Fusing Automatically Extracted Annotations for the Semantic Web

Thesis

FUSING AUTOMATICALLY EXTRACTED ANNOTATIONS FOR THE SEMANTIC WEB

Andriy Nikolov, MSc

Knowledge Media Institute

The Open University

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy in Computer Science

30 September 2009

Abstract

This research focuses on the problem of semantic data fusion. Although various solutions have been developed in in the research communities focusing on databases and formal logic, the choice of an appropriate algorithm is non-trivial because the performance of each algorithm and its optimal configuration parameters depend on the type of data, to which the algorithm is applied. In order to be reusable, the fusion system must be able to select appropriate techniques and use them in combination. Moreover, because of the varying reliability of data sources and algorithms performing fusion subtasks, uncertainty is an inherent feature of semantically annotated data and has to be taken into account by the fusion system. Finally, the issue of schema heterogeneity can have a negative impact on the fusion performance.

To address these issues, we propose KnoFuss: an architecture for Semantic Web data integration based on the principles of problem-solving methods. Algorithms dealing with different fusion subtasks are represented as components of a modular architecture, and their capabilities are described formally. This allows the architecture to select appropriate methods and configure them depending on the processed data. In order to handle uncertainty, we propose a novel algorithm based on the Dempster-Shafer belief propagation. KnoFuss employs this algorithm to reason about uncertain data and method results in order to refine the fused knowledge base. Tests show that

these solutions lead to improved fusion performance.

Finally, we addressed the problem of data fusion in the presence of schema heterogeneity. We extended the KnoFuss framework to exploit results of automatic schema alignment tools and proposed our own schema matching algorithm aimed at facilitating data fusion in the Linked Data environment. We conducted experiments with this approach and obtained a substantial improvement in performance in comparison with public data repositories.

Acknowledgements

This research could not be possible without many people: here I can only briefly acknowledge them.

First, I owe thanks to my teachers and mentors from the Kharkov National University of Radioelectronics and, in particular, to Prof. Vagan Terziyan for introducing me into the world of AI and the Semantic Web and opening the path that would eventually lead me to KMi.

During the four years of my PhD study, I have been privileged to work with the best team of supervisors I could wish for: Dr. Victoria Uren, Prof. Enrico Motta, and Prof. Anne de Roeck. I would like to thank Victoria for her constant support, encouragement, technical guidance, and her admirable patience with reading all my drafts. For me it has been a great pleasure working with her. I am grateful to Enrico for his invaluable constructive feedback and new insights he always provided. His research was an inspiration for a large part of the work described in this thesis. I would like to thank Anne for her comments and our fruitful discussions.

I would like to thank my colleagues and friends from KMi who not only provided me with all possible help, but also were a constant source of joy and helped to make these years the happiest in my adult life so far.

I am grateful to all the people I collaborated with in the X-Media project for

always valuable discussions and all the things I learned from them. This project has provided me with a unique opportunity to conduct research in an area I have always been interested in.

And last, but foremost, I want to thank my parents to whom I owe everything.

This work was partially funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

Publications

The research reported in this dissertation has contributed to the following publications:

- Andriy Nikolov (2006) Fusing automatically extracted semantic annotations. KnowledgeWeb PhD Symposium 2006 (KWEPSY 2006), Budva, Montenegro.

- Andriy Nikolov (2006). Fusing automatically extracted annotations for the Semantic Web. KMi Technical report (kmi-06-12).

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2007). KnoFuss: A comprehensive architecture for knowledge fusion. In Poster Track, 4th International Conference on Knowledge Capture (K-CAP 2007), Whistler, Canada

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2007). Using the Dempster-Shafer theory of evidence to resolve ABox inconsistencies. Workshop: Uncertainty Reasoning for the Semantic Web, 6th International Semantic Web Conference (ISWC 2007), Busan, Korea.

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2008). Using the Dempster-Shafer theory of evidence to resolve ABox inconsistencies. In:

Uncertainty Reasoning for the Semantic Web I, Eds: Paulo C.G. da Costa, Claudia d'Amato, Nicola Fanizzi, Kathryn B. Laskey, Ken Laskey, Thomas Lukasiewicz, Matthias Nickles, Mike Pool.

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2008). Handling instance coreferencing in the KnoFuss architecture. Workshop: Identity and Reference on the Semantic Web, 5th European Semantic Web Conference (ESWC 2008), Tenerife, Spain.

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2008). Integration of semantically annotated data by the KnoFuss architecture. 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008), Acitrezza, Italy.

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2008). Refining instance coreferencing results using belief propagation. 3rd Asian Semantic Web Conference (ASWC 2008), Bangkok, Thailand.

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2009). Towards data fusion in a multi-ontology environment. Workshop: Linked Data on the Web (LDOW 2009), 18th International World Wide Web Conference (WWW 2009), Madrid, Spain.

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2009). Overcoming schema heterogeneity for data-level integration. In Poster Track, 6th European Semantic Web Conference (ESWC 2009), Heraklion, Greece.

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2009). Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. 4th Asian Semantic Web Conference (ASWC 2009), Shanghai, China.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter provides a brief introduction to the thesis. It includes the motivation for the work, a brief overview of the approach and the main research questions the thesis focuses on. It also contains the definitions of the important terms we use and outlines the structure of the dissertation.

1.1 Motivation

The amount of information accessible online is constantly growing. According to [46], the total amount of digitised information increases by a factor of 10 every five years, and in 2007 it amounted to 281 exabytes (281 billion GB). In parallel, the heterogeneity of information also rises: relevant information about the same topic is distributed over increasing number of sources. These factors present new challenges for applications which need to handle this data, as more and more automation is needed for data management.

Even if we consider the level of a single enterprise and its needs, pieces of information can be stored in internal databases, corporate reports, e-mail threads, spreadsheets, etc., not counting relevant information from the public domain accessible via the Web. The ability to locate and access all information relevant to his/her work is essential for an employee to perform tasks efficiently and make correct decisions.

This is even more challenging on a Web scale, given the number of sources that must be taken into account, the dynamic aspects of information, which make it become obsolete with time, and the contradictory viewpoints expressed by different sources. While popular Web search engines, such as Google, can perform well on standard search tasks, there are many tasks which require viewing information to be combined from multiple sources in order to address a user need. For instance, an individual needs to keep track of all information available about him/her on the Web to manage the risks of identity theft [105].

Semantic Web technologies have been proposed as a way to make this growing

amount of information more manageable by describing it in a standardised machine-processable format and by using shared identifiers (URIs) to encode entities [7]. However, although the standard data format makes integration easier to perform automatically, this process is still not trivial as several issues need to be solved. In particular, it is necessary to identify coreferent individuals describing the same real-world entity but having different URIs, to process potential inconsistencies, and to handle provenance and reliability of information from each source.

Data integration has long been recognised as an important research problem in different communities such as databases [40, 36, 126], logic-based AI [65, 68], and XML data [58, 123]. Commonly recognised challenges, which have to be dealt with, include schema-level and data-level heterogeneity. While in the Semantic Web community a considerable amount of work has focused on schema-level heterogeneity [38], data-level issues have received less attention. However, with the recent emergence of the Linking Open Data initiative[1] and the growing amount of RDF[2] data being published, data integration issues have acquired much more importance.

