\}$$

Let $F_0 = F$, and for every $i \geq 0$, let $F_{i+1} = T_R(F_i)$. There exists a unique least fixed point F_n (that will be denote by $SAT(F, R)$) such that for every $k \geq n$ $F_k = T_R(F_n)$, i.e., there exists a step in the iterative application of the immediate consequence operator for which no new fact is inferred.

The evaluation of a query $q(u) : TP_1(v_1), \dots, TP_k(v_k)$ over a deductive dataset $dds(u)$ consists in finding substitutions θ such that the facts $TP_1(\theta.v_1), \dots, TP_k(\theta.v_k)$ can be **inferred** from the deductive dataset, or equivalently belong to the result $SAT(F, R)$ of the facts that can be inferred from F and R :

$$Answer(q(u), \langle F, R \rangle) = Answer(q(u), SAT(F, R))$$

Therefore, a **boolean query** $q(u)$ is evaluated to **true** if and only if $q(u) \in SAT(F, R)$, i.e., if and only if $\langle F, R \rangle \vdash q(u)$, where the symbol \vdash is the standard notation for logical inference.

Within the vocabulary of a deductive dataset, we distinguish the extensional predicates (EDB predicates for short) that appear in the triplets of the dataset F , from the intentional predicates (IDB predicates) that appear in conclusion of some rules in R . Like in deductive databases, and without loss of generality (i.e., by possibly renaming predicates and adding rules), we suppose that these two sets are disjoint. We will denote ODB predicates the *external* predicates (i.e., defined in a different namespace than the considered deductive dataset) that possibly appear in the dataset or in the rules. These predicates are the core of Linked Data in which a good practice is to re-use existing reference vocabularies. We suppose (again, without loss of generality) that the set of ODB predicates is disjoint from the set of IDB predicates (but not necessarily from the set of EDB predicates).

2.4.5 Problem statement

Given a deductive dataset $dds(u) = \langle F, R \rangle$, and a boolean query $q(u)$ the local evaluation of which gives an empty answer set (i.e., $\langle F, R \rangle \not\vdash q(u)$), we aim to construct a set of external queries $q_1(u_1), \dots, q_k(u_k)$ for which we can guarantee that the subsets of external facts resulting

from their evaluation over the (possibly huge) external datasets are sufficient to answer the initial query, i.e., more formally:

$$\begin{aligned} \langle F \cup_{i \in [1..k]} \text{Answer}(q_i(u_i), ds(u_i)), R \rangle \vdash q(u) \\ \text{iff } \langle F \cup_{i \in [1..k]} ds(u_i), R \rangle \vdash q(u) \end{aligned}$$

The more specific the external queries are, the less external facts have to be added and stored to the local dataset and therefore the more interesting a proposed approach is to solve this problem.

2.5 The iterative Import-by-Query Algorithm

We now describe the general algorithm that we have designed and implemented for solving the problem stated above.

Given an input boolean same-as query q , a deductive dataset $\langle F, R \rangle$, and a set \bar{u} of query entry points to external datasets, Import-by-Query iteratively alternates steps of sub-query rewriting based on backward chaining and of external query evaluation.

Each sub-query rewriting step is realized by an adaptation of the Query-Subquery algorithm [3, 68] that is a backward chaining method used in deductive databases for evaluating Datalog programs. We will describe in detail (see Section 2.5.1) this adaptation that we have called Query-External-Subquery (QESQ for short). QESQ either succeeds in proving locally the goal q (i.e., using F and R just like Query-Subquery) and then the process is stopped and the result returned by Import-by-Query is **true**, or it produces a set $\{q_1(\bar{u}_1), \dots, q_k(\bar{u}_k)\}$ of external queries compatible with the vocabulary of the corresponding external datasets, the evaluation of them is likely to bring missing facts to F for proving the goal q using R . This set may be empty: in this case, the process is stopped and the result returned by Import-by-Query is **false**.

Each evaluation step simply consists in choosing one of the external query $q_i(\bar{u}_i)$ produced by the sub-query rewriting step and to submit it to Linked Data through the specified query entry points. The result is either an empty set (negative result) or a set of external facts (positive result) that can be added to the current local dataset. In both case, the result is memorized in an associated answer table for the sub-query $q_i(\bar{u}_i)$ that will be thus marked as a already processed subgoal for which the (positive or negative) result is known and can be directly exploited later on. If the result is positive, a new iteration of Import-by-Query is started on the same input except for the set of facts F that is enriched with the facts obtained as the result of the evaluation of the external query $q_i(\bar{u}_i)$. If the result is negative, another external query $q_j(\bar{u}_j)$ in the set produced by the current call to QESQ is evaluated. If the evaluation of all the external queries in the set

returns 'false', then the process is stopped and the result returned by Import-by-Query on q is **false**.

We now focus on the algorithm QESQ that is the core of the sub-query rewriting step to explain in more detail how it works.

2.5.1 The QESQ algorithm

QESQ is an adaptation of the Query-Subquery algorithm that is a set-oriented memoing backward chaining method [39]. Besides being complete with a guaranteed termination even for recursive sets of rules (in contrast with the method underlying Prolog), the memoing methods can also increase dramatically efficiency by reusing previously computed answers. For this memoing purpose, QESQ handles two triple tables $answer_p$ and $goal_p$ (empty at the beginning) for each ODB and IDB predicate p , for storing in $goal_p$ the different subgoals on predicate p that are under process, and in $answer_p$ the answers that have been computed for any atomic query involving p .

QESQ is a recursive algorithm that starts with an input boolean atomic query $q : \langle s, p, o \rangle$ and treats it as the goal to solve using the input dataset F and the input set of rules R . For this, if it is not trivially solved by a direct evaluation over the dataset F (i.e., if $\langle s, p, o \rangle \notin F$), it would try to rewrite the current goal into a list of subgoals obtained by unfolding a rule r whose conclusion $Conc_r$ can be matched with the current goal, i.e., if there exists a substitution θ such that $\theta.Conc_r = \langle s, p, o \rangle$. The initial goal is then replaced by the list of subgoals $\theta.TP_1(v_1), \dots, \theta.TP_k(v_k)$ composed of the partial instantiation by θ of the triple patterns $TP_1(v_1), \dots, TP_k(v_k)$ in the condition of r . If there is no such a rule, QESQ stops and returns 'false'. Otherwise, it is recursively called on the new list of subgoals.

A general recursive call of QESQ applies then to a list $SG = [g_1, \dots, g_k]$ where each subgoal g_i is a triple pattern $\langle s_i^v, p_i, o_i^v \rangle$. It differs in the treatment of each subgoal depending on whether the predicate p_i is an extensional (EDB), external (ODB) or intentional (IDB) predicate. By a slight abuse of notation, it will return as output either **TRUE** or **FALSE** (if it has enough local information to infer a result to the input boolean query), or a non empty set of external queries (compatible with the vocabulary of the given external datasets). A recursive call of QESQ can be summarized as follows:

1. **IF** there exists $g_i = \langle s_i^v, p_i, o_i^v \rangle$ in the subgoals such that the predicate p_i is an EDB predicate, or such that g_i has been handled before (i.e., g_i can be matched to a triple in $goal_{p_i}$)
THEN evaluate the atomic query $g_i = \langle s_i^v, p_i, o_i^v \rangle$ over F , or over $answer_{p_i}$:
 - **IF** there is no answer and p_i is not an ODB predicate **THEN** return **FALSE**

- **ELSE** for each answer, let θ be the corresponding substitution (it can be empty if the query g_i is boolean), i.e., such that $\theta.g_i \in F$, or $\theta.g_i \in answer_{p_i}$:
 - Let $NewSG$ be the list of new subgoals obtained from SG by removing g_i and by replacing the other subgoals g_j by $\theta.g_j$.
 - **IF** $NewSG$ is empty **THEN** return **TRUE**
 - **ELSE** trigger a new recursive call of QESQ on $NewSG$.
2. **IF** there exist subgoals not seen before with ODB predicates **THEN**:
 Let q_{ext} be their conjunction.
IF q_{ext} is compatible with the vocabulary of \bar{u} (the given entry points to external datasets) **THEN**:
- Let $NewSG$ be the list of new subgoals obtained from SG by removing all subgoals in q_{ext}
 - **IF** $NewSG$ is empty **THEN** return q_{ext} **ELSE**
 - for each subgoal $\langle s^v, p, o^v \rangle$ in q_{ext} , add it to the table $goal_p$
 - trigger a new recursive call of QESQ on $NewSG$:
 - **IF** it does not return **TRUE** or **FALSE** **THEN** return $Q_{ext} \cup q_{ext}$ where Q_{ext} is the result of QESQ($NewSG$)
3. **ELSE** (all the subgoals have IDB predicates and they have not been seen before)
- let $g = \langle s^v, p, o^v \rangle$ be the first subgoal in SG and let R_g be the set of rules whose conclusion can be matched with g
 - **IF** $R_g = \emptyset$ **THEN** return **FALSE**
 - **ELSE**:
 - success \leftarrow false ; $SetOfQ_{ext} \leftarrow \emptyset$
 - **WHILE** $R_g \neq \emptyset$ and \neg success
 - * choose r in R_g and let θ the substitution s.t. $\theta.Conc_r = g$,
 - * let $NewSG$ be the list of new subgoals obtained from SG by replacing g with $\theta.TP_1(v_1), \dots, \theta.TP_k(v_k)$ where $TP_1(v_1), \dots, TP_k(v_k)$ are the conditions of r , and by replacing the other subgoals g_j with $\theta.g_j$,
 - * add g to the table $goal_p$,
 - * remove r from R_g ,
 - * trigger a new recursive call of QESQ on $NewSG$,
 - * **IF** it returns **TRUE** **THEN** success \leftarrow true
 - * **ELSE IF** it does not return **FALSE** **THEN**
 $SetOfQ_{ext} \leftarrow SetOfQ_{ext} \cup Q_{ext}$ where Q_{ext} is the result of QESQ($NewSG$)
 - **IF** success **THEN** return **TRUE** **ELSE IF** $SetOfQ_{ext} = \emptyset$ **THEN** return **FALSE** **ELSE** return $SetOfQ_{ext}$

The termination is guaranteed by the same memoing technique as Query-Subquery (i.e., in the above algorithm, by handling the tables $goal_p$ and $answer_p$ for each ODB and IDB predicate). If the result returned by QESQ is **TRUE** or **FALSE**, it did not pass through the case 2 and behaved exactly like Query-Subquery. By the soundness and completeness of Query-Subquery [68], if the output is **TRUE**, it means that the boolean input query is entailed by the local facts and the rules, and if the output is **FALSE**, it means that all the rules likely to infer it are blocked by

lack of local facts to instantiate the EDB predicates in rule conditions and thus no external fact would help. If the result returned by *QESQ* is neither **TRUE** nor **FALSE**, it ends up with a non empty set of external queries that correspond to all the possible branches of reasoning that have not been blocked. Once an answer is returned for a particular external query, the corresponding branch of reasoning will be ended (either by true or false) by the next call of *QESQ* that will behave on this branch just like *Query-Subquery*. If it ends with **TRUE**, *Query-by-Import* stops (with success), and if it ends with **FALSE**, *Query-by-Import* chooses another external query to evaluate in the waiting list. It stops with failure (returning **FALSE**) when the list of external queries is empty.

This shows that the *Query-by-Import* algorithm is sound and complete for solving the problem stated in Section 2.4.5.

2.5.2 Combining forward and backward chaining

Like any backward chaining method, *Import-by-Query* (and its main component *QESQ*) restarts from scratch for each new goal it tries to solve, even if the facts and the rules remain unchanged. The intermediate subgoals generated and handled by *QESQ* can be simplified if the input rules are replaced by their (partial) instantiations obtained by the propagation of the facts into (the conditions of) the rules.

Fact propagation is a forward chaining method used in inference engines such as *RETE* [29] for rule-based systems. It avoids redundant evaluation of same conditions appearing in several rules by memorizing, for each fact f , which condition it satisfies in which rule (possibly already partially instantiated by facts previously propagated), and the corresponding variable substitution that is then applied to all the remaining conditions of the rules.

In our setting, we perform fact propagation as a pre-processing step of the import-by-query algorithm, by computing at the same time the set $SAT(F, R)$ of facts that can be inferred locally, and the set $PI(F, R)$ of partial instantiations of the rules in R . This forward reasoning step can be summarized as follows, where $SAT(F, R)$ is initialized as F and $PI(F, R)$ is initialized as R :

- **FOR** each f in $SAT(F, R)$
 - FOR** each rule $Cond_r \Rightarrow Conc_r$ in $PI(F, R)$ having a condition c that can be matched with f , i.e., there exists θ such that $\theta.c = f$
 - * **IF** c is the only condition in $Cond_r$ **THEN** add $\theta.Conc_r$ to $SAT(F, R)$
 - * **ELSE** add to $PI(F, R)$ the rule obtained from $\theta.Conc_r \Rightarrow \theta.Conc_r$ by removing the condition $\theta.c$ (that is satisfied by the fact f).

- Remove from $PI(F, R)$ those rules whose condition contains EDB predicates that are not ODB predicates (and thus cannot be satisfied by local facts).
- **RETURN** $\langle SAT(F, R), PI(F, R) \rangle$

Each partially instantiated rule r_i returned in $PI(F, R)$ is issued from an input rule r in which some conditions have been matched to facts f_1, \dots, f_k that have been inferred before (and added to $SAT(F, R)$), and thus enables to infer the same conclusion as the input rule r on any set of facts including f_1, \dots, f_k .

The result $SAT(F, R) \cup PI(F, R)$ is then logically equivalent to the input deductive dataset $F \cup R$ for inferring facts on IDB predicates from the union of F and a set OF of external facts (with ODB predicates), i.e. for every fact f an external set of facts OF :

$$\langle F \cup OF, R \rangle \vdash f \text{ iff } \langle SAT(F, R) \cup OF, PI(F, R) \rangle \vdash f$$

Therefore, it can be equivalently used for proving goals by checking whether they belong to $SAT(F, R)$, or for rewriting goals by applying $QESQ$ to the $PI(F, R)$ (instead of the original R).

Compared to other forward chaining methods (such as semi-naive bottom-up evaluation in deductive databases), fact propagation may require an additional cost (in time and space) that is likely to be amortized very fast, either for answering boolean queries locally (by checking if they belong to $SAT(F, R)$), or for rewriting those that cannot be answered locally, by applying $QESQ$ on $PI(F, R)$. Note that since facts and rules are parameters, it is possible to propagate only a subset local facts into the set of rules.

2.6 Experiments

We have conducted a series of experiments on a real deductive dataset composed of ~ 6 millions of RDF facts from the INA dataset, and 35 rules. The facts describe videos of French TV programs including information about their duration, title, subject and publishing date, and of French personalities (actors, singers, presenters, etc.) including their name, and additional, but maybe incomplete, information about their date of birth and nationality, and their roles in the videos (producer, presenter, participant, etc.).

As hinted in our illustrative example of Section 2.2, the INA dataset contains many homonymous persons, i.e. individuals of the class `ina:PhysicalPerson` that have the same value for the property `ina:name` (see Figure 2.1 in Section 2.2). INA experts, however, do not know whether these entities represent the same or different persons. There exist two subclasses of `ina:PhysicalPerson`,

namely, `ina:Person` and `ina:VideoPerson`. The class `ina:Person` is devoted to representing French personalities, whereas `ina:VideoPerson` is used for identifying entities of persons that play a role for a video. Although `ina:Person` entities come with additional, but maybe not complete, information, `ina:VideoPerson` entities just come with information about their names and videos in which they play a role. INA experts are particularly interested in disambiguating individuals within `ina:Person`, as well as linking these individuals to the ones of `ina:VideoPerson`.

Concerning the second component of the deductive dataset, the 35 rules, these rules capture available knowledge in the domain (e.g. functional properties and keys declared as schema constraints, and domain-specific rules provided by INA experts), mappings between the INA and DBpedia vocabularies, and general properties on the predicates `owl:sameAs` and `owl:differentFrom`. One rule expresses the transitivity of `owl:sameAs` and another rule states the relation between `owl:sameAs` and `owl:differentFrom` (the fact that if x_1 is the same as x_2 and x_2 is different from x_3 then x_1 is different from x_3). Interestingly, these last two rules happen to be recursive. 11 of the 35 rules enable to conclude different-from facts, some of them requiring built-in predicates such as `not-equal`, `less-or-equal`, `greater-or-equal`, `minus`, `sum`, etc. In Section 2.2, we showed 7 of the 35 rules (Table 2.1) but the complete set of rules can be found in <http://bit.ly/1slxLHj>.

It is worth noting that the 35 rules could be extended or modified without changing the algorithmic machinery of our approach.

Built-in functions allow us to be tolerant to slight differences when comparing literal values. For example, we can consider a rule like the one shown in Table 2.2 which states that two persons with the same birthdate and deathdate, and “similar” names, must be the same person. Whenever our algorithm encounters a ground triple that match $\langle ?n1, \text{built-in:name-similar}, ?n2 \rangle$, it will replace it with a boolean value returned by a built-in function which checks whether the similarity of the two names — for a specific similarity metric — is above a given threshold. In our experiments we considered edit distance and 0.99 as threshold.

	IF	THEN
R8	$\langle ?x1, \text{ina:name}, ?n1 \rangle, \langle ?x2, \text{foaf:name}, ?n2 \rangle, \langle ?n1, \text{built-in:name-similar}, ?n2 \rangle, \langle ?x1, \text{ina:birthdate}, ?b \rangle, \langle ?x2, \text{foaf:birthdate}, ?b \rangle$	$\langle ?x1, \text{owl:sameAs}, ?x2 \rangle$

TABLE 2.2: A rule for dealing with similarities between names.

2.6.1 Experimental Goals and Set-Up

The goal of our experiments was threefold: (1) to demonstrate that external information available in Linked Open Data can be useful to infer `owl:sameAs` and `owl:differentFrom` facts within INA referenced persons, and, thus, disambiguate local homonyms; (2) to assess the gain in

reduced imported facts of our import-by-query approach compared to approaches based on forward reasoning only; and (3) to evaluate the runtime of our import-by-query algorithm and the possible amortized gain when fact propagation is performed beforehand.

The external datasets from Linked Open Data with which the INA vocabulary shares terms are DBpedia.org and DBpedia.fr. The baseline for evaluating our two first goals is a set of 0.5 million external facts obtained by downloading from DBpedia.org and DBpedia.fr (using their SPARQL endpoints) all the facts about entities having the same name as one of the homonyms in the INA dataset. We applied a preprocessing step on the original INA dataset to keep only the facts on predicates appearing in the rules conditions. The resulting dataset contains almost 1,150 millions of RDF facts will be the INA dataset referred to henceforth.

Our algorithms have been implemented in SWI-Prolog. All the evaluations were done on a machine with an Intel i7 Quad-core processor and 6 GB of memory.

2.6.2 Experimental Results

For evaluating our first experimental goal, we applied (using our forward reasoner) the set of 35 rules to (a) the INA dataset only, and (b) the union of the INA dataset with the baseline external facts, and we compared the number of owl:sameAs and owl:differentFrom facts obtained on INA homonyms that we obtained.

The rules applied to the INA dataset only allowed to infer 110 facts (2 owl:sameAs facts and 108 owl:differentFrom facts) compared to the 14,648 facts (4,884 owl:sameAs facts and 9,764 owl:differentFrom facts) inferred when the external facts were added to the process. This clearly demonstrates the benefit of using external information coming from Linked Open Data for local disambiguation.

The resulting 14,648 facts are guaranteed to be correct under the assumption that both the rules and data are correct. Although the correctness of the rules was guaranteed by INA experts, DBpedia and INA datasets may contain noisy data. Thus, we asked INA experts to evaluate a random sample of 500 facts, and all of them resulted to be correct.³

It is worth noting that the rule expressing transitivity of owl:sameAs is crucial to infer all the owl:sameAs facts that cannot be inferred locally, and more generally that full reasoning is important to discover owl:sameAs and owl:differentFrom facts. To demonstrate this, we applied Silk to the same two datasets (the INA dataset only, and the union of the INA dataset with the baseline external facts). For doing so, we first translated our rules into the Silk specification language. It is not possible, however, to translate into Silk our rules concluding on owl:differentFrom atoms. Thus, we focused on the rules leading to owl:sameAs inference. Among the 4,884 owl:sameAs

³This sample size ensures with 95% confidence that the proportion of correct facts obtained is greater than 0.99.

facts discovered by our full forward reasoner, Silk only discovered 88 of them, i.e. less than 3% of the total.

For evaluating our second experimental goal, we took as reference boolean queries the above sample of 500 owl:sameAs and owl:differentFrom facts, and we applied our import-by-query algorithm to each of these boolean queries. The number of external facts imported by our algorithm for all boolean queries was 6,417, which makes, on average, 13 imported facts per boolean query. In contrast, the total number of baseline external facts needed to conclude the boolean queries with the forward reasoner was much higher (500,000). This demonstrates that our import-by-query algorithm reduces drastically the number of imported facts needed for disambiguating local data. The results of this experiment are summarized in Table 2.3.

	Import-by-Query	Forward Reasoner
Number of imported facts	6,417 facts (13 facts on average per boolean query)	500,000 facts

TABLE 2.3: Gain in reduced imported facts.

Time to answer a boolean query <i>after</i> fact propagation	7 seconds
Time to answer a boolean query <i>without</i> fact propagation	186 seconds
Time to propagate facts (done once for all queries)	191 seconds
Gain of doing fact propagation beforehand for answering the 500 reference queries using import-by-query	96%

TABLE 2.4: Gain in runtime.

We now report on the runtime evaluation. Our import-by-query algorithm requires 3 iterations on average — it successively outputs and evaluates 3 external sub-queries (each of them being produced by calling QESQ) — before termination. It takes on average 186 seconds per boolean query when applied to the initial set of rules and the local dataset. This drops to 7 seconds when it is applied to the partially instantiated rules obtained by fact propagation beforehand.

Concerning fact propagation, we propagated all facts involving properties of class ina:Person. It took 191 seconds but this is done once for all. Its cost is thus amortized very fast, as shown in Table 2.4 that reports the cumulative runtime required for applying the combined approach on all the reference owl:sameAs and owl:differentFrom facts.

2.7 Conclusion

In this chapter, we have proposed a novel approach for data linkage based on reasoning that is adapted to the decentralized nature of the Linked Data cloud. This approach builds on the formal and algorithmic background of answering Datalog queries over deductive databases, that we have extended to handle external rewriting when local answers cannot be obtained. In contrast with existing rule-based approaches for data linkage [40, 63] based on forward reasoning to infer

same-as facts, Import-by-Query is a backward chaining algorithm that imports *on demand* only external facts useful to infer target same-as facts handled as boolean queries. Our experiments have shown that this approach is feasible and reduces the number of facts needed to be imported. Compared to the depth-first approach sketched in [2] for distributed Query-Subquery, our QESQ algorithm generates external rewriting in a breadth-first way.

Performing fact propagation beforehand in order to apply Import-by-Query to a set of more specific rules than the original ones is an optimization close to the ones proposed in QueryPIE [67] for efficient backward reasoning on very large deductive datasets. One important difference, though, is that in the QueryPIE setting, the problem of handling recursive rules can be fully delegated to forward reasoning because all the facts are given and the recursive rules concern a well identified subset of them (so called terminological facts). In the decentralized setting that we consider, and also because we allow domain-specific rules with no restriction on the facts to which they can apply, Import-by-Query has to handle recursive rules and the termination issues they raise for backward reasoning. Another major difference is that while QueryPIE is designed for answering queries only, Import-by-Query performs query rewriting if no local answer can be obtained from the input deductive dataset.

Compared to the many recent works on ontology-based data access initiated by [24], in which query rewriting is done independently of the data, we have designed an *hybrid* approach that alternates (external) query rewriting and (local) query answering. To deal with possibly recursive Datalog queries, we distinguish IDB predicates that can be locally rewritten, from ODB predicates that can be externally rewritten. With this limitation, our approach is able to iteratively rewrite (boolean) Datalog queries on IDB predicates. We plan to look into this hybrid approach further, and in particular to deal with ontological constraints expressible in Datalog^+ [23] allowing existential variables in conclusion.

The interest of our rule-based approach is that it is generic and declarative. New rules can be added without changing the algorithmic machinery, for which not only we guarantee soundness and completeness but we also have shown experimentally that it is feasible and useful in practice. At the moment the rules that we consider are certain, as they express logical domain constraints and equivalence mappings between classes and properties. As a result, the same-as facts that they allow to infer are guaranteed to be correct (under the assumption that the input data does not contain erroneous facts). This is crucial to get automatically same-as facts that are certain, in particular when the goal of discovering same-as links is data fusion, i.e., replacement of two URIs by a single one in all the relevant facts. Another added-value to get certain same-as and different-from facts is to find noisy data thanks to contradictions. However, in many cases, domain knowledge is not 100% sure such as pseudo-keys [11] and probabilistic mappings [65]. Data itself may be uncertain due to trust and reputation judgements towards data sources [10]. Handling uncertain domain knowledge (such as pseudo-keys or constraints with exceptions)

should enable to discover more same-as facts that may be true even if they are inferred with some uncertainty. They will just have to be validated by experts. In the following chapter, we extend our rule-based approach to model any kind of data and rules uncertainty as probabilities within the framework of Probabilistic Datalog [30].