In all these communities multiple algorithms have been developed to deal with each of the data integration subproblems (e.g., string similarity [21] and machine-learning techniques [8] to identify coreferences; logical reasoning [69] and provenance analysis [82] to handle inconsistencies; mathematical frameworks such as Bayesian probability theory and Dempster-Shafer belief functions [91] to reason about uncertainty; etc.). However, the task of fusing semantic data structured according to

[1]http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
[2]http://www.w3.org/RDF/

RDFS[3] and OWL[4] ontologies has its specific characteristics, which must be considered when adapting existing methods or developing new ones. Unlike in relational datasets, data instances do not always have a strict structure, classes are organised into hierarchies, and ontological axioms allow different kinds of constraints and restrictions to be defined. Also for any of the data-level problems listed above the choice of an optimal algorithm is non-trivial: there is no single algorithm, which is the best for all possible domains and use case scenarios. Given that even within a single domain there may be a variety of types of data, the fusion process requires using a flexible workflow to select appropriate algorithms, configure their parameters, use them in combination, and aggregate their results.

This research was performed in the context of the X-Media project[5]. The project focuses on the topic of corporate knowledge management using large-scale information extraction and Semantic Web technologies to organise information contained in various documents. In this scenario, a corporate ontology is populated both automatically and manually using information from different sources and documents (see Fig. 1.1). Thus, while there is no schema-level heterogeneity, the data-level issues are crucial. These issues include coreference resolution (recognizing the cases when different sources refer to the same real-world entities and unifying the URIs of such instances) and inconsistency resolution (processing the cases when the sources contain conflicting information). The data, which have to be integrated, are often noisy: automatic extraction algorithms do not have 100% quality, human editors make occasional mistakes, data become obsolete with time and, finally, the sources may have

[3]http://www.w3.org/TR/rdf-schema/

[4]http://www.w3.org/TR/owl-features/

[5]http://www.x-media-project.org/

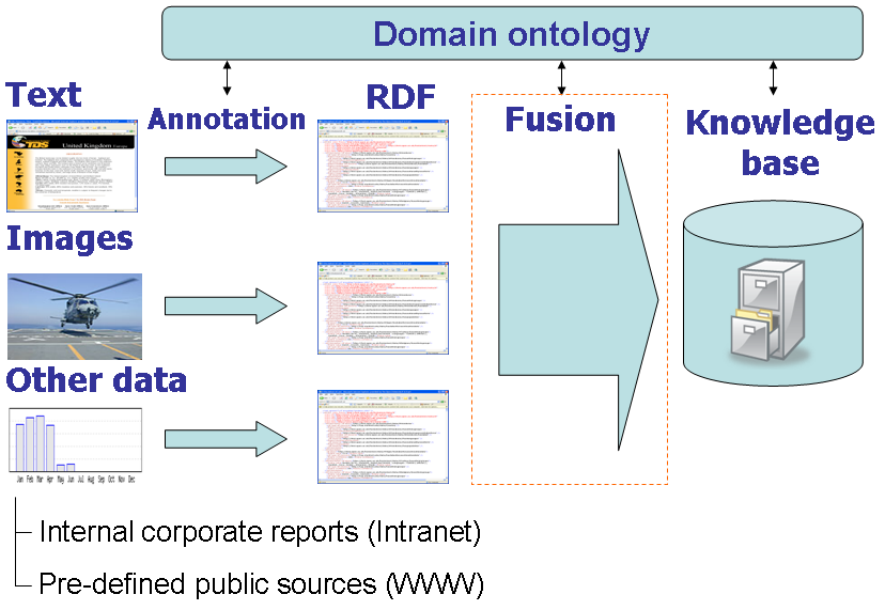

Figure 1.1: Fusion in the corporate knowledge management scenario as assumed in the X-Media project

genuinely contradictory views. The output of the coreferencing methods is also not fully reliable. Thus, performing fusion of semantic data in the presence of noise in a corporate knowledge management scenario was our initial motivation.

As our work continued, we also considered more generic fusion scenarios, in particular, fusion of Linked Data repositories published on the Web. Linked Data is defined as "a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF"[6]. This best practice was widely adopted, and multiple RDF repositories have been published on the Web (Fig. 1.2). Coreference links between entities (RDF individuals) in different datasets constitute a major added value: these links allow combining data about the same individuals stored in different locations.

[6]http://en.wikipedia.org/wiki/Linked_Data

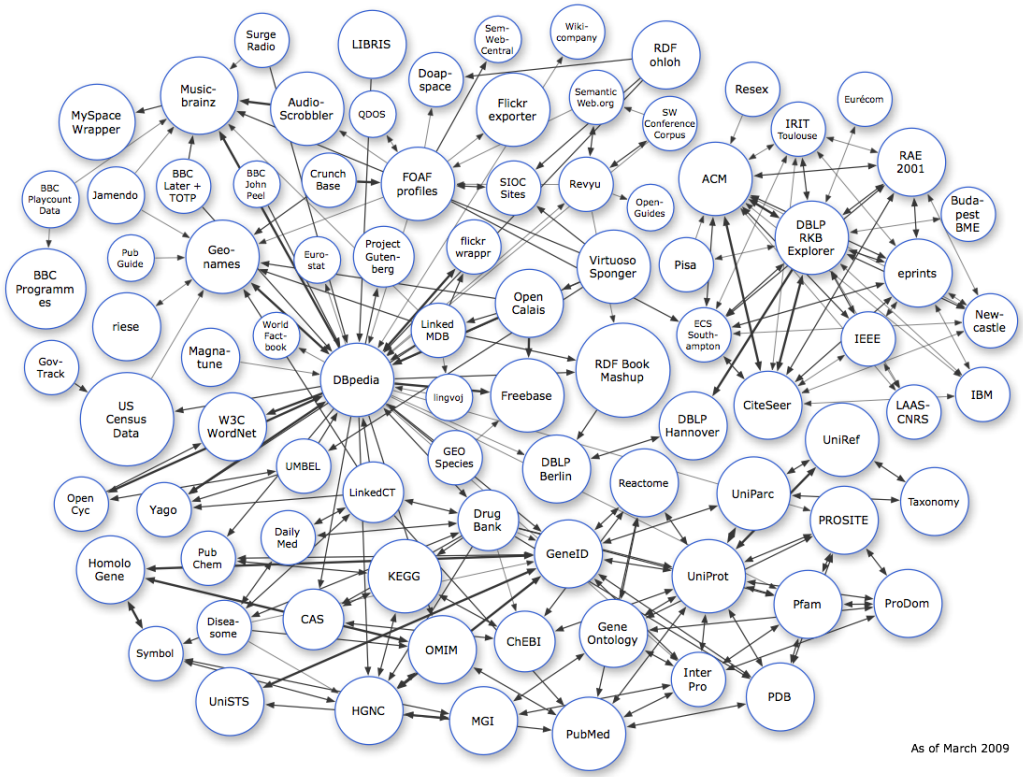

Figure 1.2: The Linking Open Data cloud diagram (March 2009).
From *http://www4.wiwiss.fu-berlin.de/bizer/pub/lod-datasets_2009-03-05.html.*

Due to large volumes of data which have to be processed, these links cannot be produced manually, so automatic techniques have to be employed. Addressing the data fusion problems in this scenario is the focus of the second part of the thesis. In both these scenarios the fusion is assumed to be performed in an off-line mode. Therefore, we did not focus on the performance of our techniques in terms of time and memory cost.

1.2 Approach

This dissertation explores the knowledge fusion problem in the context of Semantic Web data and uses the problem-solving methods approach [83, 41] as a base for constructing the fusion architecture. This approach provides a framework for organizing alternative algorithms solving primitive fusion subtasks into a library, defining their configuration parameters, and using them in combination. The capabilities and domain assumptions of the algorithms are formally described using the *fusion ontology*, which enables the automated selection and invocation of algorithms, combining their results, and passing them to the next stage in order to conduct these subtasks in sequence.

An essential component of the architecture is the exploitation of uncertainty management to reason about the reliability of pieces of data and the output of methods dealing with fusion subtasks. Our approach employs Dempster-Shafer theory of evidence in combination with ontological reasoning to aggregate the uncertainty of different factors. This aggregated uncertainty is then used to make decisions about the data being integrated and refine it.

This approach is used to address a number of research questions.

1.3 Research questions

The general research problem addressed in this dissertation is:

> How can we perform effective data-level fusion of semantically annotated data generated from different sources?

This problem includes several more specific questions:

1. How should algorithms performing fusion subtasks be used in combination to implement the semantic data fusion workflow?
2. How can we support the reusability of fusion algorithms across domains?
3. How can we exploit axioms defined in domain ontologies to improve the performance of fusion algorithms?
4. What kind of uncertainty management framework is suitable for fusion?
5. How can we exploit uncertainty and provenance information to improve the fusion performance?
6. What is the impact of ontology schema heterogeneity on the data fusion process?

Note, that since we assume an off-line fusion scenario, the question of optimising the time performance is outside the scope of this thesis. Effectiveness in our case refers only to the quality of the fusion output.

1.4 Definition of terms

Most of the terms will be defined in the course of the thesis as needed. In this section we give only the main terms, which are used throughout the dissertation and are important for understanding the narrative.

The term *fusion* is used in several domains of AI and computer science by many authors and sometimes with different interpretations. In a survey paper [12] fusion is defined as "conjoining or merging information that stems from several sources and exploiting that conjoined or merged information in various tasks such as answering questions, making decisions, numerical estimation, etc." This is an informal and very generic definition. This dissertation considers a more specific knowledge fusion scenario: fusion of semantic data represented in RDF and structured according to standard Semantic Web ontology languages (RDFS and OWL). This dissertation discusses fusion of actual RDF datasets rather than fusion of query results. Each RDF dataset consists of two components: an ontological schema and RDF data instances. Further we refer to the ontological schema (TBox in the description logic terminology) as the *ontology* and to the RDF data instances (ABox) as the *knowledge base*.

1.5 Structure of the dissertation

Chapter 2 reviews the literature on the fusion problem, especially in the domain of database integration and in the Semantic Web community. Common workflows used by existing fusion systems are analysed, the stages of the fusion process are outlined, and the existing approaches to tackling each stage are reviewed.

Chapter 3 presents the proposed knowledge fusion architecture based on the use of problem-solving methods, which supports dynamic integration of algorithms handling different subtasks od the fusion problem. This chapter mainly deals with questions 1, 2, and 3 stated above.

Chapter 4 describes our approach for resolving data inconsistencies using uncertainty management. The approach combines ontological reasoning and formal uncertainty management using the Dempster-Shafer theory of evidence.

Chapter 5 describes how this approach can be used to complete the fusion process by refining the decisions made by the algorithms at earlier stages of the process. Chapters 4 and 5 jointly address questions 4 and 5.

Chapter 6 describes the results of experiments we performed with the KnoFuss framework.

In Chapter 7 we consider the scenario in which we have to deal with both schema and data heterogeneity in the data fusion process. We discuss an extension of the KnoFuss architecture we developed to deal with this problem and we also discuss the impact that schema heterogeneity has on the data fusion stage. In addition, we propose a novel schema-matching algorithm which aims at facilitating the data fusion process in the Linked Data environment. Thus, this chapter mainly addresses question 6.

Finally, Chapter 8 summarises the main contributions of the study and outlines directions for future work.

Chapter 2

Related work

In this chapter we give an overview of the body of related work relevant to the fusion problem. We start with the description of the data integration problem in the database domain. Then, we give an overview of both the approaches developed in the area of formal logic and the more recent ones which deal with Semantic Web data. Finally, we describe existing architectures, which combine different algorithms to tackle fusion problems, and discuss their limitations.

2.1 The knowledge fusion problem: formulation and relevant areas

A survey paper [12] defines fusion at the most general level as "conjoining or merging information that stems from several sources and exploiting that conjoined or merged information in various tasks such as answering questions, making decisions, numerical estimation, etc.". Imperfections associated with the information being fused result in potential problems, which must be dealt with during the fusion process. Defective aspects of information have been classified in six categories [12]:

- *Ambiguity.*

  In general, ambiguity refers to cases where information has no clear meaning and can be interpreted in different ways. In particular, this includes ambiguous identity, where it is not clear what item a piece of information refers to and whether two pieces of information describe the same entity.

- *Uncertainty.*

  The uncertainty problem occurs when there is not sufficient information to determine whether a certain statement is true or false.

- *Imprecision.*

  The content of a statement may be imprecise with regard to its "granularity" when it allows a range of values rather than a precise value (e.g., a numeric value "between 25 and 30" or date "1981", when an exact date is required).

- *Vagueness.*

  This aspect is similar to imprecision but refers to the language vagueness of

predicate definitions and quantifiers. For instance, terms such as “young”, “high”, “cheap” primarily define fuzzy constraints rather than exact values.

- *Inconsistency.*

  An inconsistency occurs in cases where all available information cannot be simultaneously true, violating some of the domain constraints.

All these issues are interrelated, so none of them can really be treated completely in isolation from the others. For instance, ambiguity, where it is unknown whether two descriptions are about the same entity or not, occurs in the case when the descriptions are similar but at the same time contain some contradictions making them inconsistent. If in this case it is impossible to make a decision with a full confidence, then the issue of uncertainty is introduced, and so on. Existing approaches found in the literature do not address all aspects mentioned above with the same degree of attention but rather focus on some particular issues.

Probably the most significant research effort concerning data integration has been performed in the database community. The main problem considered there is identity ambiguity: finding records which describe the same entity in different databases. The inconsistency issue is less emphasised and refers to the specific problem of value inconsistency, where such identical records have different values for the same field. In contrast, the area of formal logic regards inconsistency as the most important fusion problem, given that more expressive axioms can be defined than in database schemas, and more complex inconsistency patterns are possible. The research studies in this area usually abstract from ambiguity issues and assume them to be solved and symbols to be non-ambiguous. In some sense these research directions converge

and are both relevant when we have to deal with Semantic Web data. On the one hand, datatype properties can cause similar data heterogeneity problems as are found in relational databases. On the other hand, popular ontological schema definition languages (like OWL-DL) are based on the principles of description logic and make capture of complex cases of inconsistency possible.

2.2 Data integration in the database domain

The data integration problem has been studied within the database domain for a long time, and the two top-level integration problems identified for databases [66, 94, 81] still remain relevant for Semantic Web data:

1. *Schema conflicts.* These conflicts are caused by different representation structures used by different sources to describe the same type of data.

2. *Data conflicts.* Data conflicts represent discrepancies between records in merged databases.

Two major causes of data conflicts include:

- Different representations of the same data.

- Incorrect data.