Chapter 3

Reasoning With Uncertainty For Data Linkage

3.1 Introduction

Considering only certain facts and rules for data linkage, as we did in the previous chapter, limits the possible number of inferred links. It is due to the fact that only a limited amount of certain facts and rules that are needed to infer these links are available. In practice, the majority of datasets available in Linked Data contains inaccurate data [4, 71]: produced by annotation extraction tools, outdated information or even wrong data resulted from malicious behaviors (e.g. injected by data spammers [36]). Even though there exists efforts for data cleansing [20, 44, 70], The huge amount of data available in Linked Data makes the task of verifying and guaranteeing that all of them are certain is very hard in practice. Analogously, rules often come with uncertainty weights produced by automatic rule generation tools. Even the domain experts' rules are frequently given in the form "In many cases if ... then ...". It is therefore inevitable to take uncertainty into account while solving the problem of data linkage.

In the same time, keeping a connection with the logical reasoning is useful for tracking the derivation of the inferred facts. This allows the domain experts to determine easily the facts or rules that lead to infer incorrect same-as or different-from facts and update the corresponding uncertainty values if needed.

Our contribution in this chapter is twofold. First, we propose an extension of our rule-based logical approach that accepts facts and rules attached with uncertainty values. This extension is based on Probabilistic Datalog [31] which combines Datalog and probability theory together to do uncertain inference. Second, we show how to interpret the uncertainty values calculated for

the inferred facts and how to benefit from the logical inference to return the best results to the user.

We start by explaining how to model uncertainty values for solving the problem of data linkage in Section 3.3. In Section 3.4 we adapt the forward reasoning algorithms presented in Chapter 2 to deal with uncertainty. Then, a discussion about the complexity of the algorithm is presented in Section 3.5. The possible application of the produced uncertainty values to tune the parameters of the reasoning algorithms is presented in Section 3.6. We experiment and evaluate our approach in Section 3.7. Finally we restate the problem of the import by query studied in Chapter 2 in this probabilistic settings in 3.8 and we present the probabilistic import-by-query algorithm in Section 3.8.1.

3.2 Sources of Uncertainty

In this section we give examples of the possible different sources of uncertain knowledge that exist in practice. These sources may provide uncertain facts or uncertain rules attached with weights that can be used later in our approach:

- On the rule level:
 - Pseudo-Key generator tools [12, 58]: These tools generate keys that may have some exceptions in a given dataset. For example in a dataset that describe people $\langle \text{name}, \text{birthYear} \rangle$ can be considered as a pseudo-key. Each generated pseudo-key is thus attached with a weight calculated based on the number of the exceptions for that pseudo-key in the dataset. These weights can be attached to the uncertain rules that translate these pseudo-keys.
 - Ontology alignment tools [25]: Most of the existing ontology alignment tools attach weights to the alignments they generate. These weights can be used as the probabilistic weights for the rules that translate these alignments.
 - Domain experts: Domain experts may provide rules in the form (in many cases, IF ... THEN ...) with qualitative assignment. These qualitative assignment can be translated into numerical assignments to obtain rules assigned with weights. An example of this translation is given in our experiments in section 3.7.
- On the data level:
 - Trust and reputation tools [41]: which may assign to different data sources trust values. These values can be attached to the data imported from these source. For example, a trust tool may attach to DBpedia the trust value 0.9, then all facts downloaded from DBpedia can be assigned with a probabilistic weight 0.9

- String similarity tools [33]: Many linking rules disambiguate entities if they share exactly the same values for certain properties (e.g. rules that translate keys, pseudo-key). However, in practice the exact matching between the values is not always guaranteed especially for string values (e.g. shortened names, names which may differ because of the cases of the letters, the existence of some special characters ...). Uncertain facts that unify these different but similar string values can be used to overcome this problem, and the similarity value between the string values can be used as a weight for these facts. An example of these uncertain facts is given in the following illustrative example.

3.3 Modeling Uncertainty in Linked Data

Even-though we are aware that facts and rules may come with uncertainty weights from various sources in different natures, we took the decision to model uncertainty in linked data as probabilistic weights and we extended our rule based approach presented so far in Chapter 2 to deal with these probabilistic weights using Probabilistic Datalog. In this section we start by giving an illustrative example that introduces this model and extension then we give the formal background for the new probabilistic setting.

3.3.1 Illustrative Example

Let us consider an extract of the INA and DBpedia datasets that is depicted in Figure 3.1. It describes two person homonyms from the INA dataset that share the same name “Jacques Martin”. Only the birth year of `ina:per1` is known and only the death year of `ina:per2` is given. The figure also describes a DBpedia entity with the name “Jacques Martin (homme politique)” for which both the birth year and the death year are known. Similar to the illustrative example presented before in Section 2.2, we are interested in disambiguating the two INA entities `ina:per1` and `ina:per2` with the help of the external DBpedia entity `db:per1`.



1.0	<code><ina1: ina:per1, ina:name, "Jacques Martin" ></code>	
1.0	<code><ina2: ina:per1, ina:birthYear, "1933" ></code>	
1.0	<code><ina3: ina:per2, ina:name, "Jacques Martin" ></code>	
1.0	<code><ina4: ina:per2, ina:deathYear, "2007" ></code>	
1.0	<code><db1: dbpedia:per1, dbpedia:name, "Jacques Martin (homme politique)" ></code>	
1.0	<code><db2: dbpedia:per1, dbpedia:birthYear, "1933" ></code>	
1.0	<code><db3: dbpedia:per1, dbpedia:deathYear, "2005" ></code>	

FIGURE 3.1: A sample of certain facts from the INA and DBpedia datasets

To solve this problem, one may apply the set of rules in Table 3.1 over the set of facts. However, the rule r_1 which considers that two persons are the same if they share the same name and the

same birth year is not 100% certain. For example, there exist in DBpedia two different athletes with the name “Michael Jackson” that are both born in the same year 1969. We chose to model the uncertainty of the rules by associating probabilistic weights to them (e.g. the probabilistic weight 0.7 can be associated to the rule $r1$). These probabilistic weights are interpreted as the probability that the conclusion of the corresponding rule is true if the condition is true (i.e. instantiated by facts)

Rules $r1$, $r2$ and $r3$ in Table 3.1 cannot be applied directly over the set of facts to disambiguate the entities. Although the name of the two INA homonyms is “similar” to the name of the DBpedia entity, applying these rules, however, requires an exact string matching between the two names. To insure this matching, the uncertain fact that attaches to the DBpedia entity `dbpedia:per1` the name “Jacques Martin” i.e.:

0.6 db4:⟨dbpedia:per1, dbpedia:name “Jacques Martin”⟩

can be added to the set of facts. Similarly to the uncertain rules, the uncertainty of this fact can be modeled by attaching a probabilistic weight to it. It gives the probability that the corresponding fact is true. The similarity value 0.6 between the two names:

$sim(\text{“Jacques Martin”}, \text{“Jacques Martin (homme politique)”})$, returned by the normalized edit distance algorithm can be used as a weight for this uncertain fact.

Prob. Weights	Id	IF	THEN
0.7	r1	⟨?x1, ina:name, ?n⟩, ⟨?x2, dbpedia:name, ?n⟩, ⟨?x1, ina:birthYear, ?y⟩, ⟨?x2, dbpedia:birthYear, ?y⟩	⟨?x1, owl:sameAs, ?x2⟩
1.0	r2	⟨?x1, ina:name, ?n⟩, ⟨?x2, dbpedia:name, ?n⟩, ⟨?x1, ina:deathYear, ?y1⟩, ⟨?x2, dbpedia:deathYear, ?y2⟩ ⟨?y1, notEqual, ?y2⟩	⟨?x1, owl:differentFrom, ?x2⟩
1.0	r3	⟨?x1, owl:sameAs, ?x2⟩, ⟨?x2, owl:differentFrom, ?x3⟩	⟨?x1, owl:differentFrom, ?x3⟩

TABLE 3.1: A set of rules. The rule $r1$ is uncertain while the other rules are certain

The certainty of the facts ($ina1$, $ina2$, $db1$, $db2$) and the rules ($r2$, $r3$) can be modeled in this probabilistic setting by assigning the probabilistic weight 1 to them.

Applying the facts ($ina1$, $ina2$, $db4$, $db2$) and the rule ($r1$) leads to infer that the two person `ina:per1` and `dbpedia:per1` are the same (i.e. ⟨`ina:per1 owl:sameAs dbpedia:per1`⟩) with a probability value. By supposing that the facts and rules are independent we can compute the probability of the inferred `sameAs` link as the multiplication of the probabilities of the facts ($ina1$, $ina2$, $db4$, $db2$) and the rule ($r1$) participated in the inference of this link (i.e. $1 \times 1 \times 0.6 \times 1 \times 0.7 = 0.42$). In the same manner the link ⟨`ina:per2 owl:differentFrom dbpedia:per1`⟩ can be inferred from the facts ($ina3$, $ina4$, $db4$, $db2$) and the rule $r2$ with a probability ($1 \times 1 \times 0.6 \times 1 = 0.6$).

These two examples of uncertain inferred links indicate that, the idea of combining the rule-based reasoning with probabilities yields to very powerful link discovery methods. However,

if we want to apply the probability theory consequently, then we soon run into difficulties. Applying this simple multiplication of the probabilistic weights involved in the inference process would give us for the link $\langle \text{ina:per1 owl:differentFrom ina:per2} \rangle$: $((0.6 \times 0.7) \times (0.6)) = 0.25$. This is not correct, since the probability (0.6) of the fact $db4$ is considered twice. Thus, the proper result should be 0.42.

To overcome this problem we use the settings of the Probabilistic Datalog in which event keys and event expressions are used to detect duplicates while combining the probabilistic weights and thus avoid wrong calculations. Each ground fact and rule is attached with an event key. Each event key attached to a fact or a rule is assigned with a probability equal to the probabilistic weight of that fact or rule. For example, the event key e_{db4} is attached to the fact $db4$. The probability assigned to the event key is e_{db4} is 0.6. Then, the inferred facts are attached with an event expression which is a boolean combination of the facts and rules participated in the inference of this fact. For example, the event expression of the fact $\langle \text{ina:per1 owl:sameAs dbpedia:per1} \rangle$ is $(e_{ina3} \wedge e_{ina4} \wedge e_{db4} \wedge e_{db2})$. The event expressions of the other inferred links are depicted in Table 3.2. The event expressions calculated for the inferred facts can be logically simplified so that duplicates inside them can be removed by applying boolean operations as it is depicted in Table 3.2.

Inferred Fact	Event Expression	Simplified Event Expression
$\langle \text{ina:per1 owl:sameAs db:per1} \rangle$	$(e_{ina1} \wedge e_{ina2} \wedge e_{db4} \wedge e_{db2} \wedge e_{r1})$	$(e_{ina1} \wedge e_{ina2} \wedge e_{db4} \wedge e_{db2} \wedge e_{r1})$
$\langle \text{ina:per2 owl:differentFrom db:per1} \rangle$	$(e_{ina3} \wedge e_{ina4} \wedge e_{db4} \wedge e_{db3} \wedge e_{r2})$	$(e_{ina3} \wedge e_{ina4} \wedge e_{db4} \wedge e_{db3} \wedge e_{r2})$
$\langle \text{ina:per1 owl:differentFrom ina:per2} \rangle$	$(e_{ina1} \wedge e_{ina2} \wedge e_{db4} \wedge e_{db2} \wedge e_{r1} \wedge e_{ina3} \wedge e_{ina4} \wedge e_{db4} \wedge e_{db3} \wedge e_{r2} \wedge e_{r3})$	$(e_{ina1} \wedge e_{ina2} \wedge e_{db4} \wedge e_{db2} \wedge e_{r1} \wedge e_{ina3} \wedge e_{ina4} \wedge e_{db3} \wedge e_{r2} \wedge e_{r3})$

TABLE 3.2: The event expression of the inferred facts and their simplified form.

The probabilistic weight of each inferred fact in the Datalog setting is equal to the probabilistic weight of their simplified event expressions. For example, calculating the probabilistic weight of the simplified event expression depicted in Table 3.2 attached to the inferred link $\langle \text{ina:per1 owl:differentFrom ina:per2} \rangle$ will give the correct probabilistic weight $(1 \times 1 \times 0.6 \times 1 \times 0.7 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 = 0.42)$ instead of 0.25.

In contrast to the case where only certain knowledge is used, the usage of uncertain facts and rules may lead to uncertain facts, some of them are wrong. The standard way of filtering results to be returned to the user is to return only the inferred facts with a probabilistic weight greater than a given threshold. Such facts having a high probabilistic weights are likely to be correct. For this, we have to set up a threshold that maximizes the number of inferred facts while minimizing the number of wrong facts. In this illustrative example, if we accepted that the 3 inferred links are correct in practice, then the value of the threshold should be lower than 0.42. In section 3.5.2 we show an automatic technique that may be used to set up the value of this threshold.

3.3.2 Formal Background

In what follows we extend the formal background presented before in Section 2.4 to deal with uncertain facts and rules. Our formalism is based on Probabilistic Datalog [31] and its underlying semantics.

Probabilistic Deductive RDF Datasets

Probabilistic deductive RDF dataset $pr_dds(u) = \langle F, R, Pr \rangle$ extends a deductive RDF dataset $dds(u) = \langle F, R \rangle$, defined in Section 2.4.4, with a function Pr that assigns to each fact and rule a probabilistic weight using the mapping Pr . The intended meanings of these weights are as follows:

- For a fact f : $Pr(f)$ is the probability that the fact is correct.
- For a rule r : $Pr(r)$ is the probability that the the conclusion of the rule r when it is instantiated by any substitution θ , ie. $\theta.Conc_r$, is correct if its instantiated condition $\theta.Cond_r$ is correct.

The application of the rules enables to infer the set of facts $SAT(F, R, Pr)$ with probabilistic weights. These probabilistic weights are attached to the inferred facts using the mapping Pr .

Evaluating a query $q(u)$ over a probabilistic RDF dataset $pr_dds(u) = \langle F, R, Pr \rangle$ returns the answers of evaluating the same query over the deductive dataset $dds(u) = \langle F, R \rangle$ accompanied with their probabilistic weights. i.e:

$$Answer(q(u), \langle F, R, Pr \rangle) = \bigcup_{\{\theta | \theta.q(u) \subseteq F\}} (\theta.q(u), Pr(\theta.q(u)))$$

The probabilistic weight of each answer is calculated based on the probabilistic weight of the ground facts and rules needed to infer this answer.

The same can be applied to the case of a boolean query $q(u)$ where the query is evaluated to true attached with a probabilistic weight $w = Pr(q(u))$ if and only if $q(u) \in SAT(F, R)$ and in this case we write:

$$\langle F, R, Pr \rangle \vdash_w q(u)$$

. The query is evaluated to false with a probabilistic weight $w = 1$ if $q(u) \notin SAT(F, R)$.

Event keys and event expressions

In probabilistic Datalog, each ground triple and rule is associated with a unique probabilistic event key that may be either true if the corresponding fact or rule is correct or false otherwise. We denote by ε the mapping that maps each ground fact or instantiated rule with an event key:

$$\varepsilon : F \cup R \rightarrow EK$$

where EK denotes the set of event keys

The probability of these event keys is equal to the probability of the corresponding fact or rule.

Then, each derived fact f is assigned with an event expression $EE(f)$ consisting of a boolean combination of the event keys of the ground facts and rules involved in its derivation. More formally, it has the form:

$$EE(f) = (e_1 \wedge \dots \wedge e_m) \vee \dots \vee (e'_1 \wedge \dots \wedge e'_n)$$

$$\text{s.t. } e_1, \dots, e_m, e'_1, \dots, e'_n \in EK$$

where:

- The event keys of the ground facts and rules that are necessarily and sufficient to infer the fact f , i.e one reasoning branch, form a conjunction. More formally, if $(e_1 \wedge \dots \wedge e_m)$ is a conjunction in the event expression of the fact f and ε^{-1} is the inverse function of ε that maps each event key with its corresponding ground fact or rule then:

$$\{\varepsilon^{-1}(e_1), \dots, \varepsilon^{-1}(e_m)\} \vdash f$$

and

$$i = 1, \dots, m$$

$$\{\{\varepsilon^{-1}(e_1), \dots, \varepsilon^{-1}(e_m)\} \setminus \{\varepsilon^{-1}(e_i)\}\} \not\vdash f$$

- The event expressions of the different branches of reasoning that infer the fact f form a disjunction.

These event expressions are simplified to remove duplicates and stored in their disjunctive normal form (DNF). This simplification can be done by applying boolean operations $\{\vee, \wedge\}$ to the expressions. The probability of the simplified event expression of an inferred fact f is used as the probabilistic weight of this fact f .

To calculate the probability values of the event expressions assigned to the inferred facts, the inclusion-exclusion formula [18], which computes the probability of a disjunction of events, must be used. In the simple case where the event expression is a disjunction of only two event keys ($e_1 \vee e_2$) the probability of this event expression is $Pr(e_1 \vee e_2) = Pr(e_1) + Pr(e_2) - Pr(e_1 \wedge e_2)$. In the general case the probability of the expression ($k_1 \vee \dots \vee k_n$) where k_i is a conjunction of event keys can be computed as follows:

$$Pr(k_1 \vee \dots \vee k_n) = \sum_{i=1}^n (-1)^{(i-1)} \left(\sum_{1 < j_1 < \dots < j_i \leq n} Pr(k_{j_1} \wedge \dots \wedge k_{j_i}) \right) \quad (3.1)$$

To calculate the value of this probability we have two options:

- We suppose that the probability of all conjuncts formed in the inclusion-exclusion formula is given.
- We assume the all the event keys are independent and thus the probability of any conjunct of event keys can be calculated as follows:

$$Pr(k_1 \wedge \dots \wedge k_n) = Pr(k_1) \times \dots \times Pr(k_n)$$

It is clear that the assumption in the first option is very hard to fulfill in practice. Thus, We chose to follow the second option.

3.4 ProbFR: The Probabilistic forward Reasoner

In this section we show our probabilistic forward reasoner algorithm, a fact propagation algorithm that deal with probabilistic weights. It accepts probabilistic deductive RDF dataset as input and computes not only the set of facts that can be inferred locally but also their probabilistic weights. Since some facts can be inferred with low probabilistic weights, only the inferred facts with probabilistic weights greater than or equal to a certain acceptance threshold λ_t given as input is returned.

Fact propagation algorithms such as RETE [29] are used to apply a set of rules over a set of facts and they are known for their efficiency thanks to their memorizing techniques. They avoid redundant evaluation of same conditions appearing in different rules by memorizing the facts that are confirmed to match these conditions and the corresponding variable substitutions. To calculate the event expressions of the inferred facts our probabilistic forward reasoner algorithm memorizes for each condition, in addition to the substitutions and facts, their corresponding event keys and event expressions.

At the end of the probabilistic forward reasoner the event expressions calculated for each inferred fact are simplified and their probabilistic weights are computed using the function *Calculate_Pr*. Then all inferred facts that have a probabilistic weight lower than the acceptance threshold are eliminated from the output set of facts.

The ProbFr algorithm can be summarized as follow:

ProbFR
Input: a deductive RDF data $pr_dds(u) = \langle F, R, Pr \rangle$
Input: an acceptance threshold $\lambda_{acceptance}$
Output: the set of inferred fact F' and their probabilistic weights attached using the mapping Pr'

Generate_Event_Keys(F, R, Pr, EE)
FOR each condition triple c that appears in the conditions of the rules in R

- add c to the set of conditions C if there is no condition $c' \in C$ that match with c

$F' \leftarrow F$

FOR each f in F' not seen before

- **FOR** each condition $c \in C$ that can be matched with f , i.e., there exists θ such that $\theta.c = f$
 - **IF** θ is not in the set of already found substitutions $Substitutions_c$ **THEN**
 - * add θ to $Substitutions_c$
 - * $EE(\theta.c) \leftarrow EE(f)$
 - **ELSE** $EE(\theta.c) \leftarrow EE(\theta.c) \vee EE(f)$
 - **FOR** each rule $r : Cond_r \Rightarrow Conc_r$ that have a condition $c' \in Cond_r$ that match with c
 - * **IF** there exists for each condition $c_i \in \{Cond_r \setminus c\}$ a substitution $\theta_i \in Substitutions_{c_i}$ and all these substitutions are compatible with each other and with θ
 - consider θ' as the union of these substitutions and ee' the conjunction of the event expressions of the corresponding facts $\theta_i.c_i$.
 - $\theta_r \leftarrow \theta \cup \theta', ee_r \leftarrow ee' \wedge EE(f)$
 - $f_{new} \leftarrow \theta_r.Conc_r$
 - add f_{new} to F'
 - $EE(f_{new}) \leftarrow EE(f_{new}) \vee ee_r$

FOR each f in F'

- Simplify the event expression $EE(f)$ if possible by applying the boolean operations $\{\wedge, \vee\}$ and convert it to the disjunctive normal form (DNF)
- $Pr'(f) \leftarrow Calculate_Pr(EE(f))$
- **IF** $Pr'(f) < \lambda_{acceptance}$ **THEN** remove f from F'

RETURN $\langle F', Pr' \rangle$

The procedure *Generate_Event_Keys* generate randomly event keys for each input fact and rule and give it a probability value equal to the probability of the corresponding fact or rule.

Calculate_Pr is a procedure that calculates the probabilistic weight of an event expression ee giving the probability of each event key that appears in this event expression E :

Calculate_Pr

Input: an event expression ee , an array Pr the contains probability of each event key in ee

Output: the probability of the input event expression as a float number $n \in [0, 1]$

-
- Initialize s to empty set
 - For each conjunction $conj$ in input event expression E
 - $temp \leftarrow 1$
 - For each event key e in $conj$
 - * $temp \leftarrow temp * Pr(e)$
 - $s \leftarrow concat(s, temp)$
 - return (Inclusion-Exclusion(s))

The Inclusion-Exclusion procedure accepts as input a set of probabilistic values and applies the inclusion-exclusion formula 3.1 as described below:

Inclusion-Exclusion

Input: a set of probabilistic weights s

Output: a float number $res \in [0, 1]$

-
- For $i = 1$ to $|s|$
 - $sign \leftarrow (-1)^{(i-1)}$
 - For each possible subset ss with the size i that can be created from the input set s
 - * $temp \leftarrow 0$
 - * For each k in ss
 - $temp \leftarrow temp + k$
 - * $res \leftarrow res + sign * temp$
 - return res

Instead of implementing our ProbFR algorithm from scratch we decided to reuse the implementation of the RETE algorithm in the JENA framework. We injected inside this implementation:

- The generation of the event keys for the ground facts and rules at the beginning of the algorithm.
- The memorization of the event keys and event expressions of the facts that match the condition patterns.

- The calculation of the probabilistic weights for the inferred facts once the set of inferred facts is calculated.
- The elimination of the facts inferred with probability lower than the input acceptance threshold at the end of the algorithm.

3.5 Data complexity and approximation techniques

In this section we show the data complexity of the probabilistic forward chaining algorithm. We show that the maximum size of an event expression calculated for the inferred facts is polynomial to the size of the dataset. However in practice, even if the size of the event expressions is polynomial to the size of the dataset storing these expressions and dealing with them leads to many difficulties. Thus, we also propose approximation techniques with the aim to avoid such problems.

3.5.1 Maximum size of an event expression

Our goal is to study the maximum size of an event expression in terms of the size of the dataset. We do not consider the number of rules into account while studying the complexity since the number of rules in practice is very small compared to the size of the dataset.

The event keys of the facts and rules in each branch of reasoning appears in the event expression as a conjunction. It is clear that the maximum size of each conjunction is the size of the dataset N . It is the case where all facts in the dataset are needed to infer a particular fact (very rare in practice).

The different branches of reasoning that lead to infer the same fact form a disjunction in the event expressions. The maximum number of these branches is also the size of the dataset N . It is the case where each fact in the dataset is enough to infer a particular fact (very rare in practice).

Then, the maximum size of an event expression is in $\mathcal{O}(N^2)$ in terms of the dataset size complexity.

3.5.2 Approximating probabilistic weights

Although the size of an event expression is polynomial to the size of the dataset, dealing with the full event expressions may lead to many difficulties in:

- The needed memory to store these expressions.
- The time needed to estimate their probability values. The inclusion-exclusion algorithm which calculates the probability of a disjunction of events generates the power set of the set of these events. The worst time complexity to generate the power set is $\mathcal{O}(2^N)$

For these reasons we propose to use the following techniques to control the size of the event expressions:

- For each event expression, remove all conjunctions that have a probability less than a given threshold $\lambda_e \in [0, 1]$. λ_e can be set as a parameter.
- For each event expression, order the conjunctions by their probabilities and keep only the first $\lambda_{max} \in \mathcal{N}$ ones. λ_{max} can be set a parameter.

3.6 Setting up the acceptance threshold

Tuning the acceptance threshold, used as a parameter for the probabilistic forward reasoner presented in 3.4, enables the user to control the number of the inferred facts and their correctness. While setting this parameter to a high value ~ 1 leads the algorithm to return only the certain inferred facts and ignore all other facts, choosing a low value ~ 0 increases the size of the resulted set of facts but increase also the number of wrongly inferred facts that may present in the answer.