Different representations of the same data make it hard to identify overlapping data in different sources and lead to the occurrence of duplicate records referring to the same real-world entity. A special integration stage is usually used to identify such duplicate records and merge them. This task is known as *record linkage*, *coreference resolution* (or *coreferencing*), or *object identification* and involves generating

mappings: correspondences between pairs of records considered equivalent[1]. The occurrence of wrong data (either incorrect or obsolete) also leads to the appearance of conflicting values when data from different sources is merged.

2.2.1 Record linkage

The problem of record linkage was recognised a long time ago and was initially defined in [84]. A seminal paper by Fellegi & Sunter [40] formalised the problem and described a generic model for a solution algorithm. In its classic form, the problem is formulated as follows. There are two files (lists of elements) A and B and the task is to classify pairs in a product space $A \times B$ into two sets: $M = (a, b); a = b, a \in A, b \in B$ (set of matched pairs) and $U = (a, b); a \neq b, a \in A, b \in B$ (set of non-matched pairs). Given that each real-world entity a or b is described using records $\alpha(a)$ and $\beta(b)$, the classification decision is based on a comparison of two records expressed as a vector function $\gamma(\alpha(a), \beta(b)) = \gamma(a, b)$ (distance function). A decision is represented in the form of a linkage rule $d(\gamma) = P(A_1|\gamma), P(A_2|\gamma), P(A_3|\gamma); \sum_{i=1}^{3} P(A_i|\gamma) = 1$, where A_i denote specific decisions: A_1-positive link, A_3-positive non-link and A_2-possible (uncertain) link. Then the authors propose a probabilistic model introducing conditional probabilities $m(\gamma) = P\gamma[\alpha(a), \beta(b)]|(a, b) \in M = \sum_{(a,b)\in M} P\gamma[\alpha(a), \beta(b)] \cdot P[(a, b)|M]$ and $u(\gamma)$ (similar for $(a, b) \in U$). The obtained value $m(\gamma)/u(\gamma)$ is used to make a decision about the equivalence of two entities by comparing it with thresholds T_μ, such that if $m(\gamma)/u(\gamma) > T_\mu$ then $P(A_1|U) < \mu$, and T_λ, such as if $m(\gamma)/u(\gamma) < T_\lambda$ then

[1]Note that we use the term *mapping* to refer to a correspondence between a pair of data instances (database records or RDF individuals). This is different, for example, from [38] where such correspondences are called *mapping rules*, and the term *mapping* denotes a set of mapping rules.

$P(A_3|M) < \lambda$, where μ and λ are desired error levels. The challenges in this model include calculating $m(\gamma)$ and $u(\gamma)$, threshold values T_μ and T_λ, and the form of the comparison function γ. Subsequently many approaches were proposed to determine these parameters.

In the method proposed by Fellegi & Sunter together with their model, components of the vector $\gamma = (\gamma^1, \gamma^2, ..., \gamma^K)$ are the results of pairwise field values comparison, which are assumed to be mutually statistically independent. This assumption allows calculating $m(\gamma)$ and $u(\gamma)$ using the Naïve Bayes approach. In [124] an approach using the *general expectation maximisation* algorithm was suggested for calculation of $m(\gamma)$ and $u(\gamma)$, which does not require conditional independence of γ^i.

Subsequently, newly developed machine learning techniques were applied to the record linkage problem as alternatives to the Fellegi-Sunter method. In particular, [20] experimented with supervised algorithms such as binary decision trees, linear discriminant analysis and "vector quantisation", which is a more general version of the nearest neighbour algorithms. Other applications of the adaptive learning approach include support vector machines [9] and conditional random fields [77].

In addition to different methods for calculating the decision rule over the vector distance function, another challenge included measuring each component of the vector γ, i.e., similarity between field values. Most approaches consider field values as strings and try to measure similarity between two string values[2]. One of the earliest approaches was the edit distance (or Levenshtein distance) [72], where the

[2]String similarity indicates a similarity of representation between two character strings and does not imply any similarity of meaning between two strings.

distance between two string values depends on the number of edit operations (*insert/delete/replace*), which need to be applied to characters, in order to transform one string into another. Subsequent studies proposed several improvements of the edit distance function taking into account regularities occurring in real-world text data. For example, the affine gap metric [122] introduced additional edit operations (*open gap* and *extend gap*), which improved the performance when dealing with truncated or abbreviated strings. The Smith-Waterman distance [112] further developed the affine gap metric by reducing the cost of mismatches at the beginning and at the end of strings. The Jaro distance [61] uses a different distance calculation function, which takes into account the number of common characters, the number of transpositions and the lengths of both strings. The Jaro-Winkler metric [125] adjusts this function to give higher weights to prefix matches. In another algorithm proposed by Monge and Elkan [80] the matching is performed at the level of atomic strings (substrings delimited by punctuation symbols) rather than at the level of characters. All above-listed metrics have their advantages and disadvantages and there is no optimal distance measure. For example, the Monge-Elkan metric performs best over long multi-word strings, but does not capture small differences caused by mistypes; Jaro and Jaro-Winkler distances were specifically developed for comparing peoples' names in the census domain and so on. Thus, choosing an appropriate field matching technique has to take into account the domain and data features of the task at hand.

The classical Fellegi-Sunter model assumes that all attributes which are relevant for determining the equivalence of two instances are contained in the attribute vector. This assumption does not hold for scenarios where the relevant data are distributed

between different instances, which are related to each other. Approaches which analyse relations between data instances of different classes have received significant attention in recent years. In order to make a decision about whether two instances refer to the same entity, these approaches analyse other instances belonging to the neighbourhood graphs of two instances in question and mappings between them.

One algorithm which focuses on exploiting links between data objects for personal information management was proposed in [32], where the similarities between interlinked entities are propagated using dependency graphs. The `MOMA` system [117] employs a set of matching algorithms (or *matchers*). Among them there is a neighbourhood matcher, which considers mappings between related entities. `RelDC` [17] proposes an approach based on analysing entity-relationship graphs to choose the best pair of coreferent entities in the case where several options are possible. The authors of these algorithms reported a good performance on evaluation datasets and, in particular, a significant increase in the performance achieved by the analysis of relations between entities. Given the nature of Semantic Web data, where relevant data properties (fields) are likely to be distributed between several individuals and the number of attached properties (columns) is not fixed, relation analysis is especially important for coreference resolution.

Alternative record linkage approaches try to involve additional knowledge, not represented explicitly in records and field values. For instance, in [29] the authors propose the use of domain knowledge in the form of profilers: specific restrictions constraining instances of certain concepts. Such profilers are used to eliminate incorrectly created equivalence mappings, which are inconsistent with available domain knowledge. For instance, if a person “Mike Smith” with the age “9” is considered

equivalent to a person "Mike Smith" with the salary "200K", such a mapping can be eliminated if there is a corresponding profiler rule about people, restricting such combinations of age and salary. `Soccer`, proposed by Shen et al [107], takes into account characteristics of the sources, from which the records to be matched originate. In particular, the *semantic ambiguity* of a source is taken into account, i.e., whether there are duplicate records referring to the same real-world entity inside the source. This allows adjusting the record linkage procedure: for less ambiguous data sources, where duplicates are unlikely to appear, a more "relaxed" matcher is used, which requires less evidence to decide that two records can be matched. A more"conservative" one (or just a more restrictive threshold value for the same matcher) can be used for sources containing a high proportion of ambiguous records. Again, considering Semantic Web data, which (i) can be distributed across many sources and (ii) may define more sophisticated data constraints, it is even more important to involve domain and provenance knowledge into the coreference resolution procedure than for the traditional database integration problem.