The goal is to have the maximum number of the returned correct facts. A straightforward technique to choose this threshold is to vary this parameter over many values in $[0, 1]$, run the algorithm over each value and ask the user to verify manually a sample of the returned results. Then, the best candidate value for this parameter is the value that maximize the size of the results and minimize the number of the facts that are confirmed by the user to be wrong. However, verifying manually many samples is a hard tasks for the user. In what follows, we propose an automatic way based on contradictory facts to set the acceptance threshold without a manual assistance.

We consider that two facts are contradictory if they link the same two entities with both owl:sameAs and owl:differentFrom links. For example the following two facts are contradictory facts:

$$\langle \text{per1}, \text{owl:sameAs}, \text{per2} \rangle$$

$$\langle \text{per1}, \text{owl:differentFrom}, \text{per2} \rangle$$

One direct application of the contradictions is to discover errors in the dataset. It is clear that one of the contradictory facts is wrong and can be determined thanks to the probabilistic weights attached to the two facts.

To find automatically the best acceptance threshold, we assume that minimizing the number of contradictions in the results leads to minimize the number of the incorrect facts. Thus, finding the acceptance threshold turns into a problem of minimizing the number of the contradictions in the resulted set of facts and maximizing the size of the resulted set. Our experiments in section 3.7 validate this assumption experimentally.

3.7 Experiments

Our experiments were conducted over real data from both INA and DBpedia datasets and have the following goals: (1) To show that an important increase in the number of the inferred same-as and different-from links is obtained when uncertain facts and rules are taken into account (2) To show that facts inferred with weights over a particular threshold, that has been set up with manual assistance from the user, are correct. (3) To show that this acceptance threshold can be set up automatically without the manual assistance of the user using the weighted contradictory facts.

3.7.1 Experimental Set-Up

To meet our experimental goal we need: (a) a deductive RDF dataset that contains set of certain facts and rules $\langle F_c, R_c \rangle$. (b) a probabilistic RDF deductive dataset $\langle F_c \cup F_{uc}, R_c \cup R_{uc}, Pr \rangle$ which contains the previous set of certain facts and rules enriched with a set of uncertain facts F_{uc} and uncertain rules R_{uc} and their probabilistic weights.

Our set of **certain facts** F_c contains 30.7 million facts: 30 million RDF facts from the INA dataset, used in our experiments of Chapter 2, and 0.7 million certain facts obtained by downloading from DBpedia.fr and DBpedia.org all the facts about entities having a name similar to the name of one of the homonyms in the INA dataset up to a threshold 0.9. The total size of this dataset in turtle format is 2 GB. For the set of **certain rules** R_c we use the 35 certain rules that have been used before in the experiments of Chapter 2 that are listed in Annex A.

The set of **uncertain rules** R_{uc} contains 16 uncertain rules that enable to conclude sameAs facts. They translates pseudo keys (e.g. name and birthYear or name and deathYear), mappings and other domain knowledge provided by INA experts (e.g. a rule that conclude that two person homonyms are the same if they are participated in the same video entity). We asked the INA experts to rank the uncertain rules on a scale from 1, for rules that are less likely to be correct, to 3, for rules that are most likely to be correct. Then, we attached the probabilistic weight 0.3 to the rules that are assigned with ranking 1 by experts, 0.6 for rules with ranking 2 and 0.9 for rules with ranking 3. The full set of rules and their probabilistic weights are listed in Annex B.

All the facts available in the INA dataset are considered as certain by the INA experts. Thus, we generated a set of **uncertain facts** F_{uc} that contains 30,400 RDF facts about DBpedia person entities that have names similar to one of the homonyms in the INA dataset up to a threshold 0.9. To generate these facts we create for each INA homonym name a query that returns DBpedia's entities which have a name similar to INA homonym name up to a threshold 0.9 and we applied it over DBpedia. Then, we generate the facts that attach to each returned entity the INA name using the property `ina:altLabel`. The normalized edit distance between the two names is used as a probabilistic weight for the generated fact.

During the experiments we set the parameters $\lambda_\epsilon, \lambda_{max}$ to 0.1, 10 accordingly.

For running the experiments we used a machine with intel i7 Quad-core processor, 32 GB of memory and 500 GB sata 2 hard disk with 7200 rpm .

3.7.2 Experimental Results

For evaluating our first experimental goal, (a) we first run our forward reasoner over the deductive RDF dataset $\langle F_c, R_c \rangle$. (b) Then, we run our probabilistic forward reasoner over the probabilistic deductive RDF dataset $\langle F_c \cup F_{uc}, R_c \cup R_{uc}, Pr \rangle$. We compared the number of `owl:sameAs` and `owl:differentFrom` facts obtained on the INA homonyms in each case. We set the acceptance threshold $\lambda_{acceptance}$ for the probabilistic forward reasoner to 0.5.

Applying only the the set of certain rules over the set of certain facts allowed to infer 23,689 facts (including 8,238 `owl:sameAs` facts and 15,451 `owl:differentFrom` facts) compared to the 147,712 facts (130,741 `owl:sameAs` facts and 16,971 `owl:differentFrom` facts) inferred with probabilistic weights greater than 0.5 when uncertain facts and rules are taken into account. This clearly shows the importance increase (about 623%) in the number of inferred facts when uncertain facts and rules are taken into account. We notice that most of the newly inferred facts are `owl:sameAs` since all the uncertain rules that we used conclude `owl:sameAs` facts. The 147,712 facts and their corresponding weights are inferred in 19 minutes and 41 seconds.

The probabilistic weights of the inferred facts are distributed over the range $[0.5, 1.0]$ as it is depicted in Figure 3.2. It is worth noting that the probabilistic weight attached to the certain facts (i.e. the 23,689 facts inferred when only certain facts and rules are used) when the probabilistic forward reasoner is launched over the whole set of certain and uncertain facts and rules is equal to 1.

To find manually the best acceptance threshold, we randomly took four samples $s_{\geq 0.5}$, $s_{\geq 0.6}$, $s_{\geq 0.7}$, $s_{\geq 0.8}$. Each sample contains 200 facts randomly chosen from the inferred `owl:sameAs`

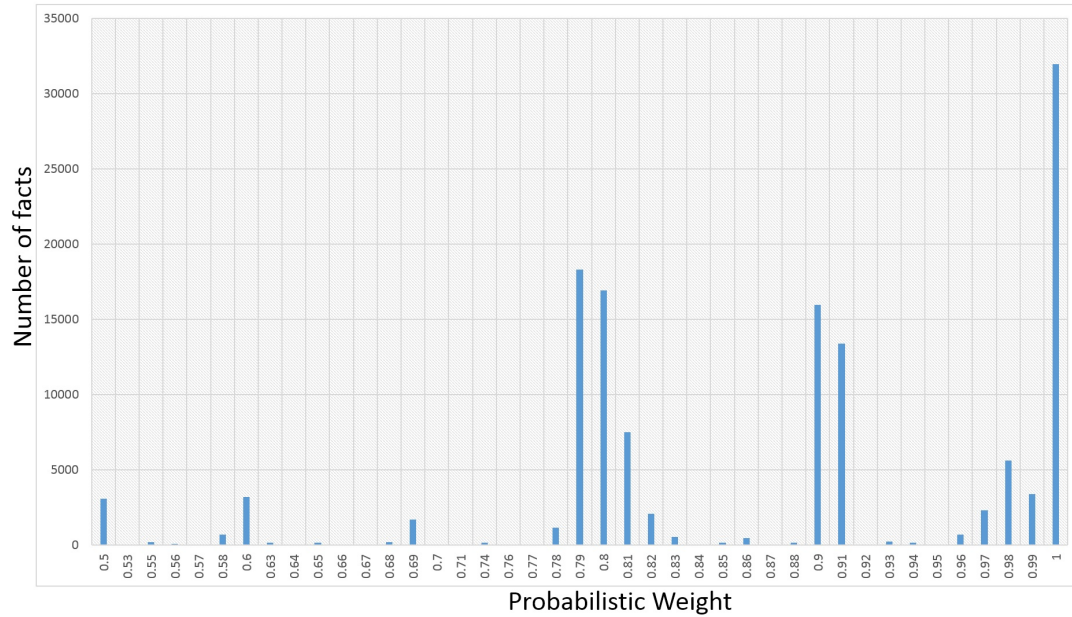


FIGURE 3.2: The distribution of the number of the inferred facts over the range of the probabilistic weights

and owl:differentFrom facts that have probabilistic weights bigger than 0.5, 0.6, 0.7, 0.8, respectively, and we asked INA experts to verify these samples manually. Table 3.3 shows the number of facts confirmed to be correct in each sample by INA experts.

The results show that the higher the probabilistic weight attached to the fact is the lower its probability to be wrong. This confirms the usefulness of the probabilistic weights returned by our algorithms in practice. The results shows also that setting the acceptance threshold parameter of the probabilistic forward reasoning algorithm in this experimental setting to 0.7 leads to get only correctly inferred owl:sameAs and owl:differentFrom facts. It is worth knowing that: among the 147,712 newly inferred sameAs and differentFrom facts, 81% of them have probabilistic weights greater than 0.7. Thus, 81% of them are correct.

Sample	Number of correct facts
$s \geq 0.5$	196
$s \geq 0.6$	199
$s \geq 0.7$	200
$s \geq 0.8$	200

TABLE 3.3: The number of inferred facts confirmed to be correct in each sample

In our third experiment we want to validate experimentally the automatic technique proposed in Section 3.5.2 to choose the acceptance threshold by checking if applying this automatic technique leads to a threshold near 0.7 which has been found manually in our second experiment.

To do so:

1. we created the set of the facts that are confirmed to be wrong by contradiction. i.e. each fact f in this set contradicts with another fact f' that has higher probabilistic weight, i.e. $Pr(f') > Pr(f)$. It is clear that eliminating this set of facts from the resulted set of facts solves the contradiction problem.
2. we studied the distribution of the probabilistic weight attached to these wrong facts.

Figure 3.3 shows that all these wrong facts have probabilistic weights lower than 0.84 and 99.5% of them have probabilistic weights lower than 0.68. By knowing that 83% of the inferred facts have probabilistic weight greater than 0.68% comparing to only 41% of them have probabilistic weights higher than 0.84, we can say that the best value candidate for the acceptance threshold is then 0.68. This conforms with our second experiments in which we found manually that the best value for the acceptance threshold should be near 0.7 and thus validate this automatic technique to find the acceptance threshold experimentally.

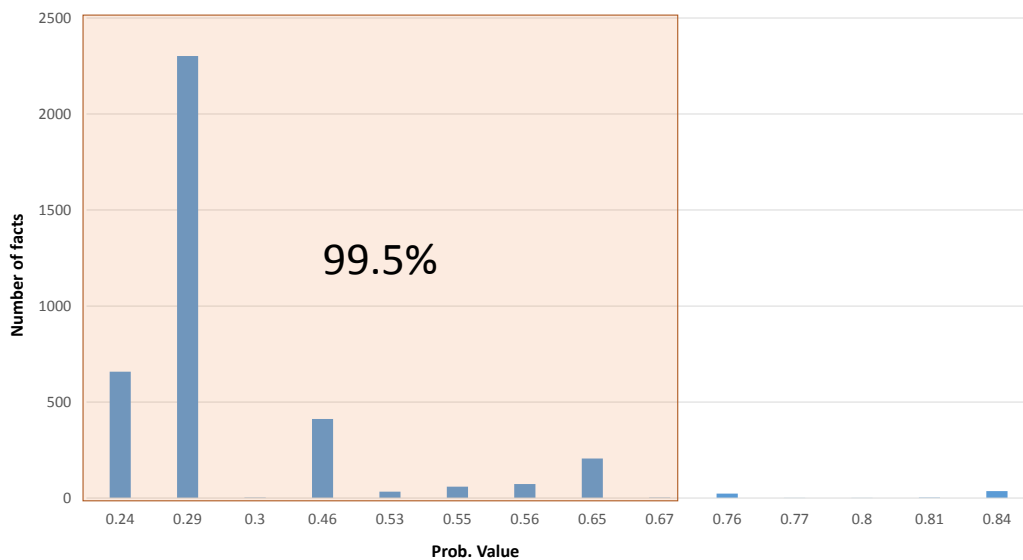


FIGURE 3.3: The distribution of the facts that are confirmed to be wrong by contradiction over the range of the probabilistic weights

3.8 Extending the import by query approach to uncertainty

In this section we want to revisit the import-by-query problem, studied before in Chapter 2, in this probabilistic setting. Given a probabilistic deductive RDF dataset $pr_dds(u) = \langle F, R, Pr \rangle$, and a boolean query $q(u)$ the local evaluation of which gives an empty answer set, our problem statement in this probabilistic settings is to construct a set of external queries $q_1(u_1), \dots, q_l(u_k)$ for which we can guarantee that the subsets of external facts resulting from their evaluation over

the (possibly huge) external datasets are sufficient to answer the initial query with a probabilistic weight bigger than a certain threshold t .

$$\begin{aligned} & \langle F \cup_{i \in [1..k]} Answer(q_i(u_i), ds(u_i)), R, Pr \rangle \vdash_w q(u) \text{ and } w \geq t \\ & \text{iff } \langle F \cup_{i \in [1..k]} ds(u_i), R, Pr \rangle \vdash_{w'} q(u) \text{ and } w' \geq t \end{aligned}$$

3.8.1 The Probabilistic Iterative Import-by-query

The Probabilistic Import-by-query algorithm is an extension of the Import-by-query explained in Section 2.5 to deal with probabilistic weights. Similar to the original algorithm, it accepts as input (1) a boolean same-as query q , (2) a probabilistic deductive RDF dataset $\langle F, R, Pr \rangle$ and (3) set of \bar{u} of query entry points to external datasets in addition to (4) an acceptance threshold t . It alternates steps of sub-query rewriting based on backward chaining and of external query evaluation and returns as output not only a (true or false) answer as the original version but a probabilistic weight greater than or equal to t .

Each query rewriting step is realized using the PQESQ (described in detail in section 3.8.2) which is an adaptation of the Query-External-Subquery, described in Section 2.5.1, to deal with probabilistic weights. PQESQ either succeeds in proving the query locally i.e. over $\langle F, R, Pr \rangle$ with a probabilistic weight greater than or equal to the given acceptance threshold t and then the process is stopped and the result return by the Iterative Import-by-query algorithm is $(true, w)$, or it produces as the original QESQ a set of external queries but attached with probabilistic weights i.e. $\{(q_1(\bar{u}_1), w_1), \dots, q_k(\bar{u}_k), w_k)\}$. The weight of each external query $q_i(\bar{u}_i)$ is the maximum probabilistic weight the input query can be inferred with if there is an answer to q_i in the external datasets. In the case of empty set the process stopped and the answer $(false, 1)$ is returned.

In Each Evaluation step the algorithm first verifies whether there is a chance to answer the query q with a *true* answer attached with a probabilistic weight greater than t . To do this verification, it calculates the maximum probabilistic weight w_{max} to answer the q with *true*: it is the case when there is a *certain* answer for all the external queries. This maximum probabilistic weight can be calculated by applying the inclusion-exclusion algorithm presented in section 3.4 on the set of the weights of the external queries. If $w_{max} < t$ the process is stopped and the algorithm return $(false, 1)$. In the other cases the algorithm choose the external query $q_i(\bar{u}_i)$ that have the highest weight among the set of the external queries and continue exactly as the original version. It submits $q_i(\bar{u}_i)$ to the Linked Data through the specified entry points, memorizes the results in the answer table and marks q_i as an already processed query. In the case of positive results, a new iteration of Iterative Import-by-query is started on the same input except of the set

of facts F enriched with the facts obtained as the results of the evaluation of the external query $q_i(\bar{u}_i)$. If the result is negative, q_i is removed from the set of external queries and evaluation step is recalled over the new set of external queries. In the case of empty set of external queries the answer (*false*, 1) is returned by the algorithm. The probability values attached to the imported facts are equal to a trust values the user has to the external datasets from which these facts are imported. These trust values are either given in the input for each specified SPARQL endpoint or are assumed to be equal to one.

3.8.2 The PQESQ algorithm

The Probabilistic Query-External-Subquery, PQESQ, algorithm extends the QESQ recursive algorithm presented in Section 2.5.1 to handle probabilistic weights. Similar to QESQ, the PQESQ algorithm starts with an input boolean atomic query $q: \langle s, p, o \rangle$ and treats it as the goal to solve using the input probabilistic deductive dataset $\langle F, R, Pr \rangle$. The answer of PQESQ is either (*true*, w) or (*false*, 1) if the query can be evaluated locally with a probabilistic weight w greater than a given threshold t or a non empty set of external queries compatible with vocabulary of the given external datasets attached with probabilistic weights. Although PQESQ returns also the event expressions of the calculated answer or external queries, they can be omitted in the primary call of the algorithm. These event expressions are useful when PQESQ is triggered recursively.

At the beginning of the algorithm each ground fact and rule is attached with a randomly generated event key accessible through the function EE . Then, PQESQ follows the same reasoning steps in QESQ to calculate the answer sets or the external queries. However, during these steps it calculates the event expression composed of the event keys of the ground facts and rule participated in calculating these answer sets or the external queries. When a true answer or an external query is returned by the algorithm the probabilistic weight of the corresponding event expression is calculated using the algorithm presented in section 3.4. False answers means that it is impossible to evaluate the query to true with a probabilistic weight greater than t and no external query can be calculated for it, thus they are always attached with a probabilistic weight 1.

In contrast to QESQ, PQESQ does not stop if an answer is found for the input goal. Instead of that, it checks if the probabilistic weight of the event expression calculated while searching for this answer is greater than the input threshold t . In the positive case the algorithm returns true attached with this probabilistic weight while in the negative case the algorithm continues exploiting the other possible reasoning branches to infer q and thus increasing its probabilistic weight.

PQESQ**Input:** a deductive RDF data $pr_dds(u) = \langle F, R, Pr \rangle$ **Input:** an acceptance threshold $\lambda_{acceptance}$ **Output:** $(True, w', ee)$ or $(False, 1)$ if the query can be answered locally else a set of external queries, their probabilistic weights and their event expressions $\{(q_1(\bar{u}_1), w_1, ee_1), \dots, q_k(\bar{u}_k), w_k, ee_k)\}$. $ee \leftarrow \emptyset$

1. **IF** there exists $g_i = \langle s_i^v, p_i, o_i^v \rangle$ in the subgoals such that the predicate p_i is an EDB predicate, or such that g_i has been handled before (i.e., g_i can be matched to a triple in $goal_{p_i}$)

THEN evaluate the atomic query $g_i = \langle s_i^v, p_i, o_i^v \rangle$ over F , or over $answer_{p_i}$:

- **IF** there is no answer and p_i is not an ODB predicate **THEN** return $(FALSE, 1)$
- **ELSE** for each answer, let θ be the corresponding substitution (it can be empty if the query g_i is boolean), i.e., such that $\theta.g_i \in F$, or $\theta.g_i \in answer_{p_i}$:
 - Let $NewSG$ be the list of new subgoals obtained from SG by removing g_i and by replacing the other subgoals g_j by $\theta.g_j$.
 - **IF** $NewSG$ is empty **THEN**
 - * $ee \leftarrow ee \vee EE(\theta.g_i)$, $w \leftarrow calculate_pweight(ee)$
 - * **IF** $w > t$ **THEN** return $(TRUE, w)$
 - * **ELSE** trigger a new recursive call of QESQ on $NewSG$.
 - **IF** it returns $(TRUE, w', ee')$ **THEN** success \leftarrow true ; $w \leftarrow calculate_Pr(ee \vee (ee' \wedge EE(\theta.g_i)))$
 - **ELSE** (not success and not returned false)
 - add each resulted external query q_{ext} with an event expression ee' to $SetOfQ_{ext}$ with probability $w \leftarrow calculate_Pr(ee \vee (ee' \wedge EE(\theta.g_i)))$

2. **IF** there exist subgoals not seen before with ODB predicates **THEN**:

Let q_{ext} be their conjunction.

IF q_{ext} is compatible with the vocabulary of \bar{u} (the given entry points to external datasets) **THEN**:

- Let $NewSG$ be the list of new subgoals obtained from SG by removing all subgoals in q_{ext}
- **IF** $NewSG$ is empty **THEN** $w \leftarrow calculate_pweight(EE)$ return (q_{ext}, w) **ELSE**
 - for each subgoal $\langle s^v, p, o^v \rangle$ in q_{ext} , add it to the table $goal_p$
 - trigger a new recursive call of QESQ on $NewSG$ and EE :
 - **IF** it does not return **TRUE** or **FALSE** **THEN**
 - consider (Q_{ext}, w) to be the result of $PQESQ(NewSG, EE)$
 - return $(Q_{ext} \cup q_{ext}, w)$

3. **ELSE** (all the subgoals have IDB predicates and they have not been seen before)

- let $g = \langle s^v, p, o^v \rangle$ be the first subgoal in SG and let R_g be the set of rules whose conclusion can be matched with g
- **IF** $R_g = \emptyset$ **THEN** return $(FALSE, 1)$
- **ELSE**:

```

- success  $\leftarrow$  false ;  $w \leftarrow 0$ ;  $SetOfQ_{ext} \leftarrow \emptyset$ ;  $ee \leftarrow \emptyset$ 
- WHILE  $R_g \neq \emptyset$  and  $\neg$  success
  * choose  $r$  in  $R_g$  and let  $\theta$  the substitution s.t.  $\theta.Conc_r = g$ ,
  * let  $NewSG$  be the list of new subgoals obtained from  $SG$  by replacing  $g$  with
     $\theta.TP_1(v_1), \dots, \theta.TP_k(v_k)$  where  $TP_1(v_1), \dots, TP_k(v_k)$  are the conditions of  $r$ ,
    and by replacing the other subgoals  $g_j$  with  $\theta.g_j$ ,
  * add  $g$  to the table  $goal_p$ ,
  * remove  $r$  from  $R_g$ ,
  * trigger a new recursive call of PQESQ on  $NewSG$ ,
  * IF it returns ( TRUE,  $w'$ ,  $ee'$  ) THEN success  $\leftarrow$  true ;  $w \leftarrow$  calculate.Pr( $ee \vee (ee' \wedge$ 
     $EE(R_g)$ )
  * ELSE IF it does not return FALSE THEN
    · add each resulted external query  $q_{ext}$  with an event expression  $ee'$  to  $SetOfQ_{ext}$ 
      with
      probability  $w \leftarrow$  calculate.Pr( $ee \vee (ee' \wedge EE(R_g)$ )

IF success THEN return ( TRUE,  $w$  ),  $EE$  ELSE IF  $SetOfQ_{ext} = \emptyset$  THEN return ( FALSE, 1 ) ELSE return
 $SetOfQ_{ext}$ 

```

3.9 Related Work and Conclusion

In this chapter we proposed an adaptation of our rule-based approach that deal with uncertain facts and rules to solve the problem of data-linkage in Linked Data. This adaptation is based on Probabilistic Datalog which has a solid theoretical background presented before in [31, 52]. Although that the problem of making decisions over uncertain knowledge has been studied before in AI and many other domains (e.g. databases ...) since many years, We found the way Probabilistic Datalog approach deal with uncertainty is very suitable and intuitive to extend our rule-base approach for data-linkage: (a) it is built over Datalog settings (b) it deals with not only certain facts as the approached captured by the provenance semirings [35] but also uncertain rules (c) it attaches probabilistic point values to the inferred facts and not probabilistic ranges as other works that are built also over Datalog settings like [53] (d) our experiments showed that this approach can scale to large datasets. A detailed comparison between probabilistic Datalog and other similar works can be found in [31].

Many instance matching tools has been introduced to produce uncertain links between pairs of individuals in Linked-Data. Some tools do value matching: they match values of certain properties that describe the two individual to be compared (e.g. calculating the similarity between strings) and then aggregating the results using weighted aggregation functions (e.g. weighted average ...) [54]. Others exploit also the schema: PARIS [64] proposes a fully automated

probabilistic approach that needs no tuning parameters to align instances, classes and relations. It exploits the ontologies delivered with the data and generate instance alignments that cross-fertilize with alignments at the schema level between. LN2R [63] combines logical inference on the schema level (by exploiting functionality, inverse functionality . . .) and a numerical method based on equations that model the influences between similarities. Most of these approaches fails in discovering sophisticated matching (e.g. dealing with structural heterogeneity: if one entity is presenter of a title “Dimanche Martin” and other entity is presenter of a video that belongs to a program ‘Dimanche Martin’).

Our proposed approach for Data-linkage is general enough to capture facts and rules attached with weights that may come from many different sources. Thus, the sameAs facts produced automatically using the existing instance matching tools and their weights can be given as an input to our approach. This allows, by using the suitable rules, our approach to use these sameAs facts to discover more facts that require knowledge that has not been handled by the matching tools, e.g. domain expert rules that dealing with structural heterogeneity.

In our experiments, we showed that this approach works well even for a large scale datasets and produce meaningful probabilistic weights. We showed also how to interpret the calculated probabilistic weights to decide the correct facts among the inferred ones.

Part II

Trust in Semantic Web

Chapter 4

Trust in Networks of Ontologies and Alignments

4.1 Introduction

Trust is meant to play a primary role in the realization of the semantic web [17]. As a consequence, trust has gained a lot of attention from researchers of this community [9]. Trust aims at assessing the quality of information contained in the semantic web, which can be considered unsatisfactory, for example, because it is untrue, incomplete or outdated. Just as “one man’s meat is another man’s poison”, trust is commonly agreed not to be objective, but rather subjective to the point of view of every user/agent/peer.

Discrepancy between viewpoints may indeed be a reason for information to be considered unsatisfactory. Moreover, unsatisfactory information can be caused intentionally as the result of some sort of malicious behavior from a user/agent/peer when answering a query. There exists another reason, mostly overlooked by current mechanisms of trust: unsatisfactory information can be due to users/agents/peers’ incapacity to understand each other. Certainly, with the increasing proliferation of ontologies in the semantic web, the target of a query may provide an unsatisfactory answer simply because the query is specified in an ontology different from the target’s ontology.

Today’s most widely followed approach to tackle ontology heterogeneity is ontology matching [27]. Ontology matchers can find alignments, i.e. sets of correspondences between entities (classes, properties or instances) of different ontologies. These correspondences can be used later for query translation. However, computed alignments may be limited — unsound and/or

incomplete — and yet generate flawed query translations. Our proposal is to profit from alignments in the beginning and then proceed *à la trust*. Before going into details, we explain our setting and the goal of our work, and we summarize our contribution.

4.1.1 Semantic P2P Networks

In line with Adjiman et al. [6], we will consider semantic peer-to-peer (P2P) networks, i.e. fully decentralized overlay networks of people or machines (peers) sharing and searching for resources (documents, videos, photos, data) based on their semantic annotations using ontologies. As an ontology language we have restricted ourselves to sets of classes equipped with less-general-than and disjointness relations. In a semantic P2P network, every peer is free to organize its local resources as instances of classes of its own ontology serving as a query interface for other peers. Then alignments between ontologies make it possible to reformulate queries from one local peer vocabulary to another.

The answer to a query is a set of resources (e.g. documents) which are instances of classes corresponding, typically via subsumption or equality, to the initial query posed to particular peers. If the alignment correspondence used to translate a query is incorrect, though, the sender peer might be returned with unsatisfactory instances, i.e. instances it does not consider to be instances of its local class. Think of, for example, a network for exchanging bookmarks, in which a peer Scott organizes his bookmarks according to two main categories: cinema and music. Suppose that Scott is acquainted with Zelda who is interested in music and cities. Scott's and Zelda's ontologies include the class **Paris**, but with different meanings: Scott's **Paris** refers to a film, whereas Zelda's **Paris** refers to the capital of France. Also, imagine that the two ontologies have been aligned by a matcher which has not been able to overcome this homonymy problem, and has returned a correspondence stating that Scott's **Paris** and Zelda's **Paris** are equivalent classes. Thus, if Scott wants to acquire bookmarks of the film "Paris" and queries Zelda, then he will be presented with bookmarks related with the capital of France instead.

4.1.2 Trust in Semantic P2P Networks

Our goal is to design a mechanism of trust for guiding the query-answering process in semantic P2P networks. We define the trust of a peer P_1 towards another peer P_2 with respect to a pair $\langle C, D \rangle$, where C and D are classes of P_1 's and P_2 's ontologies, resp. as the probability that an arbitrary instance of D in P_2 's ontology is considered satisfactory, i.e. an instance of C by P_1 . Here the class C is meant to be queried by P_1 , and P_2 's answer, after C is reformulated as D , is the set of instances of D in P_2 's ontology. Notice that trust is subject to classes and translations. Going back to our example of bookmarks, imagine that Scott's and Zelda's ontologies include

the class `Gainsbourg` with which Scott and Zelda both organize bookmarks about the French songwriter Serge Gainsbourg. Although Scott should not trust Zelda with respect to `Paris`, he should trust her concerning `Gainsbourg`.

4.1.3 Summary of our Contribution

This chapter extends preliminary theoretical and experimental results published in [13] and [7], respectively.

The distinguishing point of our approach is to exploit knowledge provided by alignments (possibly unsound and/or incomplete) together with user feedback in order to compute and refine trust over time. More precisely,

- The trust that a peer P_1 has towards another peer P_2 with respect to the query translation $\langle C, D \rangle$ is modeled as a random variable representing P_1 's belief that instances of D returned by P_2 are satisfactory instances for its local class C .
- Alignments between peers' ontologies are used to construct prior distributions of these beliefs, from which initial values of trust are calculated.
- Bayesian inference is performed to build posterior distributions of peers' beliefs from their prior distributions and user feedback on the number of satisfactory and unsatisfactory instances included in a sample of answers.

It is worth noting that in our approach we profit from ontologies for limiting the recourse to user feedback by substituting manual assessment with logical inference when possible. This is the case when some instances in the sample to be processed by a given peer for a given class C already belong to its local populated ontology:

- If one of such instances belongs to C or to a subclass of C , it can be logically inferred that it is a satisfactory instance for C .
- If one of such instances belongs to a class C' that is declared to be disjoint from C or from one of its superclasses, it can be logically inferred that it is not a satisfactory instance for C .

A by-product contribution of trust is the computation of probabilities for the new instances of a class C to be satisfactory depending on the trusted peers from whom they were obtained. Such instances are added to the populated ontology along with their associated probability values.

These values are computed on the basis of provenance information and with the help of a function which aggregates the trust values towards the peers that provided these instances. Peers' populated ontologies, thus, evolve over time so that each class C in peer P_i 's ontology includes instances P_i is 100% sure of, and instances it believes they belong to C with some probability degree p . In this way, the different instances of a class can be ranked with respect to their associated degrees.

For the evaluation of our mechanism of trust, we have designed a scenario representative of the novel topic of social networks and the semantic web [48]. Based on this scenario, we have built a semantic P2P bookmarking system in which we can vary different quantitative and qualitative parameters so as to measure their influence on (i) the convergence of trust, and (ii) the gain in the quality of peer answers — measured with precision and recall — when query-answering is guided by trust.

The remainder of this chapter is structured as follows. Section 4.2 includes the necessary preliminaries to understand the theoretical model of trust introduced in Section 4.3. An evaluation of this model is presented in Section 4.4. Section 4.5 concludes the chapter and summarizes the related work

4.2 Preliminaries

The components of a semantic P2P network are presented below: ontologies and populated ontologies, alignments and acquaintance graphs. We also describe the query answering process that we consider in this setting.

4.2.1 Ontologies and Populated Ontologies

We draw a distinction between the ontological structure and the instances used to populate it. We deal with lightweight ontologies: classes linked by means of less-general-than and disjointness relations.

Definition 4.1. An ontology is a tuple $O = \langle C, \leq, \perp \rangle$ where C is a non-empty finite set of class symbols; \leq is a reflexive, antisymmetric and transitive relation (a partial order) on C ; \perp is an irreflexive and symmetric relation on C ; and for all $c, c', d, d' \in C$,

$$\text{if } c \perp d, c' \leq c \text{ and } d' \leq d \text{ then } c' \perp d'$$

A populated ontology is the result of adding instances to an ontology in accordance to the intended meaning of the two ontological relations.

Definition 4.2. A populated ontology \mathcal{O} is a tuple $\langle O, \mathcal{I}, ext \rangle$, where O is an ontology, \mathcal{I} is a set of instances, and ext is a function that maps each class c of O with a subset $ext(c)$ of \mathcal{I} called the extension of c , in such a way that the family of class extensions covers \mathcal{I} , and for all classes c, d the following hold:

1. if $c \leq d$ then $ext(c) \subseteq ext(d)$
2. if $c \perp d$ then $ext(c) \cap ext(d) = \emptyset$

4.2.2 Alignments

In an open and dynamic environment such as a P2P network, the assumption that peers share the same ontology is not realistic. Nonetheless, if peers fall back on different ontologies, there must be a way to connect ontologies and translate queries so that their addressees are able to process them. Typically this is done by means of alignments — collections of correspondences between semantically related ontological entities — and finding alignments is the aim of ontology matching [27].

We consider correspondences between two classes c and c' of two distinct ontologies O and O' as tuples $\langle c, c', r \rangle$ with $r \in \{=, \leq, <, \geq, >, \bowtie, \perp\}$, where $c = c'$ (more rigorously, $\langle c, c', = \rangle$) is read “ c is equal to c' ”, $c \leq c'$ (respectively $c < c'$) is read “ c is (strictly) less general than c' ”, $c \geq c'$ (respectively $c > c'$) is read “ c is (strictly) more general than c' ”, and $c \perp c'$ is read “ c is disjoint from c' ”. The non-standard symbol \bowtie expresses overlapping between classes the extensions of which share instances but no one is equal to or contained into the other.

Definition 4.3. Let O and O' be two ontologies, and let c and c' be two classes of O and O' , respectively. A correspondence between c and c' is a tuple $\langle c, c', r \rangle$ with $r \in \{=, \leq, <, \geq, >, \bowtie, \perp\}$. An alignment \mathcal{A} between O and O' is a non-empty set of correspondences between classes of O and O' .

4.2.3 Peers and Acquaintance Graphs

We consider a finite set $\mathcal{P} = \{P_i\}_{i=1}^N$ of peers. In this work, P_i will be identified by i . We assume that each peer P_i is associated with one populated ontology $\mathcal{O}_i = \langle O_i, \mathcal{I}_i, ext_i \rangle$ (where $1 \leq i \leq N$).

An acquaintance graph stands for peers' acquaintances (or neighbors) in the network. As usual, a link between two peers reflects the fact that they know the existence of each other. In addition, we assume that there exists one alignment between their respective ontologies.

Definition 4.4. An *acquaintance graph* is a labeled directed graph $\langle \mathcal{P}, \text{ACQ} \rangle$ where $\mathcal{P} = \{P_i\}_{i=1}^n$ is the set of vertices and any edge in ACQ is of the form $\langle i, j \rangle$ with $i \neq j$, and it is labeled with an alignment \mathcal{A}_{ij} between ontologies O_i and O_j . Peer P_j is said to be an *acquaintance* of peer P_i if $\langle i, j \rangle \in \text{ACQ}$. The set of acquaintances of P_i is denoted by $\text{ACQ}(P_i)$.

4.2.4 Queries and Query Translations

Peers pose queries in order to obtain information about other peers' populated ontologies. We deal with a simple query language, as peers can only query class instances: if peer P_j is an acquaintance of peer P_i , it may be asked

$$Q = c(X)? \quad (4.1)$$

by P_i with $c \in O_i$. Now, since we do not assume that all peers share the same ontology, queries may require to be translated for their recipients to be able to process them. Query translations are determined by correspondences of the alignments of the network. Specifically, if peer P_i wants to send Q to P_j , it will first choose one correspondence $\langle c, d, r \rangle \in \mathcal{A}_{ij}$ (typically r is equal to $=$, \geq or $>$) and then send to P_j the translation

$$Q' = d(X)? \quad (4.2)$$

The answer to (4.1) through its translation (4.2) is the set of instances of class d in P_j 's populated ontology. Unlike queries, we assume that no translation of instances is ever required. Nonetheless, since alignments may be unsound and incomplete, this answer may contain *unsatisfactory* instances, *i.e.* instances which are not considered instances of c by P_i .

A peer cannot foresee which of its acquainted peers will answer its query with satisfactory instances. Furthermore, even if an answer is received, this might be too big to be manually processed, so the peer is left to accept it with the risk of populating its ontology with unsatisfactory instances. This uncertainty can be estimated with the help of a trust mechanism.

4.3 The Trust Mechanism

In this section we detail the trust mechanism that we have designed to assist peers to select the peers in the network better suited for their queries, that is, the peers that will provide answers with more satisfactory instances. For doing so, we adopt an approach based on Bayesian probabilities. The whole trust-based query answering process that we have designed is not simple and we have structured its presentation as follows. In Section 4.3.1, we present our probabilistic

model of trust defined as beliefs on the quality of answers depending on the sources (classes in peers' populated ontologies) they come from. These beliefs are first modeled by prior distributions (built from the alignments) and then refined into posterior distributions when more data become available, as explained in Section 4.3.2. As a result of trust refinement over time, peers' populated ontologies are updated. Section 4.3.3 sets the principles we have chosen to filter the new instances that can be added by exploiting their provenances. Finally, we explain in Section 4.3.4 our use of trust for guiding query answering in a P2P setting.

4.3.1 Definition and Estimation of Trust

We look at trust as a way to estimate each peer's belief that answers returned by other peers include satisfactory instances for classes of its own ontology. The notion of a satisfactory instance can be faithfully captured by means of reference populated ontologies which are not known in practice but useful for the formal definition of our problem.

For each peer P_i , we denote by $\mathcal{O}_i^* = \langle O_i^*, \mathcal{I}_i^*, ext_i^* \rangle$ its reference populated ontology with $O_i^* = O_i$. This corresponds to the ideal case in which peer P_i had access to all instances in the network and classified them according to its ontology O_i . This allows to express that P_i considers an arbitrary instance a as an instance of $c \in C_i$ by $a \in ext_i^*(c)$.

Once an answer is received, it can be (partially) added or not to the extension of the queried class. In order to capture the evolution of class extensions in the network, we consider a time variable $t \in \mathbb{N}$, and write \mathcal{O}_i^t to denote peer P_i 's populated ontology at t (beginning with \mathcal{O}_i):

$$\mathcal{O}_i = \mathcal{O}_i^0, \mathcal{O}_i^1, \dots, \mathcal{O}_i^t, \dots \quad (4.3)$$

It is important to note that the time parameter t is local to each peer and is just a way to model successive queries. It is assumed that the underlying ontology does not ever change: $O_i = O_i^t$ for every $t \in \mathbb{N}$. In addition, peers join the network only with satisfactory instances: $ext_i^0(c) = ext_i^0(c) \subseteq ext_i^*(c)$ for every $c \in C_i$. Later, though, peers may accidentally add unsatisfactory instances to their populated ontologies due to inaccurate trust-based estimations. For this reason, we make a distinction between the instances peers are 100% sure of from the rest. Specifically, at any time $t \in \mathbb{N}$, $ext_i^t(c)$ is split into two disjoint sets:

$$ext_i^t(c) = \overline{ext}_i^t(c) \uplus \widetilde{ext}_i^t(c) \quad (4.4)$$

where $\overline{ext}_i^t(c)$ is the greatest subset of $ext_i^t(c) \cap ext_i^*(c)$ known by P_i . The axioms of Definition 4.2 only hold for this kind of instances. The way these sets are built is explained in Section 4.3.2. Certainly, at $t = 0$, we have that $\overline{ext}_i^0(c) = ext_i^0(c) = ext_i(c)$, or, equivalently, $\widetilde{ext}_i^0(c) = \emptyset$.

With this new terminology, P_j 's answer to query (4.1) through its translation (4.2) at time t is the extension $ext_j^t(d)$, and an arbitrary instance $a \in ext_j^t(d)$ is qualified as satisfactory as long as $a \in ext_i^*(c)$. The proportion of satisfactory instances in $ext_j^t(d)$ is given by the conditional probability $p(ext_i^*(c)|ext_j^t(d))$.¹ Our standpoint is that the higher this value is, the more P_i trusts P_j with respect to the translation $\langle c, d \rangle$.

Definition 4.5. Let us consider two peers P_i and P_j ($i \neq j$) and classes c and d of O_i and O_j , respectively. The trust that P_i has towards P_j with respect to the translation $\langle c, d \rangle$ at time t is the conditional probability $p(ext_i^*(c)|ext_j^t(d))$ and it is denoted by $trust^t(P_i, P_j, \langle c, d \rangle)$.

This idea differs from all existing approaches to trust. In our setting unsatisfactory answers are seen as the result of peers' incapacity to understand each other. Also, trust depends on translations: peers may be trustworthy with respect to some translations but not to others.

The exact value of trust is unknown and can only be estimated. More precisely, $trust^t(P_i, P_j, \langle c, d \rangle)$ is the unknown parameter θ of the Bernoulli distribution of the binary random variable defined on $ext_j^t(d)$ and assigning the value 1 to instances that are satisfactory instances of the class c in P_i . The Bernoulli distribution is the simplest discrete distribution having two possible outcomes labeled by 1 and 0 in which 1 (satisfactory) occurs with probability θ and 0 (failure) occurs with probability $1 - \theta$, where θ is called the parameter of the distribution.

Our approach consists in estimating this parameter by Bayesian inference. For this, we consider a probability distribution $T^t(P_i, P_j, \langle c, d \rangle)$ which represents peer P_i 's belief about the parameter $\theta = trust^t(P_i, P_j, \langle c, d \rangle) = p(ext_i^*(c)|ext_j^t(d))$. In this way, we can give the estimation

$$\widehat{trust}^t(P_i, P_j, \langle c, d \rangle) = E(T^t(P_i, P_j, \langle c, d \rangle)) \quad (4.5)$$

where $E(\cdot)$ denotes the expected value.

From here on, we focus on the construction of prior and posterior distributions of $T^t(P_i, P_j, \langle c, d \rangle)$.

4.3.2 Computation and Refinement of Trust Estimation

In case that P_i has no direct experience interacting with P_j , the alignment \mathcal{A}_{ij} is taken to construct a prior distribution of $T^t(P_i, P_j, \langle c, d \rangle)$ from which a prior belief is computed (as its expected value, according to Equation (4.5)). P_j 's later answers are used to compute posterior distributions of $T^t(P_i, P_j, \langle c, d \rangle)$ and to revise beliefs. We use beta distributions $Beta(\alpha, \beta)$ as they are typically used to estimate the parameter of a Bernoulli distribution. More precisely, it

¹The probability space under consideration here is the triple $(\Omega, \mathfrak{A}, p(\cdot))$ where Ω is the set of instances of the network (a finite set), the σ -algebra \mathfrak{A} is the power set of Ω , and $p(\cdot)$ is Laplace's definition of probability.

is known that if its prior distribution is a Beta distribution of parameters α and β , then its posterior distribution, after n observations o_1, \dots, o_n (each o_i is either 1 or 0) of a random sample of answers, is also a Beta distribution, whose parameters are $\alpha + \sum_{i=1}^n o_i$ and $\beta + (n - \sum_{i=1}^n o_i)$ [21].

Thus, (4.5) can be replaced by

$$\widehat{trust}^t(P_i, P_j, \langle c, d \rangle) = E(\text{Beta}(\alpha, \beta)) = \frac{\alpha}{\alpha + \beta} \quad (4.6)$$

We explain now how the parameters α and β are computed and updated.

No direct experience: alignment-based trust. If $T^t(P_i, P_j, \langle c, d \rangle)$ is not defined (e.g. when $t = 0$), the alignment \mathcal{A}_{ij} is brought into play in order to set the beta distribution serving as a prior distribution.

Figure 4.1 shows the density functions of the beta distributions associated with the different alignment relations. The construction is based on the intended meaning of these relations. In case \mathcal{A}_{ij} is sound, then

$$\begin{aligned} \langle c, d, = \rangle \in \mathcal{A}_{ij} & \quad \text{iff} \quad ext_i^*(c) = ext_j^*(d) \\ \langle c, d, > \rangle \in \mathcal{A}_{ij} & \quad \text{iff} \quad ext_i^*(c) \supset ext_j^*(d) \\ \langle c, d, < \rangle \in \mathcal{A}_{ij} & \quad \text{iff} \quad ext_i^*(c) \subset ext_j^*(d) \\ \langle c, d, \perp \rangle \in \mathcal{A}_{ij} & \quad \text{iff} \quad ext_i^*(c) \cap ext_j^*(d) = \emptyset \\ \langle c, d, \not\sim \rangle \in \mathcal{A}_{ij} & \quad \text{iff} \quad \text{none of the above holds} \end{aligned}$$

Hence, provided that $ext_j^t(d) \subseteq ext_j^*(d)$, we have

$$\begin{aligned} \text{if } \langle c, d, = \rangle \text{ or } \langle c, d, > \rangle \in \mathcal{A}_{ij} & \quad \text{then} \quad p(ext_i^*(c) | ext_j^t(d)) = 1 \\ \text{if } \langle c, d, \perp \rangle \in \mathcal{A}_{ij} & \quad \text{then} \quad p(ext_i^*(c) | ext_j^t(d)) = 0 \\ \text{if } \langle c, d, < \rangle \text{ or } \langle c, d, \not\sim \rangle \in \mathcal{A}_{ij} & \quad \text{then} \quad p(ext_i^*(c) | ext_j^t(d)) \in [0, 1] \end{aligned}$$

For this reason, in the case of the relations $=$, $>$ and \geq we consider a beta distribution whose mean is close to 1, and we take its symmetric for the case of \perp . For the relations $<$ and $\not\sim$ we take the uniform distribution $U[0, 1] = \text{Beta}(1, 1)$. Finally, in the case of \leq we consider a mixture between the two different distributions associated with $=$ and $<$.

Direct experience: trust refinement. Imagine that P_i receives $B = ext_j^t(d)$ from P_j as an answer to the query $c(X)?$ ($c \in C_i$). Sampling with replacement is performed over the set B in

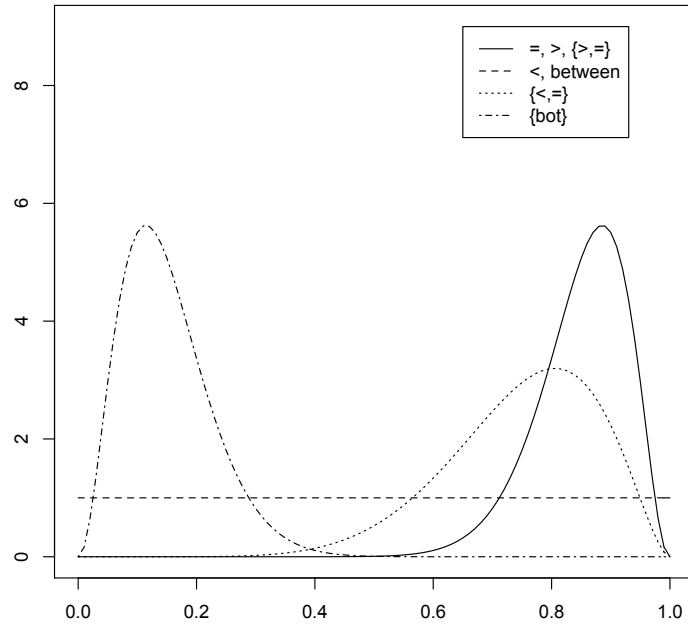


FIGURE 4.1: Beta density functions for different alignment relations.

order to estimate the number of satisfactory instances in B . We assume that each peer can call an oracle (typically the user) to find out whether an instance is satisfactory. More specifically, given $a \in B$, P_i 's oracle is able to give a yes/no response to the question

$$a \in ext_i^*(c)?$$

We can lighten the burden on P_i 's oracle by exploiting the information contained in P_i 's populated ontology. Indeed, the set B can be partitioned into three disjoint subsets:

$$B = B_{aut}^+ \uplus B_{aut}^- \uplus B_{aut}^- \quad (4.7)$$

where

- $B_{aut}^+ = \{a \in B : a \in \overline{ext}_i^t(c)\} = B \cap \overline{ext}_i^t(c)$
- $B_{aut}^- = \{a \in B : \text{there exists } c' \in C_i \text{ with } a \in \overline{ext}_i^t(c') \text{ and } c \perp c'\}$
- $B_{aut}^- = B \setminus (B_{aut}^+ \cup B_{aut}^-)$

Thus, given $a \in B$, there is no need to call the oracle in case that $a \in B_{aut}^+$ or $a \in B_{aut}^-$ since a will be considered, respectively, satisfactory or unsatisfactory automatically. Notice that only instances peers are 100% sure of are taken into account in the construction of B_{aut}^+ and B_{aut}^- .

Algorithm 1 shows the process of sampling with replacement to estimate trust at time t . It is assumed that only a maximum number $N \geq 0$ of oracle calls is permitted (in practice, this number is given by the user). The outputs of the algorithm are the numbers of satisfactory and unsatisfactory instances included in the sample, r and s , respectively, and the sets of satisfactory and unsatisfactory instances identified by the oracle, B_{ora}^+ and B_{ora}^- , respectively. Notice that we impose the condition $N < |B_{aut}^-|$. Indeed, if $N \geq |B_{aut}^-|$ (this happens when, for instance, $B_{aut}^- = \emptyset$) the algorithm will not terminate. In this case, however, we do not need to do sampling, as the whole answer can be evaluated with the help of the oracle. This process also results in numbers r and s , and sets B_{ora}^+ and B_{ora}^- defined similarly as before.