2.2.2 Inconsistency resolution

While major research efforts have concentrated on resolving schema-level conflicts, data-level conflicts have received comparatively less attention in the database research community. Among the first practical database integration systems which considered resolving data value conflicts were `TSIMMIS` [89] and `HERMES` [1]. These systems advocated the use of mediators - special software modules that are adjusted for integrating information from specific data sources. A conflict resolution strategy was hard-coded in the implementation of a specific mediator. This can be regarded as

a disadvantage of the mediator-based approach since the integration of a new source required implementing a new mediator to process it.

A more generic approach was proposed in the `Fusionplex` system described in [82]. This system considers a set of factors including provenance of conflicting pieces of data, such as timestamp, source access cost, or source priority, and the content itself. So it is possible to define conflict resolution rules, which, for instance, select among two conflicting values the most recent one, the one coming from a more reliable source, an average between the two, etc.

The `Humboldt-Merger` system [11] follows a similar approach and outlines several conflict handling functions such as "Choose", "Vote", "Average", "Median", etc. This system allows the user to use these functions to define SQL queries that perform fusion over the returned values.

The authors of the `KRAFT` architecture [92] considered a special case of combining information from distributed on-line data sources. The challenge here involves achieving the fusion of knowledge expressed as constraints against an object data model rather than data records themselves. The architecture looks to solve a constraint satisfaction problem using data from multiple sources. In the described scenario, the data sources provide information about different PC components, which must be combined according to the user's requirements. `KRAFT` implements a multi-agent framework, where different types of agents collect user requirements and constraints from different data providers, combine them and use them to select necessary components from providers so that the configured system satisfies the user needs.

Several approaches assume that the degree of uncertainty associated with data values is provided explicitly in the form of probabilities. These approaches do not

actually resolve conflicts but provide an answer in the form of a probability distribution. In [74] the authors propose to use the Dempster-Shafer theory of evidence to resolve value conflicts. The method assumes that attribute values in conflicting sets have attached belief functions and calculates the resulting belief distribution among conflicting values by combining these functions. An alternative Probabilistic Data Model approach [4] uses probability theory as an uncertainty management formalism and proposes a special probabilistic relational algebra. The difference between these approaches is essentially the same as the difference between the underlying formalisms: probability theory assigns probabilities only to atomic values while the Dempster-Shafer theory allows belief function assignment over sets of values (without specifying whether elements of a set are equally probable or not). In the area of XML data integration a data fusion algorithm described in [58] proposes to resolve inconsistencies by means of uncertainty representation formalisms such as probability theory, possibility theory and the Dempster-Shafer theory of evidence.

Value conflicts, which are handled by most of the algorithms described above, are relevant for the Semantic Web data as well: in OWL terms, they represent violations of functionality or cardinality restrictions imposed on properties. However, given the richer structure that can be defined in ontologies, such conflicts constitute only a limited subset of possible situations leading to inconsistencies in the semantic data fusion scenario. Thus, such methods, although they can be adapted in the RDF data context, are insufficient and new specific methods must be developed for semantic data. These new methods, however, can still benefit from many of the foundational techniques embedded in the database conflict resolution algorithms. In particular, this applies to established uncertainty representation formalisms, such as probability

theory and the Dempster-Shafer theory of evidence.

2.3 Logic-based approaches

The primary problem which arises when fusing knowledge bases containing information expressed in formal logic is inconsistency [18]. In classical logic it is impossible to reason over inconsistent knowledge bases because, by definition, an inconsistent knowledge base becomes trivial: any formula can be inferred from it, thus making reasoning useless. This feature is known as the principle of explosion. One possible approach to allow reasoning about inconsistent information is to weaken classical logic. Paraconsistent logic [23] rejects the principle of explosion, thus accepting the presence of inconsistencies. However, as a consequence, it has reduced inferencing capabilities: for instance, the disjunctive syllogism $((\alpha \vee \beta) \wedge \neg\alpha) \rightarrow \beta$ does not hold. Another compromise approach modifying classical logic is to use multi-valued logic, such as a four-valued Belnap logic [5], which allows one of four values to be assigned: "true", "false", "unknown" or "both" (it is employed, for instance, in [57]). Alternatively, modal logics [79] provide another way of reasoning with inconsistent information because they formalise the notions of possibility and necessity. Without adopting an alternative formalism, reasoning with inconsistent data is still possible if the inconsistent parts of information are circumvented. For example, minimally inconsistent subsets of information can be isolated using a diagnostic procedure [96] and not taken into account during reasoning.

Such mechanisms make it possible to reason about data containing inconsistencies but do not provide a way to resolve them. Additional data analysis is needed to

fix inconsistencies. Similarly to inconsistency resolution approaches listed in section 2.2.2, this analysis may take into account the fused data itself (object-level information), additional meta-level provenance information, and domain information. The authors of [69] define several logical properties, which a fusion operator should satisfy. As stated in [51], every fusion operator that satisfies these properties corresponds to a family of pre-orders on interpretations, which allow distinguishing between "more desirable" and "less desirable" interpretations of a merged knowledge base. Hence it can be used to select beliefs which should be weakened or deleted in order to make the merged knowledge base consistent. The authors distinguish two classes of fusion operators: majority and arbitration. Majority operators, as their name suggests, aim to resolve conflicts according to the opinion of the majority, while arbitration ones try to find a compromise solution which maximally satisfies all conflicting pieces of data. In general, a major distinguishing property for a logic-based fusion algorithm is the way it imposes an ordering over possible outcomes. Meta-level information, in particular provenance, is crucial to establish this ordering.

With respect to description-logic knowledge bases, the techniques proposed in [103] and [64] employ Reiter's hitting set tree diagnosis algorithm [96], which produces diagnoses. These are sets of axioms whose removal would restore consistency to knowledge bases. Choosing between diagnoses then can be done based on additional factors such as minimality (how many axioms have to be removed), axiom relevance (how many other axioms will affect the deletion of a particular one), and others. A weakening operator proposed in [93] weakens concept subsumption axioms $(C \sqsubseteq D)_{weak} = (C \sqcap \neg\{a_1\} \sqcap \cdots \sqcap \neg\{a_n\})$, where n is the number of individuals to be removed

from C, and assertion axioms $\phi_{weak} = \begin{cases} \forall R.(C \sqcup \{b_1, \cdots, b_n\})(a) & if \phi = \forall R.C(a) \\ \top(a) \; or \; \phi & otherwise \end{cases}$

Thus, when possible, the approach tries to list exceptions explicitly rather than to delete an axiom.

In summary, logic-based fusion approaches have focused on inconsistency resolution rather than the coreference resolution issue. They assume that they are dealing with symbolic information represented in a formal logic-based language and focus on inconsistencies caused by logical inferencing. In contrast, the techniques developed in the database community put more emphasis on data representation issues relevant to conflicts (e.g., different formats of textual values, varying precision of numeric data, etc.). Both kinds of problems are relevant for semantic data. On the one hand, semantic data is usually represented in a logic-based language such as OWL, and more complex patterns of inconsistencies can arise than value conflicts. On the other hand, given that the semantic annotations are extracted automatically from natural language sources or raw data, this information is not purely symbolic because of the availability of datatype properties, whose values may be formatted differently, contain typos or measurement errors. Thus, in the semantic data fusion scenario formal logic-based methods can be seen as complementary to the database inconsistency resolution algorithms but are not sufficient when used in isolation.