Algorithm 1: Estimation of Trust: Sampling with Replacement

Input: N {maximum number of oracle calls}, $N \geq 0$
Input: $B = B_{aut}^+ \uplus B_{aut}^- \uplus B_{\overline{aut}}^-$ {answer}, $N < |B_{\overline{aut}}^-|$
 $r \leftarrow 0, s \leftarrow 0, n \leftarrow 0$
 $B_{ora}^+ \leftarrow \emptyset, B_{ora}^- \leftarrow \emptyset$
while $n \leq N$ **do**
 select randomly $a \in B$
 if $a \in B_{aut}^+$ **or** $a \in B_{ora}^+$ **then**
 $r \leftarrow r + 1$
 else if $a \in B_{\overline{aut}}^-$ **or** $a \in B_{ora}^-$ **then**
 $s \leftarrow s + 1$
 else if oracle's answer to " $a \in ext_i^*(c)$?" is yes **then**
 $r \leftarrow r + 1, n \leftarrow n + 1$
 $B_{ora}^+ \leftarrow B_{ora}^+ \cup \{a\}$
 else if oracle's answer to " $a \in ext_i^*(c)$?" is no **then**
 $s \leftarrow s + 1, n \leftarrow n + 1$
 $B_{ora}^- \leftarrow B_{ora}^- \cup \{a\}$
 end if
end while
Output: $r, s, B_{ora}^+, B_{ora}^-$

If $T^t(P_i, P_j, \langle c, d \rangle) = \text{Beta}(\alpha, \beta)$ then we construct

$$T^{t+1}(P_i, P_j, \langle c, d \rangle) = \text{Beta}(\alpha + r, \beta + s)$$

which represents P_i 's posterior belief about $\theta = p(ext_i^*(c)|ext_j^t(d))$.

4.3.3 Update of Populated Ontologies

Going back to (4.7), the set $B_{\overline{aut}}^-$ represents the actual new information peer P_i is interested in. If B_{ora}^+ and B_{ora}^- are returned by Algorithm 1 then

$$B_{\overline{aut}}^- = B_{ora}^+ \uplus B_{ora}^- \uplus B_{\overline{ora}}^-$$

The set $B_{\overline{ora}}^-$ is just discarded, whereas B_{ora}^+ and $B_{\overline{ora}}^-$ can be added to $ext_i^t(c)$ to form $ext_i^{t+1}(c)$. We define

$$\overline{ext}_i^{t+1}(c) = \overline{ext}_i^t(c) \cup B_{ora}^+ \quad (4.8)$$

The set $B_{\overline{ora}}$, in turn, represents the set of new instances P_i is not completely sure about. Whether to add this set or not it depends on the new computed trust value:

$$\widetilde{ext}_i^{t+1}(c) = \begin{cases} \widetilde{ext}_i^t(c) \cup B_{\overline{ora}} & \text{if } \widehat{trust}^{t+1}(P_i, P_j, \langle c, d \rangle) \geq \lambda_{trust} \\ \widetilde{ext}_i^t(c) & \text{otherwise} \end{cases} \quad (4.9)$$

where $\lambda_{trust} \in [0, 1]$ is a threshold of trust (for instance, $\lambda_{trust} = 0.75$).

In order for \mathcal{O}_i^{t+1} to be a populated ontology, though, any instance added to the extension of c must also be added to the extension of any superclass of c . If $c' \in C_i$ is such that $c \leq c'$ then

$$\overline{ext}_i^{t+1}(c') = \overline{ext}_i^t(c') \cup B_{\overline{ora}}^+$$

Notice that we proceed to this inference only for the instances that peer P_i are 100% sure of.

By construction, \mathcal{O}_i^{t+1} turns out to be a populated ontology. This explains how the sequence advanced in (4.3) is built.

Provenance-based trust aggregation. Some of the instances added to $\widetilde{ext}_i^t(c)$ to form $\widetilde{ext}_i^{t+1}(c)$ may be later again received (or have already been received) by P_i as part of answers to queries posed to other peers. These peers can be trusted by P_i in different levels. Keeping the peer provenance of the instances that are not 100% sure to be satisfactory for the class c in P_i allows us to combine the corresponding trust values to estimate the probability of these instances to be satisfactory. The following definition of provenance takes into account that trust values evolve over time, which may lead to discarding instances (as explained below).

Definition 4.6. Let $a \in \mathcal{I}_i^t$ be an instance included in P_i 's populated ontology \mathcal{O}_i^t . The provenance of a at time t according to P_i , denoted by $prov_i^t(a)$, is a set of pairs $\langle P_j, d_j \rangle$, where d_j is a class of P_j 's ontology, computed recursively as follows:

- If P_i receives $ext_j^s(d_j)$ from P_j as an answer to a query and $a \in ext_j^s(d_j)$ then $prov_i^{s+1}(a) = prov_i^s(a) \cup \{\langle P_j, d_j \rangle\}$ unless $prov_i^s(a)$ is not defined in which case $prov_i^{s+1}(a) = \{\langle P_j, d_j \rangle\}$.
- If P_i receives $ext_j^s(d_j)$ from P_j as an answer to a query and $a \notin ext_j^s(d)$ but $\langle P_j, d_j \rangle \in prov_i^s(a)$ then $prov_i^{s+1}(a) = prov_i^s(a) \setminus \{\langle P_j, d_j \rangle\}$.

The idea behind provenance is that, if

$$prov_i^t(a) = \{\langle P_{j_1}, d_1 \rangle, \langle P_{j_2}, d_2 \rangle, \dots, \langle P_{j_m}, d_m \rangle\} \quad (4.10)$$

then, as far as P_i is aware, $a \in ext_{j_k}^t(d_{j_k})$ for each $k = 1, \dots, m$.

Now, if $a \in \widetilde{ext}_i^t(c)$, we can use (4.10) to estimate the probability of a to be satisfactory. The idea is to aggregate the values

$$\widehat{trust}^t(P_i, P_{j_1}, \langle c, d_{j_1} \rangle), \widehat{trust}^t(P_i, P_{j_2}, \langle c, d_{j_2} \rangle), \dots, \widehat{trust}^t(P_i, P_{j_m}, \langle c, d_{j_m} \rangle)$$

One possibility is to consider the mean:

$$p = \frac{1}{m} \sum_{k=1}^m \widehat{trust}^t(P_i, P_{j_k}, \langle c, d_{j_k} \rangle)$$

but other aggregation functions (e.g. the min or max) can be considered.

From here on, we will speak of satisfactory instances if we want to refer to instances of the subset $\overline{ext}_i^t(c)$, whereas the instances of $\widetilde{ext}_i^t(c)$ will be referred to as satisfactory instances with degree p .

Discard of instances. Instances added to populated ontologies due to wrong trust decisions might be later discarded when updating trust: if a is a satisfactory instance in $\widetilde{ext}_i^t(c)$ with degree p , and it happens that p is lower than a given threshold of discard $\lambda_{discard} \in [0, 1]$ then a is removed from $\widetilde{ext}_i^t(c)$. This justifies the second bullet of Definition 4.6.

4.3.4 Use of Trust

We use trust for guiding query answering process as follows. Each time t peer P_i has to choose among the set

$$\mathcal{P}_0 = \{\langle P_j, d_j \rangle : P_j \in \text{ACQ}(P_i) \text{ and } d_j \in O_j\}$$

which acquaintance to query for getting new instances for its class c , P_i will opt for $\langle P_{j_0}, d_{j_0} \rangle$ such that

$$\widehat{trust}^t(P_i, P_{j_0}, \langle c, d_{j_0} \rangle) = \max_{\langle P_j, d_j \rangle \in \mathcal{P}_0} \{\widehat{trust}^t(P_i, P_j, \langle c, d_j \rangle)\} \quad (4.11)$$

4.4 Experimentation and Evaluation

This section reports on an experimental campaign conducted with the aim at studying (i) the convergence of trust, and (ii) the gain in the quality of peers' answers — measured with precision and recall — when query answering is guided by trust. Specifically, we were interested in assessing the impact of parameters such that (a) number of peers, (b) quality of initial alignments and (c) thresholds of trust. For this, we have designed a P2P semantic bookmarking system representative of the convergence of the semantic web and social networks.

In what follows we first explain the experimental design (Section 4.4.1) and then summarize and analyze the results of the experimentation (Section 4.4.2).

4.4.1 Experimental Design

Some of the components of the semantic P2P bookmarking networks are generated automatically. This is the case of the topologies of the underlying networks of peers (denoted by $\langle \mathcal{P}, \text{ACQ} \rangle$ in our model of trust) once the number of peers is specified (Section 4.4.1). However, we have decided to build semi-automatically 5 ontologies of topics (O_i in the model) to be used in any generated network (Section 4.4.1). The construction of these ontologies is based on the choice of a list of homonyms, which are part of the leaves in the ontologies, as homonymy is still a Gordian knot for ontology matchers. The rest of classes are extracted from the taxonomy of categories of Wikipedia. Each class is populated with all the URLs associated with its correspondent Wikipedia category, and others returned by Google, to form the 5 reference populated ontologies (O_i^*) (Section 4.4.1). When a P2P network is generated each peer is randomly assigned one of the 5 ontologies, which is populated (O_i) by removing instances from its corresponding reference populated ontology (Section 4.4.1). Even if two peers share the same ontology, they will share few instances before the interaction begins. All the 5 ontologies have been aligned by using state-of-the-art matchers (Section 4.4.1). The quality of the resulting alignments (\mathcal{A}_{ij}) is measured on the basis of reference alignments computed manually. At the end of the section we explain how we simulate query answering in the networks (Section 4.4.1).

Generation of P2P networks One of the parameters of our system is the number of peers. Once it is set, a P2P network with the number of peers specified is generated. Social bookmarking networks are a particular case of social networks, and, since social networks are well-known to exhibit small-world characteristics [51], we only generate networks with small-world topologies. We do so by running Kleinberg’s algorithm included in the JUNG Java library.² This algorithm takes as an input an integer (whose square root is required to be an integer too) to be the total number of peers in the network. In our experiments we set this parameter to k^2 with $k = 5, 10, 15, 20, 30$.

Construction of the ontologies As hinted before, we have built 5 ontologies of topics in a semi-automatic manner. First, we selected a list of homonyms. For instance, “Paris” as the capital of France, a film, the mythological figure, and a genius of flowering plants; or “Interpol” as the police organization and an American indie rock band. Second, we looked up for the categories of these homonyms in the taxonomy of categories of Wikipedia and manually extracted

²<http://jung.sourceforge.net>

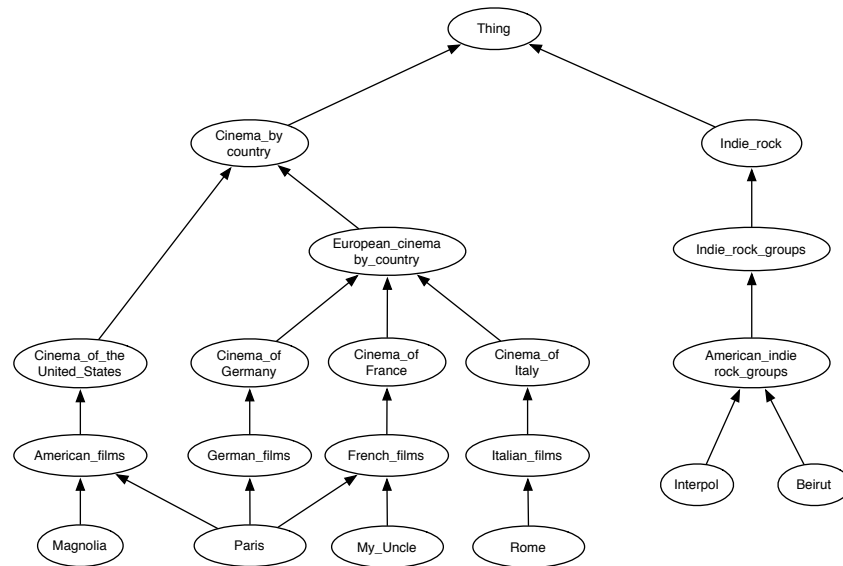


FIGURE 4.2: Ontology of cinema and music.

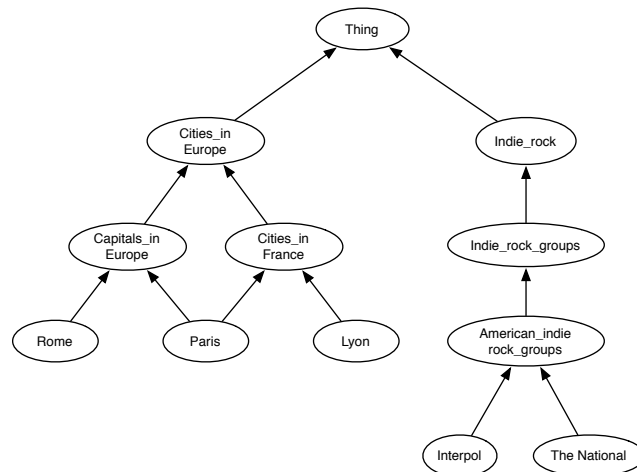


FIGURE 4.3: Ontology of cities and music.

some of their super-categories to construct the ontologies in a natural way. The homonyms are, thus, leaves (classes without proper sub-classes) of the ontologies. Two of the resulting ontologies are shown in Figure 4.2 and Figure 4.3. Any peer using the first one is interested in the topics of cinema and music, while the second is used by peers interested in cities and music. Notice that the classes *Paris* and *Rome* bring about homonymy between the two ontologies.

Construction of the reference populated ontologies Each class of the 5 ontologies has been populated with URLs to form the reference populated ontologies. For this, we used the URLs of the web pages associated to the Wikipedia category of each class, and also others returned by Google. These were duplicated (but expressed in a way that they remain syntactically different)

Matcher	Precision	Recall	F-measure
TaxoMap	0.7	0.7	0.7
Falcon	0.6	0.7	0.6461
Aflood	0.5	0.9	0.6428
Aroma	0.3	0.8	0.4363
Random	0.03	0.2	0.0521

TABLE 4.1: Evaluation of the matchers' alignments.

so as to ensure that the most specific classes (the leaves) have each one of them 50 instances at least, as well as each ontology 1000 instances at least.

Construction and peer assignment of initial populated ontologies Once a network of peers is generated, each peer is randomly assigned one of the 5 ontologies of topics. This is populated by removing instances from its corresponding reference populated ontology. The proportion of instances taken from each class is another parameter of our system. This process results in the construction of the initial populated ontologies. The idea is that, even if two peers share the same ontology of topics, they will share few instances before the interaction begins. In all our experiments this parameter was set to 10%.

Construction of initial alignments For the generation of initial alignments, we have chosen 4 matchers that regularly participate in OAEI [26] and launched them with the ontologies of topics. We have built reference alignments manually to evaluate the quality of the resulting alignments. Remember that reference alignments are taken as the preferred alignments (in practice given by domain experts) and that they are not known by peers. Table 4.1 shows the values of precision, recall and F-measure, on average, of the matchers' alignments with respect to the reference alignments. All of them are unsound and incomplete, which justifies the need of trust. As an example, *Paris* and *Rome* as cities happen to be identified with *Paris* and *Rome* as films, respectively. The matcher used is a parameter of our system that can be set to TaxoMap, Falcon, Aflood or Aroma. It can also be set to Random, which randomly assigns a class of one ontology to each class of the other, and identifies them via equivalence. The Random matcher serves as a baseline in our experiments.

Simulation of query answering For simulating query answering, we follow the discrete-event simulation approach [61], in which each event occurs at an instant in time and marks a change of state in the system. In our particular case, the state of the system is the list of all trust values, and an event embodies the process of randomly choosing a peer P_i in the network and a class $c \in C_i$ of its ontology, querying its acquaintances about this class via reformulation, and processing their answers and updating trust. Without loss of generality, we only select, for both querying and query reformulation, the classes included in the correspondences returned by

Parameter	Description
$N_{peers} \in \mathbb{N}$	number of peers in the network
$N_{queries} \in \mathbb{N}$	minimum number of queries per class and peer
$P_{oracle} \in [0, 1]$	maximum proportion of the sample about which the oracle can be called when estimating trust
$P_{initial} \in [0, 1]$	proportion of instances of the reference populated ontologies included in the initial populated ontologies
$\lambda_{trust} \in [0, 1]$	threshold of trust
$\lambda_{discard} \in [0, 1]$	threshold to discard instances
<i>matcher</i>	matcher used to compute initial alignments

TABLE 4.2: Parameters of our simulations.

the ontology matchers. As a parameter of our system, we consider, instead of the number of simulation steps, the minimum number of queries per class and peer. Once this parameter is set to a value, each simulation is interrupted at the first time t in which this value is exceeded.

In order to evaluate the gain in the quality of peer answers when query answering is guided by trust, we compare two different scenarios: a *naive* scenario in which each peer adds to its populated ontology all the answers to its queries, and a *trust-based* scenario in which answers are added on the basis of trust. More specifically,

- in the naive scenario, if P_i and $c \in C_i$ are selected at time t , and P_j is an acquaintance of P_i for whom there exists $d \in C_j$ such that $\langle c, d, = \rangle$ or $\langle c, d, > \rangle$ is in \mathcal{A}_{ij} , then $ext_j^t(d)$ is added to $ext_i^t(c)$;
- while in the trust-based scenario, $ext_j^t(d)$ is added only if the estimation of trust is above a threshold as explained in Section 4.3.2:

$$\widehat{trust}^t(P_i, P_j, \langle c, d \rangle) \geq \lambda_{trust}$$

The threshold of trust is another parameter of our simulations.

One of the key components of the estimation of trust is the user feedback or oracle calls. In Algorithm 1 an oracle is supposed to give a yes/no response to the question “ $a \in ext_i^*(c)$?” where a is in a sample taken from P_j ’s answer $ext_j^t(d)$. In our simulations this is done by simply looking into the reference populated ontology \mathcal{O}_i^* . As a parameter of our system, we also consider the maximum number of oracle calls permitted. Table 4.2 summarises all the parameters described so far and their abbreviations. The list is completed with the threshold to discard instances $\lambda_{discard}$ (see Section 4.3.3).

4.4.2 Experimental Results

We have conducted an experimentation with the aim at studying first the convergence of trust, and second the gain in the quality of peer answers when query answering is guided by trust. Moreover, we have performed a number of experiments to measure the impact of parameters such that the number of peers, the quality of initial alignments, and the thresholds of trust. In Section 4.4.2 we show and explain the results concerning the convergence of trust, whereas in Section 4.4.2 we present the results regarding the gain in query answering by using trust.

Convergence of trust In order to test the convergence of trust, we have to check that for each $P_i, P_j \in \mathcal{P}$ such that $\langle P_i, P_j \rangle \in \text{ACQ}$, and $c \in C_i$ and $d \in C_j$ for which there exists a relation R with $\langle c, d, R \rangle \in \mathcal{A}_{ij}$, the sequence

$$\{\widehat{\text{trust}}^t(P_i, P_j, \langle c, d \rangle)\}_{t \in \mathbb{N}}$$

is convergent. The natural limit is

$$\text{trust}^*(P_i, P_j, \langle c, d \rangle) =_{\text{def}} p(\text{ext}_i^*(c) | \text{ext}_j^*(d)) = \frac{|\text{ext}_i^*(c) \cap \text{ext}_j^*(d)|}{|\text{ext}_j^*(d)|}$$

Therefore, alternatively, we can check that the sequence $\{\Delta^t(P_i, P_j, \langle c, d \rangle)\}_{t \in \mathbb{N}}$ where

$$\Delta^t(P_i, P_j, \langle c, d \rangle) = |\widehat{\text{trust}}^t(P_i, P_j, \langle c, d \rangle) - \text{trust}^*(P_i, P_j, \langle c, d \rangle)|$$

converges to 0.

We have checked convergence of trust experimentally. For this, we ran 50 simulations. Each simulation stopped at $t_0 + 10$ where $t_0 \in \mathbb{N}$ was the first instant of time (number of queries) for which $\Delta^t(P_i, P_j, \langle c, d \rangle) < 0.001$ for every $t \in [t_0, t_0 + 10]$ and P_i, P_j and $\langle c, d \rangle$ as specified above. The average t_0 over the 50 simulations was $t_0 = 40$. In all the simulations, *matcher* was set to TaxoMap, which is the matcher with best F-measure value on the initial alignments (see Table 4.1) and $N_{\text{peers}} = 100$, $P_{\text{oracle}} = 0.15$, $P_{\text{initial}} = 0.1$ and $\lambda_{\text{trust}} (= \lambda_{\text{discard}}) = 0.5$.

We have performed three experiments to evaluate the impact of number of peers (Exp. 1), thresholds of trust (Exp. 2) and quality of initial alignments (Exp. 3) on the speed of convergence of trust. Table 4.3 includes the values given to the parameters in each experiment. We ran 50 simulations with each parameter configuration. Notice that we let N_{peers} , $\lambda_{\text{trust}} (= \lambda_{\text{discard}})$ and *matcher* vary, respectively, in Exp. 1, Exp. 2 and Exp. 3, whereas $N_{\text{queries}} = 30$, $P_{\text{oracle}} = 0.15$ and $P_{\text{initial}} = 0.1$ in all the experiments. In Exp. 1 and Exp. 2, *matcher* was set to TaxoMap. The thresholds of trust were set to 0.5 in Exp. 1 and Exp. 3, while the number of peers was set to 100 in Exp. 2 and Exp. 3. The same parameter configurations were used in analogous

Parameter	Exps. 1 & 1'	Exps. 2 & 2'	Exps. 3 & 3'
N_{peers}	25, 100, 125, 400, 625, 900	100	100
$N_{queries}$	30	30	30
P_{oracle}	15%	15%	15%
$P_{initial}$	10%	10%	10%
$\lambda_{trust} = \lambda_{discard}$	0.5	0.2, 0.4, 0.6, 0.8	0.5
<i>matcher</i>	TaxoMap	TaxoMap	Reference, Random, Aroma, TaxoMap, Falcon, Aflood

TABLE 4.3: Values of the parameters in the experiments of convergence of trust and gain in query answering by using trust.

experiments Exp. 1', Exp. 2' and Exp. 3' concerning the study of the gain in the quality of peer answers when using trust (Section 4.4.2).

For each experiment, we recorded the average $\Delta^t(P_i, P_j, \langle c, d \rangle)$ over the 50 simulations of each parameter configuration. We focus on results concerning pairs of classes for which the speed of convergence is slower since the impact of the parameters is more evident. This “worst case” is represented by the homonymous classes **Paris** with five different meanings, as no matcher was able to overcome this homonymy problem. Thus, we look at the average

$$\Delta^t(\mathbf{Paris}) = \frac{1}{|\text{ACQ}_{\mathbf{Paris}}|} \sum_{\langle i, j \rangle \in \text{ACQ}_{\mathbf{Paris}}} \Delta^t(P_i, P_j, \langle \mathbf{Paris}, \mathbf{Paris} \rangle)$$

where $\text{ACQ}_{\mathbf{Paris}} = \{\langle i, j \rangle \in \text{ACQ} : \langle \mathbf{Paris}, \mathbf{Paris} \rangle \in \mathcal{A}_{ij}\}$.

Figure 4.4 shows the results of Exp. 1. We can see that the number of peers actually has no significant impact on the speed of convergence. Further, note that only 10 queries are necessary for $\Delta^t(\mathbf{Paris}) < 0.1$. This may be explained by the underlying small-world topology of the network.

Results of Exp. 2 are shown in Figure 4.5. We can see that the thresholds of trust have an influence on the speed of convergence of trust: the lower they are, the slower convergence is. The reason behind this is that with low thresholds of trust even trustworthy peers are likely to accept unsatisfactory instances, which require time to be discarded, and until this happens trust values diverge.

Results of Exp. 3 are presented in Figure 4.6. Recall that we concentrate on the homonymous classes **Paris**. The initial alignments appear in the legend of Figure 4.6 in order of descending F-measure according to **Paris**. We can see that this order is preserved for convergence of trust: the higher the F-measure an initial alignment has, the faster convergence is. The extreme case is that of the reference alignment with which convergence is reached from the very first query.

We also have performed experiments to assess the impact on convergence of trust of the proportion of instances of peers' reference populated ontologies included in their initial populated

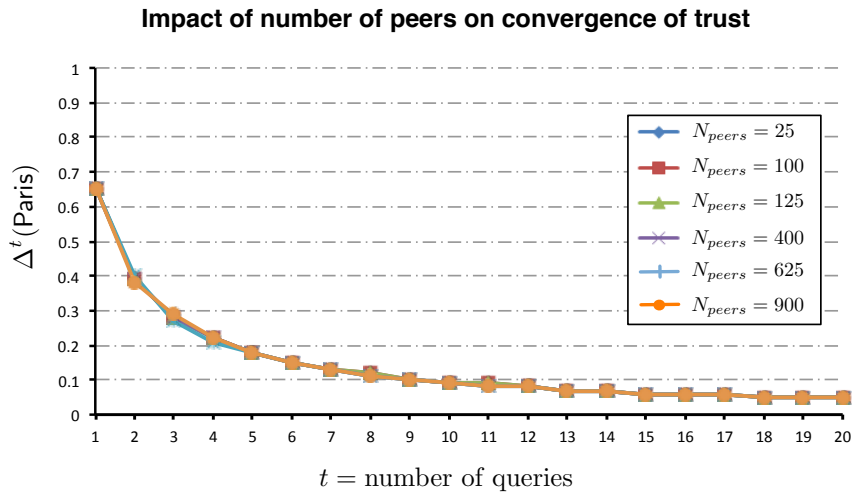


FIGURE 4.4: The number of peers has no significant impact on the speed of convergence.