2.4 Information integration in the Semantic Web

The nature of Semantic Web data makes research conducted both in the database community and in the area of formal logic relevant to the semantic data integration

problem. On the one hand, despite a different structure (graph vs table), RDF describes real-world entities in a way similar to relational databases: an individual (record) can be a subject or object of a number of instantiated properties (fields). Many basic techniques developed or applied in the database domain are reused in the Semantic Web context. These include string similarity distance measures [39, 62], machine learning algorithms [31], probabilistic techniques [15], etc. On the other hand, ontological languages such as OWL are based on the principles of formal logic. They allow definition of more kinds of constraints and interdependencies between pieces of data than it is possible when using a database schema.

2.4.1 Ontology matching

Semantic Web research has so far been primarily concentrated on the schema-level information integration (ontology matching) [38]. Ontologies have been often considered as a whole: data-level problems have been mostly treated as auxiliary and usually tackled together with schema-level matching. The primary reason was that, until the emergence of the Linked Data initiative (section 1.1), there was a lack of substantial volumes of semantic data covering overlapping domains, and, therefore, there was no specific need to focus on the data-level integration issues.

One classification of ontology matching approaches [110] divides them into two major categories with regard to their granularity:

- *Element-level* ones analyse schema concepts and data instances in isolation but not relations between them.

- *Structure-level* ones focus on relations between entities and the ontology as a

whole.

In the database community work discussed in section 2.2.1, the Fellegi-Sunter model [40] and its subsequent applications belong to the element-level category, while recent relation-analysis algorithms [111, 32, 63] operate at the structure level.

With respect to input interpretation [38], the methods can be classified into:

- *Syntactic*, which consider the structure of input data.
- *External*, which exploit auxiliary resources in order to interpret the input.
- *Semantic*, which use formal reasoning techniques.

Most of the commonly used techniques for instance matching belong to the first group. At the element level these again include string similarity techniques, implemented by such ontology matching systems as `COMA` [28], `OLA` [39], `Falcon-AO` [62], `QOM` [34] and others. A second group of syntactic element-level algorithms is constraint-based: these algorithms use internal structure of entity definitions, such as key properties and domain restrictions. In particular, they assume that objects are more likely to match if objects related to them also match and, conversely, mappings which contradict existing knowledge contained in one of the matched ontologies are unlikely to be accurate [30]. Syntactic approaches at the structure level treat matched ontologies as graphs and try to match these graphs (e.g., [78, 34], etc.).

External reference resources used in ontology matching include language databases such as WordNet [127] and external ontologies [2, 100]. However, since these resources mostly describe general knowledge (concept-level rather than instance-level), their use for instance matching is limited.

Semantic techniques concentrate on logical reasoning about matched ontologies. They reason about the original ontologies combined with the initial set of mappings. This reasoning helps to validate candidate mappings, revoke incorrect ones and possibly infer new ones [13, 47].

As was already pointed out, most ontology matching systems concentrate on schema alignment, but they also introduce some novel techniques addressing semantic data integration. Classical database record linkage algorithms primarily rely on element-level syntactic methods. More recent ones combined these with structure-level syntactic techniques. Ontology matching tools, in addition, could exploit richer ontological structure descriptions drawing on semantic methods. These semantic techniques are valuable for the semantic data fusion problem.

2.4.2 Semantic data integration

Until recently, the Semantic Web community has concentrated efforts on the schema matching problem. Now, with a constantly increasing amount of RDF data being published according to the Linked Data standards, the problem of instance-level integration is gaining importance. Dealing with RDF data sources distributed over the Web requires solving a fundamental problem of representing and managing information about URIs referring to identical entities. There are different possibilities, and several proposals have been put forward within the research community.

The standard OWL language defines the *owl:sameAs* property, which denotes full equivalence between individuals: URIs linked by this property become mutually interchangeable. This property is widely used to connect identical individuals in the Linked Data repositories. It has been argued that using this property for identity

representation for Semantic Web data on a global scale leads to several problems [60]. First, two URIs connected by the *owl:sameAs* predicate become indistinguishable, which may lead to the distortion of information if they were used in different contexts originally. For instance, one URI can denote "Russia" in a historical context (thus including Novgorod Republic, Kievan Rus', Grand Duchy of Moscow, Russian Empire, and USSR) while another may refer to the Russian Federation, a legal entity in existence since 1991. Second, technical efficiency problems may arise because an application gathering information about an entity may need to crawl the Semantic Web by following all *owl:sameAs* links from a single URI recursively until all routes are explored.

The `OKKAM` project [14] proposes a solution based on a centralised Entity Name System (ENS) for the Semantic Web. This ENS issues and maintains absolute URIs for all entities on the Semantic Web. In this scenario the user (or the client application) must query an ENS service for a suitable URI for an entity, instead of creating a local URI. In response the ENS either returns an existing URI or generates a new one. While potentially providing a central authority for identity and coreference resolution issues occurring on the Semantic Web, the `OKKAM` scenario also leads to some new problems and complications. These include difficulties with capturing context-dependent differences in the usage of the same URI in different sources. Also making irrevocable coreference decisions relying on automatic coreference resolution techniques can lead to errors. Not least there is the issue of persuading all semantic data providers to use the ENS service.

An alternative approach to ENS called the Consistent Reference Service (CSR) architecture was outlined in [60]. This approach proposes to maintain locally created

synonymous URIs but to store synonymity information in special CSR knowledge bases separately from the actual data. These CSR knowledge bases are assumed to be constructed by data providers themselves for their data repositories. Equivalent URIs (both internal for the data source and external) are grouped together into sets called "coreference bundles", which also may specify a canonical URI for an entity. This approach is possibly safer than a centralised one: local URIs are preserved, which leaves the client application a choice whether to treat them as equivalent or to reject the synonymity asserted by the CSR. Hence context-dependency can be preserved: a single CSR only specifies that URIs are considered equivalent in the context assumed by the CSR owner. Other CSRs can define different equivalence sets for the same URIs.

A particularly interesting approach is `idMesh` [22], where maintenance of links constitutes a complementary stage of the link discovery process: the system combines coreference links into graphs and considers their impact on each other to reason about their correctness and reliability of their sources.

Given the amount of data to be handled on a Web scale, the need for automatic coreference resolution techniques is recognised in the Semantic Web community [14], [49], [43]. Among the existing systems, `Sindice` [118] implements a straightforward method for coreference resolution by utilizing explicitly defined key properties (*owl:inverseFunctionalProperty*). Individuals which have equivalent values for such properties are considered equivalent. The approach yields high precision, but can only be applied to a limited subset of data where such properties are defined explicitly and have values in a standard format. Other tools implement approximate matching techniques similar to those created in the database integration and ontology

matching domains. SILK [120] employs different string-based distances and provides a configuration language to specify the metrics to use, thresholds and aggregation mechanisms for specific datasets. All these parameters have to be manually defined by the user. In this way, SILK can be seen as an adaptation of the classical Fellegi-Sunter model designed for database record linkage (section 2.2.1). As such, it has the same limitations, in particular, it ignores relevant types of evidence: the structure of the semantic data graph and knowledge defined in the ontology.