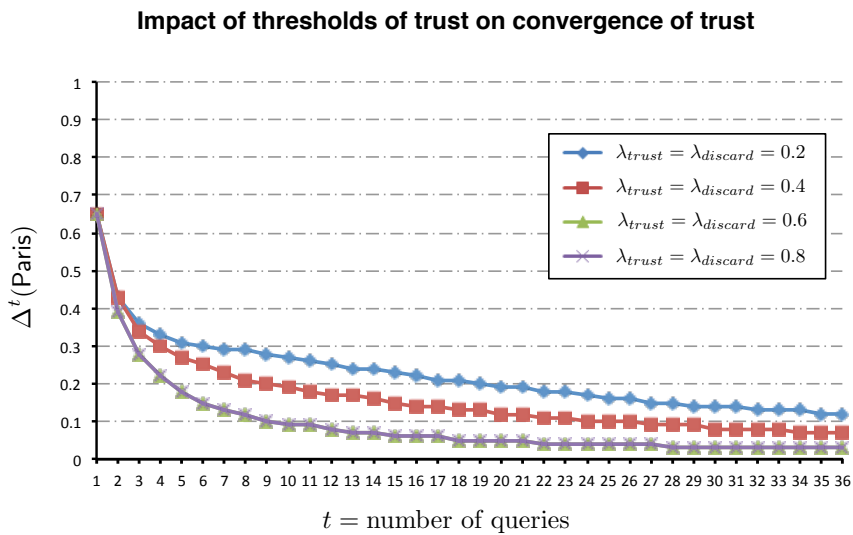


FIGURE 4.5: The lower thresholds of trust are, the slower convergence of trust is.

ontologies, and the maximum proportion of the samples of peers' answers to be processed by an oracle — denoted by $P_{initial}$ and P_{oracle} in Table 4.2. The higher $P_{initial}$ or P_{oracle} are, the faster convergence of trust is. This is not surprising since, the higher $P_{initial}$ or P_{oracle} are, the more instances peers are 100% sure of, which allows to make better estimations of trust.

Gain in the quality of peer answers when using trust In order to assess the gain in the quality of peer answers when query answering is guided by trust, we have compared a trust-based scenario with a naive scenario (see Section 4.4.1). This gain is measured with standard precision, recall and F-measure: if c is a class of P_i 's ontology then

$$precision_i^t(c) = \frac{|ext_i^t(c) \cap ext_i^*(c)|}{|ext_i^t(c)|} \quad recall_i^t(c) = \frac{|ext_i^t(c) \cap ext_i^*(c)|}{|ext_i^*(c)|}$$

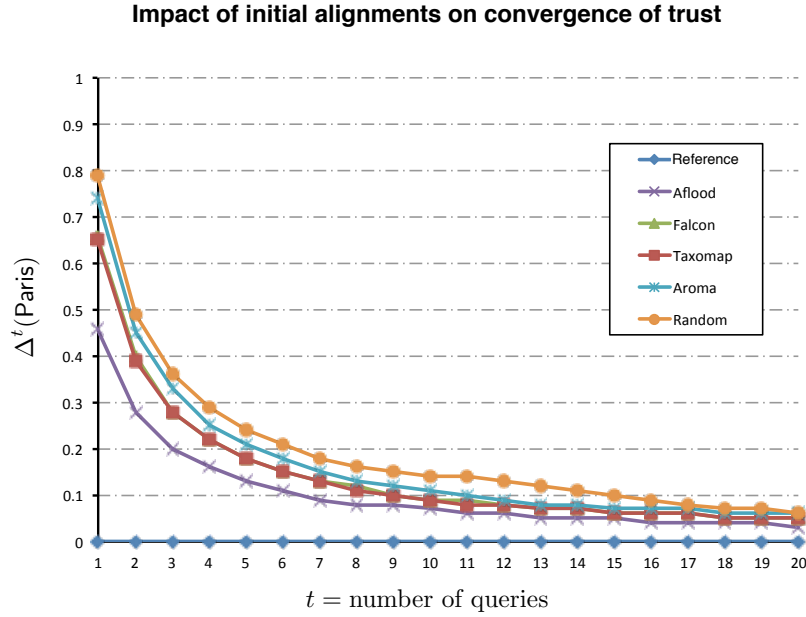


FIGURE 4.6: The higher the F-measure an initial alignment has, the faster convergence is.

and F-measure is the harmonic mean of precision and recall.

We ran 50 simulations of the two scenarios and recorded in each of the cases $precision_i^t(c)$ and $recall_i^t(c)$ for every class c of each peer P_i 's ontology. As in the case of convergence of trust, in all the simulations *matcher* was TaxoMap and $N_{peers} = 100$, $P_{oracle} = 0.15$, $P_{initial} = 0.1$ and $\lambda_{trust} (= \lambda_{discard}) = 0.5$. The average precision and recall over all classes and peers in the generated network in the trust-based scenario after 5 queries were, resp., 0.99 and 0.68 while in the naive scenario were 0.80 and 0.75. We can confirm that precision is higher in the trust-based scenario. Yet, precision is not low in the naive scenario. The reason is that TaxoMap only made mistakes when matching homonymy classes.

We have also performed experiments with the aim to evaluate the impact of number of peers (Exp. 1'), thresholds of trust (Exp. 2'), together with the quality of initial alignments (Exp. 3'), on the quality of peer answers when using trust. We ran 50 simulations of the trust-based scenario with each parameter configuration as specified in Table 4.3. Regarding the experimental results, we focus on the class **Paris** which, as argued before, is the “worst case” in our setting. Specifically, we show the average of precision, recall and F-measure over all peers whose ontologies contain the class **Paris**. For instance, in the case of precision we compute

$$precision^t(\mathbf{Paris}) = \frac{1}{|\mathcal{P}_{\mathbf{Paris}}|} \sum_{i \in \mathcal{P}_{\mathbf{Paris}}} precision_i^t(\mathbf{Paris})$$

where $\mathcal{P}_{\mathbf{Paris}} = \{i \in \mathcal{P} : \mathbf{Paris} \in O_i\}$.

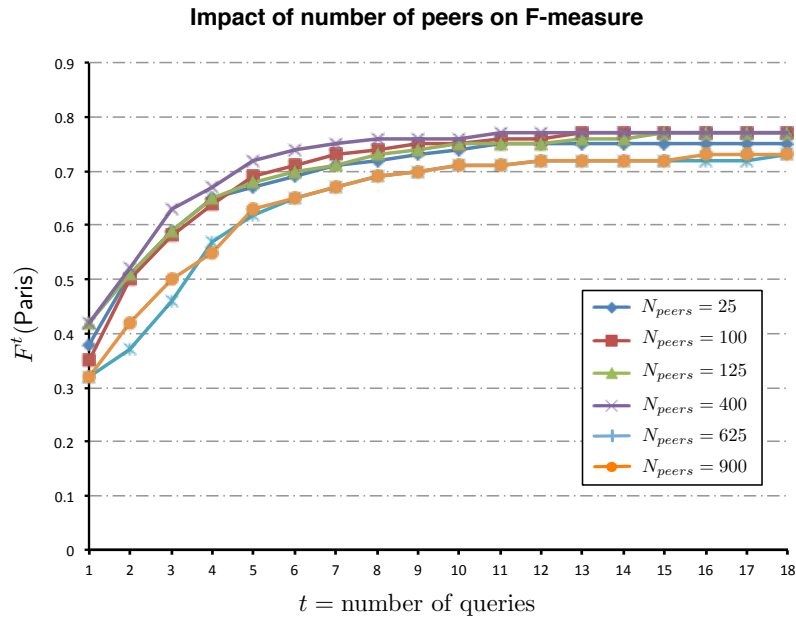


FIGURE 4.7: The number of peers has no significant impact on $F^t(\text{Paris})$.

Figure 4.7 shows the F-measure values in Exp. 1'. Again, we can see that the number of peers have no impact on the gain in query answering.

Figures 4.8 and 4.9 show, resp., precision and recall in Exp. 2'. We can see that low thresholds of trust ensure high recall but a high number of queries is needed to obtain high precision. On the contrary, high precision is guaranteed with high thresholds of trust at the expense of low recall. The extreme case is $\lambda_{trust} = \lambda_{discard} = 1$ with which peers do not accept any answer. Since our approach is based on bayesian inference, trust might be very close but never equal to 1. In practice, high and low thresholds of trust can be used to model peers with restrictive and open behavioral, respectively.

Figure 4.10 shows the F-measure values in Exp. 3'. As in the case of trust convergence, the higher the F-measure an initial alignment has, the higher $F^t(\text{Paris})$ is.

To conclude this section, we also have performed experiments to assess the impact on the quality of peers' answers of the proportion $P_{initial}$ of instances of peers' reference populated ontologies included in their initial populated ontologies, and the maximum proportion P_{oracle} of the samples of answers to be processed by an oracle. As expected, the higher $P_{initial}$ or P_{oracle} are, the higher $F^t(\text{Paris})$ is.

4.5 Conclusions and Related Work

In the most general form, trust is seen as a “method for dealing with uncertainty” [56]. In this chapter, we have presented a novel approach to trust in the context of networks of ontologies

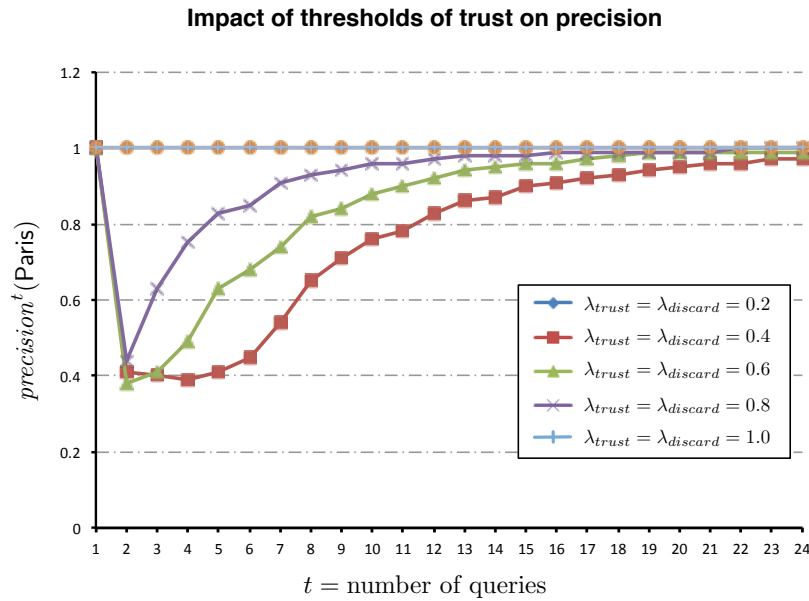


FIGURE 4.8: The higher thresholds of trust are, the higher $precision^t(\text{Paris})$ is.

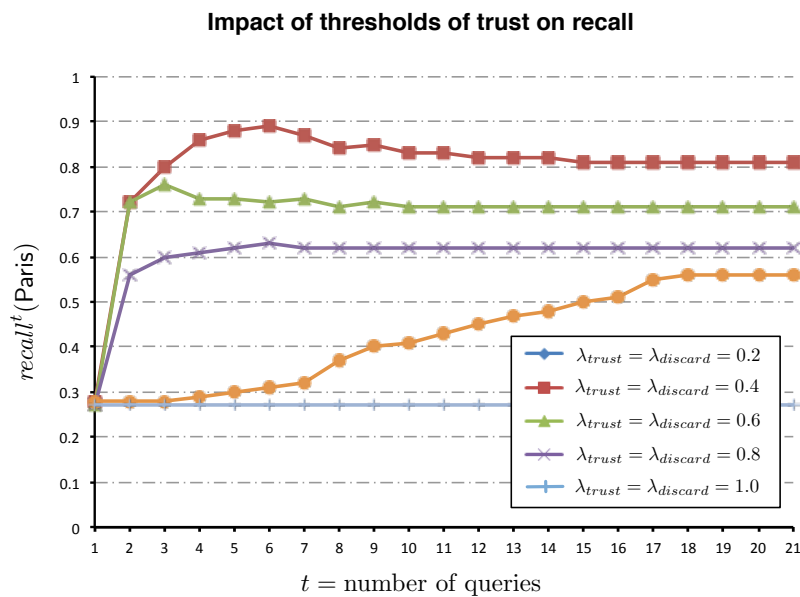


FIGURE 4.9: The lower thresholds of trust are, the higher $recall^t(\text{Paris})$ is.

and alignments. Since ontology matchers may fail in computing sound and complete alignments between peers' ontologies, query translation based on alignments may lead to answers which are uncertain to be satisfactory. Our mechanism of trust can be used by peers to find the peers in the network that are better suited to answer their queries

This work falls into the intersection of two active research areas in the context of the semantic web, namely, trust and ontology matching. Below we summarize related work in both areas.

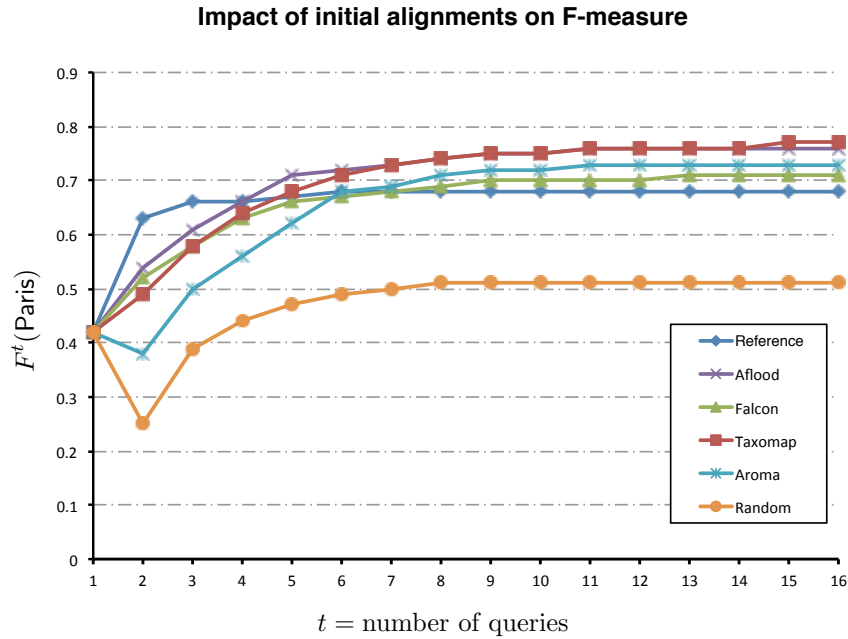


FIGURE 4.10: The higher the F-measure an initial alignment has, the higher $F^t(\text{Paris})$ is.

4.5.1 Trust

Trust decisions need to be made in many different scenarios of computer science, e.g. concerning the reliability of a computer network, the fulfillment of an e-commerce transaction, or the accuracy of some information on the web. According to Artz and Gil [9], though, there is a unifying theme: “trust is only worth modeling when there is a possibility of deception, that is, when there is a chance of a different outcome than what is expected or has been agreed upon”.

In the context of the semantic web, trust is taken as a necessary condition to assimilate knowledge coming from ontologies, rules and proofs [15]. The semantic web, just as the web, is an open environment without norms where the quality of the information a source may offer is not guaranteed. Trust mechanisms are aimed at assisting agents to choose services or information sources when performing a task, and automated reasoners to discriminate between different information sources when answering query. In this section we report, in the presence of comprehensive surveys such that [9, 41, 59, 62], a number of these mechanisms and relate them with our approach.

Most research on trust and reputation focuses on decentralized solutions where individuals, peers, agents are empowered to take trust decisions rather than relying on a centralized process. Furthermore, trust is commonly agreed to be subjective, so that trust towards a particular entity may differ from one entity to another. In line with this, Richardson et al. [60] have proposed a generalization of the algorithm PageRank [22] to the semantic web. The aim is to establish degrees of belief in statements asserted by one or more sources, and the assumption is that a user’s belief in a statement depends on her trust in the sources providing it. The authors

conceive a web of trust in which each user provides personal trust values for a small number of other users. The proposed algorithm finds paths from a user and every user with a personal belief in the statement in question. Then, it concatenates the trust values along each path to obtain recommended believes in the statement. All these resulting values are aggregated to obtain a final degree of belief of the statement. Richardson et al. do not go into detail about what statements and trust values represent. In our approach, we specifically address the problem of semantic heterogeneity. Statements are of the form “an instance a is satisfactory”, and trust values estimate the quality of query translations modeled as conditional probabilities. Concatenation of trust values cannot be applied as transitivity does not hold for probabilistic subsumptions. For this reason, only paths of length 1 are considered. Our aggregation function, though, resembles that of Richardson et al. Moreover, we also consider the possibility of using different functions (maximum, minimum, mean).

Some existing approaches to trust take into account the context in which trust judgements are to be made. This is the case of Abdul-Rahman and Hailes’ approach [1], in which trustworthiness of agents is determined on the basis of agents’ collected statistics on direct experiences (direct trust), as well as recommendations from other agents (recommender trust). Both direct and recommender trusts are given within a certain context. Using the example given by the authors, Alice may trust a specific agent Bob the Mechanic in the specific context of servicing her car but not in the context of babysitting her children. Abdul-Rahman and Hailes models context as an unstructured set. Alice’s two trust judgements, e.g. can be represented by the predicates $t(\text{Bob_the_Mechanic}, \text{Car_Repairing}, t)$ and $t(\text{Bob_the_Mechanic}, \text{Babysitting}, u)$ in which t and u stand for ‘trustworthy’ and ‘untrustworthy’, respectively. In our case, context dependency is realized in terms of query translations. For example, in the setting of the presented experimentation, it may happen that $\text{trust}^t(P_i, P_j, \text{Interpol})$ is above the threshold of trust while $\text{trust}^t(P_i, P_j, \text{Paris})$ is below the threshold.

There exist many approaches that resort to probability theory to model trust and that propose Bayesian inference for estimating trust values. One of the earliest is Mui et al.’s approach [50]. In this model, trust is defined as a subjective expectation an agent has about another’s future behavior based on the history of their encounters, which are of a dyadic nature (either cooperative or not). In mathematical terms, trust is defined as the expected value of the proportion of encounters which are cooperative, modeled as a Beta distribution, and updated whenever new encounters occur. In Mui et al.’s work, as in most of the Bayesian models of trust, the lack of previous encounters is modeled with a uniform distribution, which typically represents ignorance of prior probabilities. However, in our approach the lack of direct experience is compensated with the information provided by alignments, as we build informative prior distributions based on the correspondences of the alignments between peers’ ontologies.

Provenance refers to the origins of something. For the W3C Provenance Working Group, provenance records “contain descriptions of the entities and activities involved in producing and delivering or otherwise influencing a given object”.³ Golbeck has proposed an approach that uses provenance and trust for filtering semantic web content in social networks [32]. First, an algorithm infers trust relationships between individuals. This algorithm uses provenance of trust annotations to compute more accurate trust values based on the idea of transitivity of trust. Then, trust values are used in conjunction with provenance of statements to filter content to users. In our approach the provenance of an instance is not global, but it depends on each peer, and represents a peer’s knowledge about the instance belonging to which other peers’ populated ontologies. Provenance allows us to compute the probability of instances to be satisfactory.

4.5.2 Ontology and Schema Matching

There exists a considerable number of ontology and schema matching systems available today [27, 42]. In what follows, we review systems which share with us some features, specifically, systems which return probabilistic mappings, systems which exploit user feedback, and instance-based matching systems.

User involvement is a feature in many matching systems. A user, or a community of users, can participate in the process of matching in different ways, for instance, by specifying system parameters, by providing an initial alignment — which can just be a set of expected correspondences — or by validating and enhancing output alignments. In Pereira et al.’s approach users are asked to identify incorrect correspondences at a final step of the matching process [57]. The confidence of these correspondences is then set to 0.0. This information is taken as true and can be used in future matching operations. In this sense, users are always trusted. McCann et al. have proposed an approach that benefits from users feedback during the matching process [47]. Users are asked simple questions with the aim at improving matching accuracy (e.g. a positive answer to the question “is `monthly-fee-rate` of type `DATE`?” suggests the use of a specialized date matcher for this particular attribute). Three types of questions are considered: (i) questions to verify intermediate predictions, (ii) questions to learn domain constraints and (iii) questions to verify final matching predictions. Unlike in Pachêco et al.’s approach, users are classified into trusted and untrusted. To figure out whether a user is trustworthy or not, a set of evaluation questions (from which the answers are known) is given to users. In our approach, user feedback is used for estimating trust on peers. Users act as oracles that give a yes/no answer to questions of the form “is *a* an instance of class *c*?”. Users are trusted and their answers are always considered to be true.

³<http://www.w3.org/2011/prov/wiki/>.

Different probabilistic approaches have been applied to ontology and schema matching, such as Bayesian networks [49], Markov networks [8], or Markov logic networks [55]. Mitra et al. has proposed the use of Bayesian networks to derive new correspondences from a set of initial correspondences (typically given by an expert or an automatic tool) [49]. A Bayesian network is built with the help of meta-rules which express how initial correspondences affect other correspondences. These meta-rules are based on the semantics of the languages of the ontologies to be aligned. An example of a meta-rule is “if two properties and their domains match, then, with some certainty degree, their ranges also match”. Initial correspondences are translated into prior probability distributions which, along with meta-rules, allow to infer probability distributions for other correspondences. Rather than Bayesian networks, we follow a Bayesian approach to statistical inference in order to estimate trust. Like Mitra et al. we translate correspondences of initial alignments into prior probability distributions, which are later refined when processing instances included in peers’ answers. For this, we profit not only from user feedback, but also from local ontology reasoning for limiting the recourse to user feedback. Therefore, any instance added to an ontology as a consequence of an estimation of trust affects later trust estimations even concerning new queries since this instance will have an impact on the local ontological reasoning carried out for these estimations.

ProbaMap is an example of an instance-based matching system [66]. This system implements an algorithm which automatically discovers probabilistic inclusion mappings between classes of two populated taxonomies. In order to estimate probabilities of mappings, the authors follow a Bayesian approach to statistics which exploits content descriptions of instances categorized in each of the taxonomies under consideration. This comes down to determining how the instances of a class are classified in another class on the basis of their content descriptions. For this purpose, different classifiers (like Naive Bayes or decision trees) can be employed. The algorithm then returns mappings whose probability is greater than a given threshold. Monotonicity of the probability function is critical to avoid probability estimation of as many mappings as possible. The authors consider two different instance-based ways of modeling probabilistic mappings which guarantee monotonicity. One modelization is based on conditional probabilities, which is the one that we also have chosen to follow. Our approach is more adapted to a peer-to-peer setting than ProbaMap or classical instance-based matching systems. These systems very much depend on the availability of instances to compute sound and complete alignments. Nevertheless, in a semantic P2P network, data is distributed among peers, and instances are gradually acquired by peers as a result of a query-answering process. Therefore, when peers start interacting initial matcher-computed alignments may be unsound and incomplete, and query-translation based on these alignments may lead peers to populate their ontologies with unsatisfactory instances. Moreover, unsatisfactory instances may be due to some sort of malicious behavior by peers. Our approach helps to identify and discard these instances by computing their probability to be satisfactory on the basis of the trust towards the peers that provide them. In this

computation we profit from provenance information.

In general, in a highly decentralized and dynamic environment such as a peer-to-peer network, where data is distributed among peers, and peers can join and leave the network at their will, it is more adequate to concentrate on ontology fragments involved in queries peers are interested in, and to identify the peers that better answer these queries — and classify them as trustworthy peers — for future interactions, in a dynamic approach to ontology matching *`a la trust*. In this way, we reduce the costly operation of matching complete ontologies whenever new data is exchanged or new peers appear.

Chapter 5

Demo: TrustMe I got what you mean

5.1 Introduction

Virtual online communities (social networks, wikis...) are becoming the major usage of the web. The freedom they give to publish and access information is attracting many web users. However, this freedom is filling up the web with varied information and viewpoints. This raises important issues that concern privacy and trust. Due to their decentralised nature peer-to-peer (P2P) systems provide a partial solution for the privacy problem: each user (peer) can keep control on her own data by storing it locally and by deciding the access they want to give to other peers. We focus on semantic P2P systems in which peers annotate their resources (documents, videos, photos, services) using ontologies.

Indexing resources using the terms of an ontology enables more accurate information retrieval and query answering than indexing by keywords of a textual annotation. In particular, it is a way to overcome the problems raised by homonymy in classical keyword search engines. As an example, the word "Paris" corresponds to (at least) four meanings: a city, a film, a mythological figure, and a genus of flowering plants. Classical keyword search engines such as Google or Yahoo! returns to a keyword query "Paris" a mixture of web pages referring to its different meanings. In contrast, using as query interface an ontology in which a class named "Paris" is a subclass of a class named "Capitals" would remove the ambiguity on the interest of a user if she clicked on that particular class.

In semantic P2P systems, every peer is autonomous and free to organise her local resources as instances of classes of her own ontology serving as a query interface for other peers. Alignments between ontologies are required to reformulate queries from one local peer's vocabulary to another. Currently there exists a considerable amount of matchers available for computing

alignments automatically. Nonetheless, automatic matchers may fail to compute sound and complete semantic alignments, and thus making query reformulation not fully reliable. Still a trust mechanism can help peers to find the most reliable sources of information within the network.

This demo builds on the trust mechanism presented before in Chapter 4. Specifically, this mechanism allows to estimate the probability that a peer will provide a satisfactory answer to a query posed by another peer. This probability is taken as the *trust* that the addressing peer has towards the addressed peer, and it is subject to the posed query. Unlike other existing approaches to trust, unsatisfactory answers are seen as the result of peers' incapacity to understand each other. Trust values are refined over time as more queries are sent and answers received in a Bayesian learning process in which alignments are used to build initial prior distributions. As a by-product this trust mechanism provides the means to rank, in the scope of a particular class of a peer's ontology, the set of received instances according to their probability to be satisfactory. For this, a function aggregates the trust values of all the peers that returned these instances.

5.2 Goals of the Demo

TrustMe demonstrates the use of our trust mechanism, presented in Chapter 4, for guiding query answering in semantic P2P social communities. Our view is that semantic P2P social communities illustrate the evolution of the semantic web towards a social web. We have decided to reproduce this vision via a semantic P2P bookmarking network in which peers exchange URLs of articles (web pages) about the topics they are interested in. Unlike current bookmarking systems (e.g. Delicious), in this ideal P2P bookmarking network, information is no longer centralised, and peers need to query other peers to gather new articles. Each peer uses her own taxonomy of categories for tagging articles. These taxonomies can be seen as ontologies so that each category C in a peer P 's taxonomy is a class whose instances are URLs of articles initially indexed by C or by a subcategory (subclass) of C in the taxonomy. Then the set of instances of C can be gradually enriched by adding instances of classes D that belong to the taxonomies of P 's acquainted peers and that happen to be aligned with C .

In this demo we highlight the gain in the quality of peers' answers —measured with precision and recall—when the process of query answering is guided by our trust mechanism presented in Chapter 4. In particular, we show how trust overcomes the problem of homonymy, still a constant battle for ontology matchers. Further, a trust-based ranking of instances allows to distinguish those that are relevant to a class from those related to its homonymous classes.

5.3 Setting Up the P2P Bookmarking Scenario

Below we explain the generation of the taxonomies we have used for the semantic P2P bookmarking scenario, the alignments between these taxonomies, and the topology of the network.

Taxonomy Generation. First, we selected a number of homonyms (e.g. “Paris” and “Rome”) in different topics (e.g. cinema and cities). These homonyms are the names of some of the most specific classes (the leaves) of the peers’ taxonomies. We built up the rest of the taxonomies manually by looking up for Wikipedia categories organised in super-category and sub-category relations. In total, 5 taxonomies were generated. For example, one of the taxonomies is used by peers interested in cinema and music, and contains a category “Paris” which corresponds to a film. Another taxonomy is used by peers interested in cities and music, and contains a category “Paris” corresponding to the capital of France.

Alignment Generation. For generating alignments, we chose four matchers that regularly participate in OAEI [26] and launched them with the peers’ taxonomies. From the resulting alignments, we selected the ones with best F-measure values according to manually computed reference alignments.

The selected alignments happened to be both unsound and incomplete. In particular, matchers did not get over the homonymy of the taxonomies. For instance, the class `Paris` corresponding to a film was aligned with the class

Network Generation. Social networks are well-known to exhibit small-world characteristics. In the demo we generate small-world topologies that contain up to 1000 peers. Each peer is randomly associated with one taxonomy to reproduce the fact that acquainted peers may or not share the same topics of interest.

Taxonomy Population. For each category/class of each taxonomy we first built a reference set made up of the URLs of all the category’s associated articles in Wikipedia, and other related articles returned by Google that are manually checked to be relevant. Then, we populate each peer’s “copy” of the category by selecting randomly an initial set that contains 10% of the reference URLs. URLs in this initial set are the only URLs that are known for the peer while the reference set is not known for the peer and it is used only for evaluating the results and to simulate the user feedback. In order to ensure that every leaf had 50 articles at least, we had to duplicate the web pages of the starting set (although they were identified with different URLs). This also guaranteed that each peer’s taxonomy had at least 1000 articles.

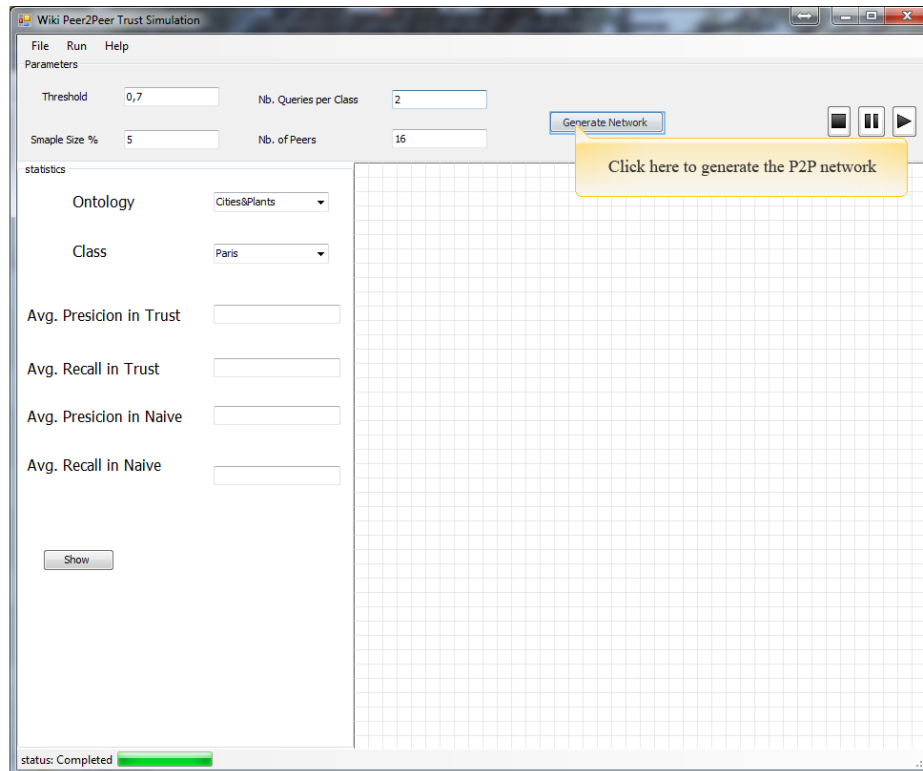


FIGURE 5.1: A snapshot of the Main screen of TrustMe

5.4 TrustMe

In TrustMe we compare two scenarios: a *naive* scenario in which peers query all their acquaintances through query reformulation based on alignments, and a *trust-based* scenario in which peers compute trust values using our trust mechanism, presented in Chapter 4, and only accept answers from their trustworthy acquaintances. More specifically, if a peer P_i is interested in gathering new articles about a topic specified by the category C in her taxonomy at time t ,

- in the naive scenario, if the correspondence $\langle C, D, = \rangle$ is in the alignment between P_i 's and P_j 's taxonomies, P_i will send the query $Q = D(X)?$ to P_j , and P_j 's answer will be added to P_i 's taxonomy, but
- in the trust-based scenario, P_j 's answer will not be added unless the trust that P_i has towards P_j w.r.t the translation $\langle C, D \rangle$ at t , is greater than a given trust threshold.

To compare the two scenarios we use precision and recall. Moreover, we evaluate the probability of articles to be relevant to the categories they belong to. This provides the means for ranking articles within categories.

When TrustMe is lunched the main screen depicted in Figure 5.1 appears to the user. Using this screen the user can set up the values of several parameters: number of peers, trust threshold

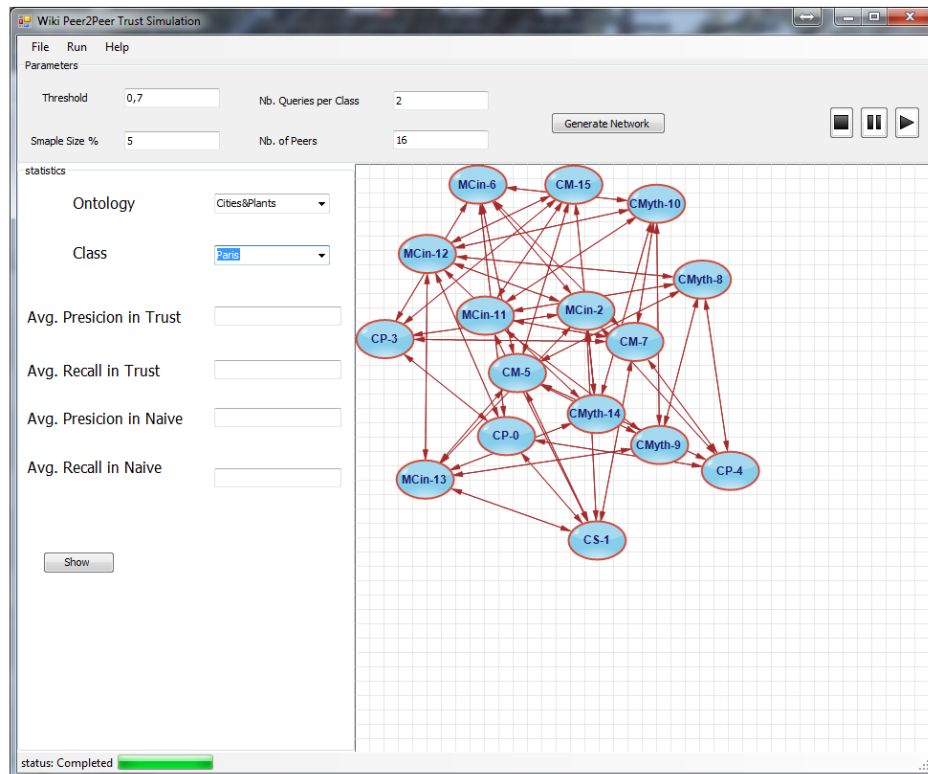


FIGURE 5.2: Snapshot showing the topology of a generated network with 16 peers.

and the size of the samples used when performing Bayesian learning in our trust mechanism, presented in Chapter 4, and the minimum number of generated queries during the simulation per class and peer.

After setting-up the required parameters the user should press the "Generate Network" button to generate the semantic P2P network. TrustMe shows the topology of the generated network to the user as depicted in Figure 5.2 if the number of peers is less than 50.

Pressing the play button runs the demo until the correspondences of all the alignments in the network are randomly selected a desired number of times set by the user in the "NB. Queries per Class" field. Figure 5.3 shows a snapshot of TrustMe when the user chose one peer from the network after running the demo. It shows the returned ranked results of querying the peer's neighbours about a category. More specifically, it presents the added articles to Paris, the capital of France, in the taxonomy of the chosen peer in both scenarios (trust based scenario's results are in the upper left while the naive scenario's results are in the lower left). We clicked on one of the trust based results and one of the naive ones. The former resulted in the Wikipedia article of Paris, and the latter in the Wikipedia article of the film "Paris". Also, the demo shows the trust values over time peer has towards her neighbours in the upper right graphics. The graphics shows 2 trustworthy neighbours against 3 untrustworthy ones. In the lower right of the screen the demo shows the precision and recall for that particular peer in both naive and trust-based scenario.

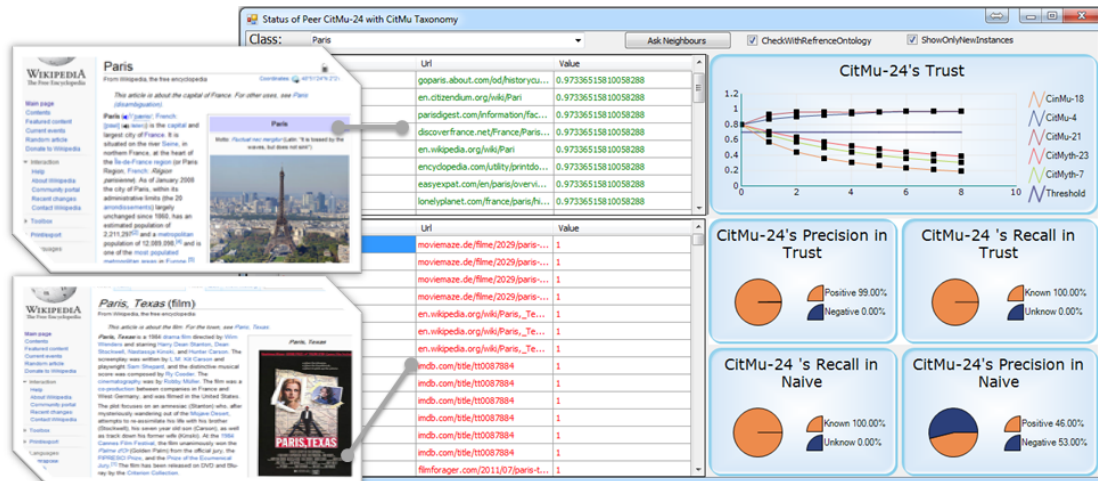


FIGURE 5.3: Snapshot showing the added articles in Paris city class of one peer's taxonomy.

Table 5.1 presents the average values of precision and recall for classes about four meanings of Paris in the two scenarios. These means were computed over the set of all peers sharing each of the meanings. To compute them we used the values: number of peers = 100, threshold = 0.7, size of the samples=15%. We ran until the correspondences of all the alignments in the network were selected randomly 5 times. The results show that our trust mechanism, presented in Chapter 4, guarantees high precision. On the other hand, as one could expect, recall is higher in the naive scenario.

category/class	Trust Scenario		Naive Scenario	
	Precision	Recall	Precision	Recall
Paris (city)	88%	64%	48%	96%
Paris (film)	85%	80%	27%	100%
Paris (mythology)	100%	66%	14%	98%
Paris (plant)	80%	97%	16%	100%

TABLE 5.1: Comparing Precision and recall averaged values for Paris homonyms.

Conclusion

Chapter 6

Conclusion and Future Works

In this thesis we have addressed two main research problems in the Semantic Web: the data linkage problem in the setting of Linked Data and the problem of modeling and computing trust in Semantic P2P Networks. In this chapter we give a summary of the primary contributions of in each of these research problems and expected future works. Figure 6.1 gives an overview of these contributions.

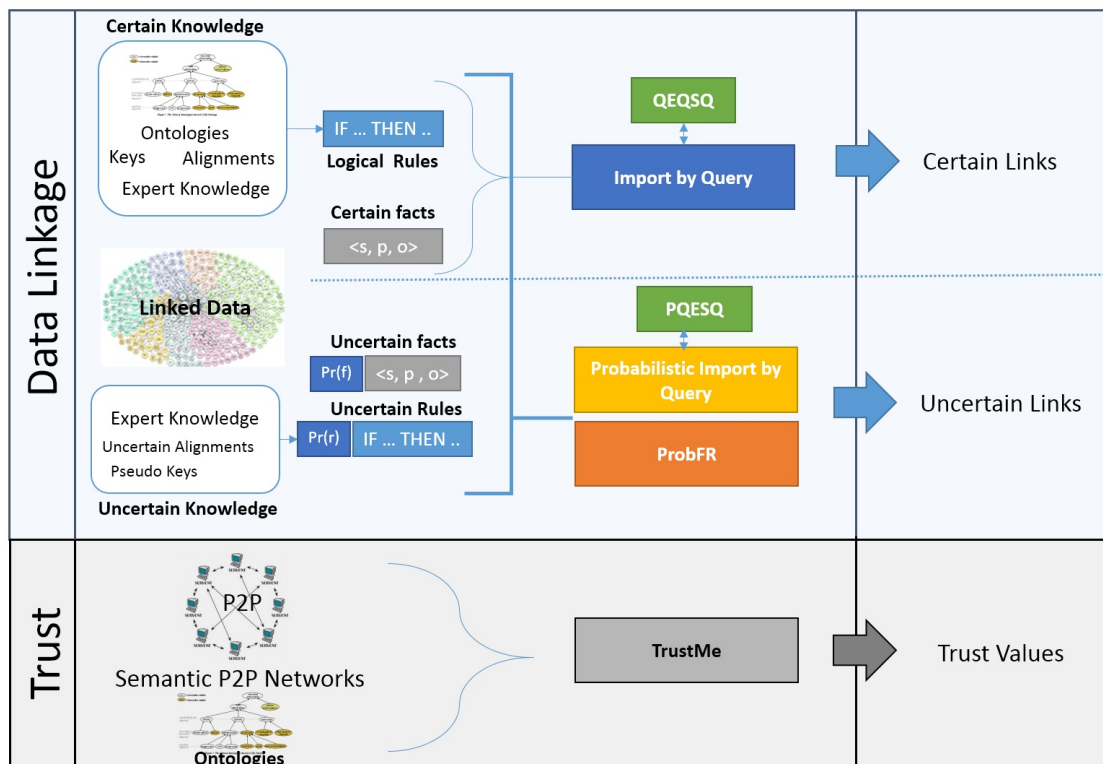


FIGURE 6.1: Summary of the contributions

6.1 Data Linkage

Data linkage is a crucial task in Linked Data. In particular, it is very important to correctly decide whether two URIs refer to the same real-world entity or not. These two URIs may appear in two different datasets in the Linked Data cloud or in the same dataset.

In this thesis, we have modeled the data linkage problem in Linked Data as a reasoning problem over possibly decentralized data. This reasoning is done using logical rules that capture in a uniform way different certain knowledge that may be available about the data. In particular, they capture ontological constraints (e.g. functional properties, keys, ...), ontology alignments and rules on the target domain.

We have described a novel import-by-query algorithm that alternates steps of sub-query rewriting and of tailored querying the Linked Data cloud in order to import data as specific as possible for inferring or contradicting given target same-as facts. Contrarily to other existing approaches, our approach do not do a global import (of a whole RDF graph of reference dataset in Linked Data such as DBpedia) to complete the local data. Such full data enrichment is not feasible in practice, as it may be very costly in terms of memory space and processing.

Experiments conducted on a real-world dataset have demonstrated the feasibility of this approach and its usefulness in practice for data linkage and disambiguation.

Furthermore, We have proposed an adaptation of our rule-based approach for data linkage to deal with uncertain facts and rules. In this adaptation we modeled uncertainty of facts and rules as probability values. We have proposed a fact propagation algorithm (ProbFR) that computes not only the set of facts that can be inferred from a given set of facts and set of rules but also their probabilistic weights. Our experiments have showed that our algorithm scales to large data sets and produces meaningful probabilistic weights.

In addition, we have proposed an extension of our import-by-query algorithm that deals with uncertainty called probabilistic import-by-query algorithm. It alternates steps of sub-query rewriting based on backward chaining and of external query evaluation. It returns as output not only a (true or false) answer as its predecessor but a probabilistic weight greater than or equal to a given threshold.

Future Works

As future works we plan to do more comprehensive experiments of our import-by-query and probabilistic import-by-query algorithms using referenced datasets in addition to the INA dataset, DBpedia.fr and DBpedia.org used so far (e.g. MusicBrainz, IMDB, ...).

Also, we plan to investigate the possible heuristics that can be used by our import-by-query and their impact on the number of imported facts per input user's query. In particular, the heuristic

that is used to rank the queries produced as output by the sub-query rewriting step to submit it to Linked Data. The heuristic we used in our experiments is based on the number of triples in an external query. Many possible factors can be used for other heuristics (e.g. statistics about the external datasets, the available number of facts about properties appear in the external query, ...).

Furthermore, we plan to benefit from the weighted contradictory facts discovered by our probabilistic algorithms to adapt the probabilistic weights attached to the uncertain facts and rules. More precisely, it may happen the probabilistic algorithms infer two contradictory facts about the same two entities. The first fact describes that these two entities are the same while the other describes that these two entities are different. This means that one or several uncertain facts or uncertain rules that lead to infer these contradictory facts are wrong. In such a case we can suppose that the fact that has the lowest probabilistic weight among the two contradictory facts is the wrong one. Then, we apply a “punishment” method that reduces all the probabilistic weights of the uncertain facts and rules that lead to infer this fact.

Another future work we plan to explore is how to use our Rule-based approach for **Data Fusion**. Once sameAs links are computed between URIs in a particular dataset and the entities identified by these URIs are confirmed to represent the same real-world object, the question is how to fusion the information attached to these entities and attach it to one representative entity in a consistent, accurate, and useful manner. In particular, any data fusion technique available in such a setting must be able to deal with conflicts that may appear in the descriptions of these entities, caused by outdated information, inaccurate information or errors. The rule-based approach can be used to solve the Data Fusion problem. An example of rules that can be used to solve this problem is: (**IF** (1) two person entities per_{new}, per_{old} are identified to be the same using owl:sameAs, (2) per_{new} has a particular occupation $occ1$ and per_{old} has another different occupation $occ2$ and (3) the fact that describes the occupation of per_{new} is created after the fact that describes the occupation of per_{old} **THEN** the entity e_{fusion} that merges $e1$ and $e1$ must have the occupation $occ1$).

6.2 Trust in Semantic P2P Networks

We have proposed an approach for trust that is adapted to a Semantic P2P Network setting. It exploits knowledge provided by alignments (possibly unsound and/or incomplete) together with user feedback in order to compute and refine trust over time. In our approach trust that a peer P1 has towards another peer P2 is subjective and depends on a specific query. It is modeled as a random variable representing P1’s belief that answers returned by P2 are satisfactory.

We have proposed a trust mechanism that uses alignments between peers' ontologies to construct prior distributions of these beliefs, from which initial values of trust are calculated. Then, it performs Bayesian inference to build posterior distributions of peers' beliefs from their prior distributions and user feedback on a sample of answers.

For the evaluation of our mechanism of trust, we have designed a scenario representative of the novel topic of social networks and the semantic web. Based on this scenario, we have built a semantic P2P bookmarking system (TrustMe) in which we can vary different quantitative and qualitative parameters so as to measure their influence on (i) the convergence of trust, and (ii) the gain in the quality of peer answers — measured with precision and recall — when query-answering is guided by trust.

Experimental results showed that trust values converge as more queries are sent and answers received. Furthermore, the use of trust improves quality of answers in both precision and recall.

We plan to extend our model of trust in different ways. In the current version of the model, peers' queries are not propagated. Even so, as our experimental results confirmed (see Section 4.4.2), trust ensures reasonable high recall values. Indeed, among the instances that a peer P_i receives from one of its trustworthy acquaintances P_j , and that P_i adds to its populated ontology, there might be instances that P_j received from one trustworthy acquaintance P_k , also trustworthy for P_i but not acquainted with it, and that P_j added to its populated ontology before. This, however, would not happen if P_i considered P_j to be untrustworthy, as P_i would not accept P_j 's instances. Thus, query propagation seems to be necessary to ensure total recall. In the case of answers coming from propagated queries, we plan to use composition of alignments to build prior beliefs of trust [25].

The ontologies considered in our approach are taxonomies of classes with disjointness relationships, and the query language studied only allows peers to request instances of classes. We plan to extend our model in order to deal with more expressive ontology and query languages.

In our model of trust, instances peers are not 100% sure of are attached a probability to be satisfactory which is computed on the basis of provenance information. The provenance of an instance a at time t with respect to a peer P_i is the set of peers that, in the light of P_i 's past experience at time t , have a in their populated ontologies. Modeling provenance as a set already allows us to estimate the probability of instances to be satisfactory. Specifically, the probability of the instance a to be satisfactory is estimated with the mean of the trust values of the peers included in its provenance set, i.e. the trust values of the peers that provided a to P_i and that, as far as P_i is aware, have not discarded it. Using the mean to aggregate trust values is a first solution but other possible aggregation functions can be studied (e.g. min or max values). Moreover, provenance can be modeled in a sequential way so that it captures the “history” of

instances (e.g. which peer provided an instance first). This may allow to identify unsatisfactory instances more efficiently.

One of the sources of information often used to compute trust is the so-called witness information, also known as transitivity of trust [46]. It can be summarized as follows: if P_i trusts P_j , and P_j trusts P_k , then P_i trusts P_k . Transitivity of trust cannot be applied directly in our framework since transitivity of logical subsumptions does not hold when these are modeled as conditional probabilities [5]. Nonetheless, there exist alternative inference rules which could be applied in our case. For these, it seems that P_j should act as a “witness” by providing information about the instances she received from P_k . With the metaphor of trust, we look at semantic interoperability as the result of some sort of “social” computing. Our claim is that ontology matching is a starting point towards semantic interoperability that can be complemented with dynamic techniques which exploit collaboration between users/agents/peers.

Appendix A

The set of rules used in the experiments

In this appendix we present certain rules that we used during our experiments.

The first set of rules depicted in Table A.1 are used to distinguish homonymous person entities inside the INA dataset. A name of a person entity in INA dataset is expressed as a value of the property `skos:altLabel` or the property `skos:prefLabel`. Thus, the rules $r1, r2, r3$ are used to identify the homonymous persons by linking them together with the property `ina:sameName`. The slight differences between the two names (including difference caused by special characters or letter cases) are tolerated using the built-in function *Similar*. The key `<name, birthYear, deathYear>` is expressed using the rule $r4$ while the rules $r5$ and $r6$ express that both `birthYear` and `deathYear` are functional properties. Our goal is to disambiguate only the person entities that share the same name. Thus, the entities that can be disambiguated through the rules $r5, r6$ are required to be homonymous.