HMatch [43] was initially designed for the schema matching task and later incorporated data integration capabilities. This made the tool applicable to knowledge bases using different ontologies. First, HMatch produces schema-level mappings and then uses them to compare corresponding properties. Descriptions of individuals are represented as graphs containing relevant assertions about them. The algorithm calculates similarities between datatype property values of two individuals and then calculates the aggregated similarity between individuals using the Dice coefficient method:

$$sim(i_1, i_2) = \frac{similarproperties}{totalproperties} = \frac{2 * m}{n_1 + n_2},$$

where n_1, n_2 are the total numbers of compared properties of i_1 and i_2 and m is the number of pairs of properties whose values were found similar[3]. RDF-AI [75] concentrates on the data-level issues which occur when combining datasets using the same schema. The algorithm builds on string (Monge-Elkan) and linguistic (WordNet) similarity measures to calculate similarities between literal property values, and then invokes an iterative graph matching algorithm akin to similarity flooding [78]

[3]Here, as in the case of string similarity, the term "similarity" primarily refers to the similarity of representation of two individuals and not to the similarity of the meaning. In the following chapters we also use the term "similarity" in this sense when referring to the similarity between individuals.

to calculate a distance between individuals. These algorithms involve more complex coreference resolution techniques than SILK. However, these tools do not address the inconsistency resolution problem.

2.5 Fusion systems combining different methods

Since there is no single best algorithm for all domains, and a method can have different optimal configuration parameters when applied to different data, it is often the case that one system employs several methods. The survey of ontology matching systems given in [38] shows that the majority of currently available state-of-the-art ontology matching systems employ a combination of several basic algorithms. Here we only discuss a few of them to illustrate common approaches to method combination.

COMA [28] features parallel composition of matchers of several types:

- *Simple matchers* including several standard element-level string similarity matchers and a constraint-based one, which checks the compatibility of datatypes.

- *Hybrid matchers*, which use a fixed combination of simple matchers. For example, NamePath is an element-level matcher, which compares elements' long names, including the names of all superconcepts.

- *Reuse-oriented matchers*, which exploit mappings created by the system when applied to different (but similar) ontologies.

Each invoked matcher produces a similarity value; these values are aggregated afterwards. The system has several aggregation strategies like *Min*, *Max* and *Average*,

and evaluation results had shown that the *Average* strategy [28] led to the best performance.

`oMap` [115] includes string similarity matchers, a Naive Bayes classifier operating on instance data (all text properties of instances belonging to a class are taken as a bag of words), and a structural classifier, which uses OWL definitions of ontological entities to propagate initial similarities produced by other matchers. Here matchers can be sequenced: the structural matcher requires an initial set of similarities as input.

Some frameworks implement specific strategies to adjust parameters depending on the use case. `Falcon-AO` [62] contains two modules: LMO (Linguistic Matching for Ontologies) and GMO (Graph Matching for Ontologies). The first uses a combination of string similarity over entity names and set similarity over sets of terms from entities' contexts. The second views ontologies as graphs, represents similarity between two entities as a linear combination of similarities between adjacent entities, and solves the resulting system of equations. The components are invoked in sequence (first LMO, then GMO) and the reliability of the output of both modules is estimated from the linguistic and structural compatibility of ontologies. Linguistic compatibility is measured from the number of mappings with high similarity produced by LMO and the size of ontologies. Structural compatibility compares the occurrences of built-in standard RDF, RDFS, and OWL properties.

While `Falcon-AO` has a rigid structure with two components, some frameworks implement a more flexible approach, where the configuration parameters of an extensible library of methods can be adjusted. The `FOAM` framework [33], primarily designed for schema-mapping, includes a special configuration architecture `APFEL` [35], which

learns optimal parameters of matching methods by exploiting user feedback. eTuner [70] aims at the same goal but constructs an artificially distorted version of the ontology to be mapped. The mappings between the original ontology and its distorted version (known in advance) are used as a gold standard for the learning algorithm, which produces the configuration parameters for atomic methods.

The systems listed above were designed for the schema-matching problem. MOMA [117] is a recent adaptation of COMA [28] specially developed to handle instance-level coreferencing. The system also employs an extensible library of matching methods, each conforming to a uniform interface, invokes them separately and combines their results afterwards in a way similar to COMA, but focuses on the instance coreferencing problem.

The ILIADS system [119] combines schema-matching and instance-level matching in an iterative workflow, where a schema-matching stage is followed by an instance-matching one and vice versa. In this way results obtained at one stage provide additional evidence for the next one by applying logical inferencing to discover new mappings.

In our view, there is still a significant need to adjust existing approaches for the Semantic Web data integration task. First, as we said, most of the ontology matching systems primarily focus on the schema-level matching and are not optimised for dealing with data-level issues [38]. In particular, schema-level matching systems, which employ a combination of individual matching techniques, try to select the optimal algorithms' parameters for a pair of ontologies (e.g., [70, 62]). Data-level coreferencing requires more fine-grained tuning: optimal decision models for individuals belonging to different classes of the same ontology might vary. We will summarise these

limitations, which have to be addressed, in the next section.

2.6 Limitations of existing approaches

As already pointed out, Semantic Web data combine features of both relational databases and symbolic logical knowledge bases, and these research communities contributed to the study of the fusion problem as well as the Semantic Web community itself.

In all these domains techniques were developed which could be adopted to address semantic data fusion. These techniques complement each other because they address different subtasks of the fusion problem (coreference resolution and inconsistency handling) and rely on different kinds of evidence (attribute similarity, graph similarity, ontological axioms, etc.). Because of this, the need to use different techniques in combination was recognised, and tools combining different methods were developed. However, several issues, which are important for the semantic data fusion task, were not addressed by existing tools.

First, existing tools target either the coreference resolution problem or the inconsistency handling one but do not address them in combination. However, these subtasks are not independent because the errors made at one stage can influence the other. For example, an inconsistency can arise because some property of an individual was incorrectly extracted or because properties belonging to two different individuals were incorrectly assigned to the same individual by a coreference resolution algorithm. Since both noisy data and incorrect coreferencing may contribute to inconsistencies, it also makes sense to consider the degrees of uncertainty introduced

by both these factors together. This is impossible in approaches in which coreferencing and inconsistency resolution are completely separated, which we call the *subtasks interdependency* limitation.

Second, as mentioned in section 2.5, the data-level integration problem has a different *problem granularity* from the schema-level one. While in the existing systems the methods are selected and configured once for each pair of input datasets, this is not sufficient for the data fusion task. Different methods can be optimally suited for different parts of datasets, and the same method may require different configuration settings for different types of data. For instance, if we need to fuse datasets containing scientific publications, we need to deal with authors' names and paper titles. While, for example, the Jaro-Winkler similarity measure was specially developed for comparing person names, it is not optimally suited for paper titles. Thus, a coreference resolution algorithm employing string distance metrics will need to be configured differently for individuals of these two classes.

In the next section we will discuss how we have chosen to address these limitations.