Table A.2 are used to link using the property `ina:sameNameDBp` the homonymous persons in the INA dataset to the corresponding persons in DBpedia that have similar names. The following alignments between the INA properties and DBpedia properties are expressed in these rules:

	IF	THEN
r1	$\langle ?x1, \text{skos:altLabel}, ?name1 \rangle, \langle ?x2, \text{skos:altLabel}, ?name2 \rangle,$ <code>Similar(?name1, ?name2, 0.99)</code>	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle$
r2	$\langle ?x1, \text{skos:prefLabel}, ?name1 \rangle, \langle ?x2, \text{skos:prefLabel}, ?name2 \rangle,$ <code>Similar(?name1, ?name2, 0.99)</code>	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle$
r3	$\langle ?x1, \text{skos:altLabel}, ?name1 \rangle, \langle ?x2, \text{skos:prefLabel}, ?name2 \rangle,$ <code>Similar(?name1, ?name2, 0.99)</code>	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle$
r4	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle,$ $\langle ?x1, \text{ina:birthYear}, ?Y1 \rangle, \langle ?x2, \text{ina:birthYear}, ?Y1 \rangle$ $\langle ?x1, \text{ina:deathYear}, ?Y2 \rangle, \langle ?x2, \text{ina:deathYear}, ?Y2 \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r5	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle,$ $\langle ?x1, \text{ina:birthYear}, ?Y1 \rangle, \langle ?x2, \text{ina:birthYear}, ?Y2 \rangle$ <code>notEqual(Y1, Y2)</code>	$\langle ?x1, \text{ina:differentFrom}, ?x2 \rangle$
r6	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle,$ $\langle ?x1, \text{ina:deathYear}, ?Y1 \rangle, \langle ?x2, \text{ina:deathYear}, ?Y2 \rangle$ <code>notEqual(Y1, Y2)</code>	$\langle ?x2, \text{ina:differentFrom}, ?x2 \rangle$

TABLE A.1: The set of rules that disambiguate homonymous person inside the INA dataset

	IF	THEN
r7	$\langle ?x1, \text{foaf:name}, ?name1 \rangle, \langle ?x2, \text{skos:altLabel}, ?name2 \rangle,$ $\text{Similar}(?name1, ?name2, 0.99)$	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle$
r8	$\langle ?x1, \text{foaf:name}, ?name1 \rangle, \langle ?x2, \text{skos:prefLabel}, ?name2 \rangle,$ $\text{Similar}(?name1, ?name2, 0.99)$	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle$
r9	$\langle ?x1, \text{rdfs:label}, ?name1 \rangle, \langle ?x2, \text{skos:prefLabel}, ?name2 \rangle,$ $\text{Similar}(?name1, ?name2, 0.99)$	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle$
r10	$\langle ?x1, \text{rdfs:label}, ?name1 \rangle, \langle ?x2, \text{skos:altLabel}, ?name2 \rangle,$ $\text{Similar}(?name1, ?name2, 0.99)$	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle$
r11	$\langle ?x1, \text{prop-fr:nom}, ?name1 \rangle, \langle ?x2, \text{skos:prefLabel}, ?name2 \rangle,$ $\text{Similar}(?name1, ?name2, 0.99)$	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle$
r12	$\langle ?x1, \text{prop-fr:nom}, ?name1 \rangle, \langle ?x2, \text{skos:altLabel}, ?name2 \rangle,$ $\text{Similar}(?name1, ?name2, 0.99)$	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle$

TABLE A.2: The set of rules that disambiguate homonymous person inside the INA dataset

	IF	THEN
r13	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?x1, \text{dbpedia:birthYear}, ?Y1 \rangle, \langle ?x2, \text{ina:birthYear}, ?Y1 \rangle$ $\langle ?x1, \text{dbpedia:deathYear}, ?Y2 \rangle, \langle ?x2, \text{ina:deathYear}, ?Y2 \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r14	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?x1, \text{dbpedia:birthYear}, ?Y1 \rangle, \langle ?x2, \text{ina:birthYear}, ?Y2 \rangle$ $\text{notEqual}(Y1, Y2)$	$\langle ?x1, \text{ina:differentFrom}, ?x2 \rangle$
r15	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?x1, \text{dbpedia:deathYear}, ?Y1 \rangle, \langle ?x2, \text{ina:deathYear}, ?Y2 \rangle$ $\text{notEqual}(Y1, Y2)$	$\langle ?x1, \text{ina:differentFrom}, ?x2 \rangle$

TABLE A.3: The set of rules that disambiguate homonymous person inside the INA dataset

$\langle \text{skos:altLabel}, \text{foaf:name}, = \rangle, \langle \text{skos:prefLabel}, \text{foaf:name}, = \rangle, \langle \text{skos:altLabel}, \text{rdfs:Label}, = \rangle,$
 $\langle \text{skos:prefLabel}, \text{rdfs:Label}, = \rangle, \langle \text{skos:altLabel}, \text{prop-fr:nom}, = \rangle, \langle \text{skos:prefLabel}, \text{prop-fr:nom}, = \rangle.$ Similar to the rules in Table A.1 the slight differences between names are tolerated using the built-in function *Similar*.

Rules that conclude the owl:sameAs and owl:differentFrom links between the entities of type ina:PhysicalPerson from the INA dataset and entities from DBpedia dataset are depicted in figure Table A.3. These rules are obtained by propagating the alignments $\langle \text{ina:birthYear}, \text{dbpedia:birthYear}, = \rangle, \langle \text{ina:deathYear}, \text{dbpedia:deathYear}, = \rangle$ in the rules *r4, r5, r6*.

Inferring the owl:sameAs and owl:differentFrom links between the entities of type ina:VideoPerson from the INA dataset and entities from DBpedia dataset is done by applying the rules depicted in Table A.4. These rules are provided by the INA experts. They express that two homonymous persons are the same if they played an important role (presenter, director ...) in the same Tv program. In DBpedia, persons are linked to program in which they played an important role using the property `dbpedia:wikiPageWikiLink`. Their particular role are rarely specified. For example, in DBpedia.fr the animator Jacques Martin appears only as the presenter of the program "le petit rapporteur" although he presented many other programs (e.g. "Dimanche Martin", "Ainsi font, font, font", "Midi-Magazine", ...). For this reason, these rules do not specify the role of the dbpedia entities in the programs.

As we explained before in Section 2.4.4, renaming technique is done for properties that are both EDB (appears in the set of ground facts) and IDB (appears in the conclusion of the ground rules)

	IF	THEN
r16	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourPresentateur}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r17	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourRealisateur}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r18	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourProducteur}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r19	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourChefOrchestre}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r20	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourCommentateur}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r21	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourAssistantRealisation}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r22	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourAuteurOeuvreOriginalTele}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r23	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourPresentateur}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r24	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourRealisateur}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r25	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourProducteur}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r26	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourChefOrchestre}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r27	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourCommentateur}, ?x2 \rangle, \langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r28	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourAssistantRealisation}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r29	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourAuteurOeuvreOriginalTele}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle, \langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$

TABLE A.4: Rules that disambiguate homonyms person based on their roles in a video

	IF	THEN
r30	$\langle ?x1, \text{owl:sameAs}, ?x2 \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$
r31	$\langle ?x1, \text{owl:differentFrom}, ?x2 \rangle$	$\langle ?x1, \text{ina:differentFrom}, ?x2 \rangle$

TABLE A.5: Rules that used in the renaming techniques for the ground sameAs and different-From facts

	IF	THEN
r32	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	$\langle ?x2, \text{ina:sameAs}, ?x1 \rangle$
r33	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle, \langle ?x2, \text{ina:sameAs}, ?x3 \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x3 \rangle$
r34	$\langle ?x1, \text{ina:differentFrom}, ?x2 \rangle$	$\langle ?x2, \text{ina:differentFrom}, ?x1 \rangle$
r35	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle, \langle ?x2, \text{ina:differentFrom}, ?x3 \rangle$	$\langle ?x1, \text{ina:differentFrom}, ?x3 \rangle$

TABLE A.6: Rules that express the semantics of owl:sameAs and owl:differentFrom

in the same time, which is the case of owl:sameAs owl:differentFrom. These properties are re-named in the ground facts to the EDB properties ina:sameAs and ina:differentFrom accordingly. Then, the rule *r30* is used to express the ina:sameAs and owl:sameAs are equivalent and the rule *r31* is used to express the ina:differentFrom and owl:differentFrom are equivalent.

Rules that express the semantics of owl:sameAs and owl:differentFrom are depicted in Table A.6.

Appendix B

The set of uncertain rules used in the experiments

In this section we list: the set of uncertain rules that are used during our experiments in Chapter 2, their ranking over a scale from 1 to 3 assigned to them by domain experts and their probabilistic weights.

Rules $r1$, $r2$ in Table B.1 are used to link two INA homonymous person with a sameAs link if they share the same birthYear or the same deathYear, while rules $r3$, $r4$ are used to link a person from the INA dataset with another person from DBpedia and they are obtained by propagating the two alignments $\langle \text{ina:birthYear}, \text{dbpedia:birthYear}, = \rangle$, $\langle \text{ina:deathYear}, \text{dbpedia:deathYear}, = \rangle$ inside $r1$, $r2$.

	IF	THEN	Rank	Prob. Weight
r1	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle,$ $\langle ?x1, \text{ina:birthYear}, ?year \rangle,$ $\langle ?x2, \text{ina:birthYear}, ?year \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	1	0.3
r2	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle,$ $\langle ?x1, \text{ina:deathYear}, ?year \rangle,$ $\langle ?x2, \text{ina:deathYear}, ?year \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	1	0.3
r3	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?x1, \text{dbpedia:birthYear}, ?year \rangle,$ $\langle ?x2, \text{ina:birthYear}, ?year \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	1	0.3
r4	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?x1, \text{dbpedia:deathYear}, ?year \rangle,$ $\langle ?x2, \text{ina:deathYear}, ?year \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	1	0.3

TABLE B.1: Rules that disambiguate homonyms person based on birthYear and deathYear

The property `ina:hasOccupation` is used in the INA dataset to describe the professions (or the situations in general) of a person entity. Thus, rule $r5$ links two INA homonymous person with a sameAs link if they have two professions with similar labels up to a threshold 0.9. The rule $r6$ propagates the alignment $\langle \text{ina:hasOccupation}, \text{dbpedia:hasOccupation}, = \rangle$ inside the rule $r5$.

	IF	THEN	Rank	Prob. Weight
r5	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle,$ $\langle ?x1, \text{ina:hasOccupation}, ?occ1 \rangle,$ $\langle ?x2, \text{ina:hasOccupation}, ?occ2 \rangle$ $\langle ?occ1, \text{rdfs:label}, ?label1 \rangle, \langle ?occ2, \text{rdfs:label}, ?label2 \rangle$ $\text{Similar}(?label1, ?label2, 0.9)$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r6	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?x1, \text{dbpedia:occupation}, ?occ1 \rangle,$ $\langle ?x2, \text{ina:hasOccupation}, ?occ2 \rangle$ $\langle ?occ1, \text{rdfs:label}, ?label1 \rangle, \langle ?occ2, \text{rdfs:label}, ?label2 \rangle$ $\text{Similar}(?label1, ?label2, 0.9)$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6

TABLE B.2: Rules that disambiguate homonyms person based on their occupations

Rules in Table B.3 are used to link two homonymous person: one from the INA dataset and the other person from DBpedia if the INA's person have a particular role in a video and this video has a title similar to one of the objects linked to the DBpedia's person. Roles that are used in these rules are less important than the ones used before in the certain rules in Table A.4)

Rules *r17* is used to link two homonymous persons if they are linked to the same video entity whatever their roles are inside this video.

	IF	THEN	Rank	Prob. Weight
r7	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourJRI}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r8	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourInterprete}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r9	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourParticipant}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r10	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourMetteurEnSceneTheatre}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r11	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourMembreOrchetre}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreCollection}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r12	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourJRI}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r13	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourInterprete}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r14	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourParticipant}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r15	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourMetteurEnSceneTheatre}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6
r16	$\langle ?x1, \text{ina:sameNameDBp}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourMembreOrchetre}, ?x2 \rangle,$ $\langle ?v, \text{ina:aPourTitreHorraire}, ?titre \rangle$ $\langle ?x1, \text{dbpedia:wikiPageWikiLink}, ?link \rangle,$ $\langle ?link, \text{rdfs:label}, ?titre \rangle$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	2	0.6

TABLE B.3: Rules that disambiguate homonyms person based on their roles in a video

	IF	THEN	Rank	Prob. Weight
r17	$\langle ?x1, \text{ina:sameName}, ?x2 \rangle,$ $\langle ?v, \text{rdf:type}, \text{ina:video} \rangle,$ $\langle ?v, ?p1, ?x1 \rangle, \langle ?v, ?p2, ?x2 \rangle,$	$\langle ?x1, \text{ina:sameAs}, ?x2 \rangle$	3	0.9

TABLE B.4: Rules that disambiguate homonyms person if they are related to the same video

Bibliography

- [1] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS-33 2000), 4-7 January, 2000, Maui, HI, USA. IEEE Computer Society, 2000.
- [2] Serge Abiteboul, Zoë Abrams, Stefan Haar, and Tova Milo. Diagnosis of asynchronous discrete event systems: datalog to the rescue! In PODS'05, pages 358–367, 2005.
- [3] Serge Abiteboul, Richard Hull, and Victor Vianu. Foundations of Databases. Addison-Wesley, 1995.
- [4] Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. Crowdsourcing linked data quality assessment. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, The Semantic Web – ISWC 2013, volume 8219 of Lecture Notes in Computer Science, pages 260–276. Springer Berlin Heidelberg, 2013.
- [5] Ernest W. Adams. A primer of probability logic. CSLI, 1998.
- [6] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie Christine Rousset, and Laurent Simon. Distributed reasoning in a peer-to-peer setting: application to the semantic web. Journal of Artificial Intelligence Research, 25:269–314, 2006.
- [7] Mustafa Al-Bakri, Manuel Atencia, and Marie-Christine Rousset. TrustMe, I got what you mean! A trust-based semantic P2P bookmarking system. In Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings, volume 7603 of LNCS, pages 442–445. Springer, 2012.
- [8] Sivan Albagli, Rachel Ben-Eliyahu-Zohary, and Solomon Eyal Shimony. Markov network based ontology matching. Journal of Computer and System Sciences, 78(1):105–118, 2012.
- [9] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. Journal of Web Semantics, 5(2):58–71, 2007.

- [10] Manuel Atencia, Mustafa Al-Bakri, and Marie-Christine Rousset. Trust in networks of ontologies and alignments. Journal of Knowledge and Information Systems, pages 1–27, 2013.
- [11] Manuel Atencia, Jérôme David, and François Scharffe. Keys and pseudo-keys detection for web datasets cleansing and interlinking. In EKAW’12, pages 144–153, 2012.
- [12] Manuel Atencia, Jérôme David, and François Scharffe. Keys and pseudo-keys detection for web datasets cleansing and interlinking. In Annette ten Teije, Johanna Völker, Siegfried Handschuh, Heiner Stuckenschmidt, Mathieu d’Aquin, Andriy Nikolov, Nathalie Aussenac-Gilles, and Nathalie Hernandez, editors, Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings, volume 7603 of Lecture Notes in Computer Science, pages 144–153. Springer, 2012.
- [13] Manuel Atencia, Jérôme Euzenat, Giuseppe Pirrò, and Marie-Christine Rousset. Alignment-based trust for resource finding in semantic P2P networks. In The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I, volume 7031 of LNCS, pages 51–66. Springer, 2011.
- [14] Tim Berners-Lee. Linked data. w3c design issues. Date: 2006-07-27.
- [15] Tim Berners-Lee, Wendy Hall, James A. Hendler, Kieron O’Hara, Nigel Shadbolt, and Daniel J. Weitzner. A framework for web science. Foundations and Trends in Web Science, 1(1):1–130, 2006.
- [16] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. Scientific American, 284(5):34–43, May 2001.
- [17] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. Scientific American, 284(5):34–43, 2001.
- [18] P. Billingsley. Probability and measure. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley, 1979.
- [19] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. Web Semantics: science, services and agents on the world wide web, 7(3):154–165, 2009.
- [20] Christoph Böhm, Felix Naumann, Ziawasch Abedjan, Dandy Fenz, Toni Grütze, Daniel Hefenbrock, Matthias Pohl, and David Sonnabend. Profiling linked open data with prolog. In Workshops Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA, pages 175–178. IEEE, 2010.

- [21] William M. Bolstad. Introduction to Bayesian statistics. Wiley, 2007.
- [22] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. Computer Networks, 30(1-7):107–117, 1998.
- [23] Andrea Cali, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. JWS, 14:57–83, 2012.
- [24] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. JAR, 39(3):385–429, 2007.
- [25] Jérôme Euzenat. Algebras of ontology alignment relations. In The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings, volume 5318 of LNCS, pages 387–402. Springer, 2008.
- [26] Jérôme Euzenat, Christian Meilicke, Heiner Stuckenschmidt, Pavel Shvaiko, and Cássia Trojahn dos Santos. Ontology Alignment Evaluation Initiative: six years of experience. Journal on Data Semantics XV, volumen 6720 of LNCS, pages 158–192, 2011.
- [27] Jérôme Euzenat and Pavel Shvaiko. Ontology matching. Springer, 2007.
- [28] Alfio Ferrara, Andriy Nikolov, and François Scharffe. Data linking for the semantic web. IJSWIS, 7(3):46–76, 2011.
- [29] Charles L. Forgy. Expert Systems, chapter Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, pages 324–341. IEEE Computer Society Press, 1990.
- [30] Norbert Fuhr. Probabilistic Datalog: Implementing logical information retrieval for advanced applications. Journal of the American Society for Information Science, 51(2):95–110, 2000.
- [31] Norbert Fuhr. Probabilistic datalog: Implementing logical information retrieval for advanced applications. Journal of the American Society for Information Science, 51(2):95–110, 2000.
- [32] Jennifer Golbeck. Combining provenance with trust in social networks for semantic web content filtering. In Provenance and Annotation of Data, International Provenance and Annotation Workshop, IPAW 2006, Chicago, IL, USA, May 3-5, 2006, Revised Selected Papers, volume 4145 of LNCS, pages 101–108. Springer, 2006.
- [33] Wael H. Gomaa and Aly A. Fahmy. Article: A survey of text similarity approaches. International Journal of Computer Applications, 68(13):13–18, April 2013. Full text available.

- [34] Bernardo Cuenca Grau and Boris Motik. Reasoning over ontologies with hidden content: The import-by-query approach. *CoRR*, abs/1401.5853, 2014.
- [35] Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In Leonid Libkin, editor, *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40. ACM, 2007.
- [36] Ali Hasnain, Mustafa Al-Bakri, Luca Costabello, Zijie Cong, Ian Davis, and Tom Heath. Spamming in Linked Data. In *Third International Workshop on Consuming Linked Data (COLD2012)*, Boston, États-Unis, November 2012.
- [37] Martin Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *Knowledge Engineering: Practice and Patterns*, pages 329–346. Springer, 2008.
- [38] Martin Hepp. The web of data for e-commerce in brief. In *Web Engineering*, pages 510–511. Springer, 2012.
- [39] Knut Hinkelmann and Helge Hintze. Computing cost estimates for proof strategies. In *ELP’93*, volume 798, pages 152–170, 1993.
- [40] Aidan Hogan, Antoine Zimmermann, Jürgen Umbrich, Axel Polleres, and Stefan Decker. Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *JWS*, 10:76–110, 2012.
- [41] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.
- [42] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.
- [43] Georgi Kobilarov, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee. Media meets semantic web – how the bbc uses dbpedia and linked data to make connections. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, editors, *The Semantic Web: Research and Applications*, volume 5554 of *Lecture Notes in Computer Science*, pages 723–737. Springer Berlin Heidelberg, 2009.
- [44] Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd International Conference on World Wide Web, WWW ’14*, pages

- 747–758, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.
- [45] Hsin-Tsang Lee, Derek Leonard, Xiaoming Wang, and Dmitri Loguinov. Irlbot: Scaling to 6 billion pages and beyond. *ACM Trans. Web*, 3(3):8:1–8:34, July 2009.
- [46] Guanfeng Liu, Yan Wang, and Mehmet A. Orgun. Trust transitivity in complex social networks. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.
- [47] Robert McCann, Warren Shen, and AnHai Doan. Matching schemas in online communities: A Web 2.0 approach. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México, pages 110–119*. IEEE, 2008.
- [48] Peter Mika. *Social networks and the semantic web*. PhD thesis, SIKS, the Dutch Graduate School for Information and Knowledge Systems, 2006.
- [49] Prasenjit Mitra, Natasha F. Noy, and Anuj R. Jaiswal. OMEN: A probabilistic ontology mapping tool. In *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, volume 3729 of *LNCS*, pages 537–547. Springer, 2005.
- [50] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A computational model of trust and reputation. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35 2002)*, 7-10 January, 2002, Big Island, HI, USA. IEEE Computer Society, 2002.
- [51] M. E. J. Newman. Models of the small world. *Journal of Statistical Physics*, 101(3-4):819–941, 2000.
- [52] Raymond Ng and V.S. Subrahmanian. Stable model semantics for probabilistic deductive databases. In Z.W. Ras and M. Zemankova, editors, *Methodologies for Intelligent Systems*, volume 542 of *Lecture Notes in Computer Science*, pages 162–171. Springer Berlin Heidelberg, 1991.
- [53] Raymond Ng and V.S. Subrahmanian. A semantical framework for supporting subjective and conditional probabilities in deductive databases. *Journal of Automated Reasoning*, 10(2):191–235, 1993.
- [54] Axel-Cyrille Ngonga Ngomo and Sören Auer. Limes a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three, IJCAI'11*, pages 2312–2317, 2011.

- [55] Mathias Niepert, Christian Meilicke, and Heiner Stuckenschmidt. A probabilistic-logical framework for ontology matching. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010. AAAI Press, 2010.
- [56] Kieron O’Hara, Harith Alani, Yannis Kalfoglou, and Nigel Shadbolt. Trust strategies for the semantic web. In Proceedings of the ISWC’04 Workshop on Trust, Security, and Reputation on the Semantic Web, Hiroshima, Japan, November 7, 2004, volume 127 of CEUR Workshop Proceedings. CEUR-WS.org, 2004.
- [57] Thiago Pachêco Andrade Pereira, Carlos Eduardo S. Pires, and Ana Carolina Salgado. Exploring web semantic knowledge and user feedback to improve ontology matching. In 2011 Database and Expert Systems Applications, DEXA, International Workshops, Toulouse, France, August 29 - Sept. 2, 2011, pages 234–238. IEEE Computer Society, 2011.
- [58] Nathalie Pernelle, Fatiha Saïs, and Danai Symeonidou. An automatic key discovery approach for data linking. Web Semantics: Science, Services and Agents on the World Wide Web, 23(0):16 – 30, 2013. Data Linking.
- [59] Ilung Pranata, Geoff Skinner, and Rukshan Athauda. A holistic review on trust and reputation management systems for digital environments. International Journal of Computer and Information Technology, 1(1):44–53, 2012.
- [60] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the semantic web. In The Semantic Web - ISWC 2003, Second International Semantic Web Conference, Sanibel Island, FL, USA, October 20-23, 2003, Proceedings, volume 2870 of LNCS, pages 351–368. Springer, 2003.
- [61] Stewart Robinson. Simulation - The practice of model development and use. Wiley, 2004.
- [62] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. AI Review, 24(1):33–60, 2005.
- [63] Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. Combining a logical and a numerical method for data reconciliation. JoDS, 12:66–94, 2009.
- [64] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: probabilistic alignment of relations, instances, and schema. PVLDB, 5(3):157–168, 2011.
- [65] Rémi Tournaire, Jean-Marc Petit, Marie-Christine Rousset, and Alexandre Termier. Discovery of probabilistic mappings between taxonomies: Principles and experiments. JoDS, 15:66–101, 2011.

- [66] Rémi Tournaire, Jean-Marc Petit, Marie-Christine Rousset, and Alexandre Termier. Discovery of probabilistic mappings between taxonomies: principles and experiments. Journal on Data Semantics XV, volumen 6720 of LNCS, pages 66–101, 2011.
- [67] Jacopo Urbani, Frank van Harmelen, Stefan Schlobach, and Henri E. Bal. Querypie: Backward reasoning for owl horst over very large knowledge bases. In ISWC'11, volume 7031 of LNCS, pages 730–745, 2011.
- [68] Laurent Vieille. Recursive axioms in deductive databases: The query/subquery approach. In Expert Database Conf., pages 253–267, 1986.
- [69] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk - a link discovery framework for the web of data. In Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW, 2009.
- [70] Aidan Hogan Y, Andreas Harth Y Z, Re Passant Y, Stefan Decker Y, and Axel Polleres Y. Weaving the pedantic web.
- [71] Amrapali Zaveri, Dimitris Kontokostas, Mohamed A. Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In Proceedings of the 9th International Conference on Semantic Systems, I-SEMANTICS '13, pages 97–104, New York, NY, USA, 2013. ACM.