2.7 Addressing the limitations

As we discussed, there is a multitude of complementary methods which can be applied to perform data fusion. They rely on different kinds of evidence which can be relevant to a different extent depending on the application scenario. For example, inconsistency handling techniques drawing on formal logic (section 2.3) can be more useful if the domain ontology has the OWL-DL expressivity than if it is represented in RDFS. Some methods are generic while others can be developed for a specific

domain. It is not feasible to specify in advance the whole range of methods needed to perform data fusion in all domains and use case scenarios. In order to maximise the reusability of our system, we decided to implement it as an extendable modular architecture where alternative, mutually complementary methods can be included, configured, and invoked depending on the application scenario.

To address the first limitation, the architecture has to tackle both fusion subtasks which we mentioned: coreference resolution and inconsistency handling. However, since these subtasks are mutually dependent, the architecture needs to ensure that no information is lost when passing the results of one stage to another one. In particular, an inconsistency resolution method should be able to know which parts of data were affected by the coreference resolution methods and to estimate, how reliable the output of these methods is. We addressed this problem by employing uncertainty reasoning: confidence weights are assigned both to the data statements (estimating the reliability of sources and extractors) and to the decisions made by algorithms (estimating the reliability of their output). The reliability measures assigned to data statements indicate the degree of confidence that information extracted from a specific source (e.g., web-site or database) is correct with respect to the real world. The reliability of methods' output indicate the degree of confidence that a decision made by an algorithm is correct (e.g., a decision about the identity of two individuals or about the optimal resolution of a conflict). These confidence weights are available to make decisions at the later stages of the workflow.

In order to overcome the problem of granularity limitation, the architecture had to be designed in a way that allows the fusion algorithms to be configured depending on the type of data they process. Thus, the architecture must be able to utilise

information from the domain ontology at the method configuration stage. Because of that, we used the *problem-solving method* approach for the architecture design. Different fusion subtasks and methods are formally described in terms of our *fusion ontology*, and method configuration parameters can be linked to specific concepts of the domain ontology: thus, for example, we can specify that a coreference resolution method should use the Levenshtein string distance for individuals of unknown types, the Monge-Elkan distance applied to the label for the *foaf:Document* individuals, and the Monge-Elkan distance over the label, venue and publication year for *foaf:Document* individuals which are also of the type *sweto:Publication*.

We designed the core version of our KnoFuss architecture to deal with these limitations to perform data fusion in the corporate knowledge management scenario studied within the X-Media project (see section 1.1). This scenario assumes that no schema-level heterogeneity is present, and therefore only data-level issues (coreference resolution and inconsistency resolution) have to be solved. After we implemented and tested this core version, we looked at the more generic data integration scenarios involving schema heterogeneity. This was motivated by the recent appearance of the Linked Data publishing standard and the need to perform coreference resolution between Linked Data repositories. To deal with this scenario, we extended the core version of the architecture to include schema matching methods and, in addition, implemented our own schema matching algorithm, which draws on pre-existing partial sets of Linked Data coreference links to assist the data fusion process.

2.8 Summary

In this chapter we discussed the existing work relevant to the semantic data fusion problem. We observed that Semantic Web data combine features of both relational databases and symbolic logical knowledge bases. The information integration problem has been studied in both these domains and multiple approaches have been developed. The main relevant perspectives in the database research on data integration include:

- the use of attribute similarity measures for record linkage;
- the use of provenance for inconsistency resolution.

Research in formal logic reasoning contributed valuable inconsistency handling techniques. These techniques have been adopted in Semantic Web research [38], [64], and novel ones emerged to exploit the semantic structure defined by standard ontological languages. However, because until recently the semantic data were not available in large volumes and instance-level fusion was not required, these ontology matching methods primarily focus on schema-level integration.

Directly applying these algorithms to the data fusion problem, however, can lead to a number of problems, in particular:

- Database record linkage systems are well suited to handle the coreference resolution issue, but they do not take account of specific properties of ontological data, such as hierarchical relations between classes and specific data restrictions defined by the OWL language axioms.
- Inconsistency handling strategies based on formal logic do not take account of coreference resolution results. These results can introduce additional noise to

the resulting knowledge base and have to be analysed.

- Ontology schema matching tools make use of semantic relations defined by ontologies, but do not consider instance-level matching in detail (e.g., the differences in problem granularity are ignored).

- Recently, tools supporting semantic data integration at the level of individuals have begun to appear but they do not handle the whole range of fusion subproblems. Most of them only focus on the coreference resolution stage.

Hence, we can conclude that for each fusion subtask there is a range of possible approaches, and that the relevant algorithms can be mutually dependent, since the output of one can serve as input for another. The consequences of these are:

- It is important to have an architecture to manage the fusion process. The architecture should support the flexible combination of relevant methods to produce a fusion workflow that is optimally suited to the specific kind of data at hand.

- Uncertainty can be present both as a result of imperfect data and also as a result of incorrect decisions made by intermediary algorithms. Hence, it makes sense to consider the uncertainty of both these factors together, which is impossible in cases where coreferencing and inconsistency resolution are completely separated.

The following chapters describe how we have sought to handle these issues. Chapter 3 describes the design of our proposed architecture supporting the combination of different methods. Chapters 4 and 5 are dedicated to the problem of handling

uncertainty and describe our approach which combines formal uncertainty modelling with logical reasoning.

Chapter 3

KnoFuss architecture

This chapter deals with the design of the KnoFuss architecture. First, a brief description of the architecture is provided. After that, the chapter outlines the architecture requirements for fusing RDF data coming from different sources. Then, the design approach based on problem-solving methods is introduced, and we discuss how this approach can help to meet these requirements. We will then describe how the KnoFuss architecture organises the problem-solving method library using the fusion ontology to describe method specifications and to guide method selection and execution. Finally, we describe how the architecture assists the configuration of problem-solving methods for specific domains using machine learning.

The material presented in this chapter contributed to the conference paper:

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2008). Integration of semantically annotated data by the KnoFuss architecture. 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008), Acitrezza, Italy.

3.1 Introduction

As was discussed in Chapter 2, there is a range of existing algorithms which tackle different data fusion subtasks: e.g., record linkage algorithms resolving coreferences between individuals (section 2.2.1) and different logical reasoning mechanisms dealing with inconsistencies in the integrated knowledge bases (section 2.3). Since there is no single best approach suitable for all domains and scenarios, existing data integration systems commonly use different basic algorithms in combination. However, as was pointed out in section 2.5, existing systems combining different data integration algorithms have their limitations. One of them is the granularity issue: different subsets of data may require different configurations of methods to process them even within the same fusion session. Another limitation concerns the interdependencies between fusion subtasks, which cannot be captured if these subtasks are tackled in isolation. For instance, an inconsistency resolution algorithm processing a merged knowledge base can benefit from knowing both about the quality of original data and about the quality of the output of the coreference resolution algorithm which was employed to merge identical individuals.

In order to address these limitations we developed an architecture called KnoFuss, which focuses on the instance-level integration of data structured according to OWL ontologies. In this chapter we discuss the core version of the architecture, which works under the assumption that data to be merged are already structured according to the same OWL ontology. This is a valid assumption, for instance, in the corporate knowledge management scenario (section 1.1). The architecture takes as input two knowledge bases: the *target knowledge base*, which contains the original data, and

the *source knowledge base*, which contains some new data to be integrated. As an output the architecture produces an updated version of the target knowledge base, into which the data from the source knowledge base are merged.

KnoFuss carries out two main subtasks of the knowledge fusion process:

- *Coreference resolution.* This stage focuses on discovering individuals in two knowledge bases which represent the same real-world entities. The output of this stage is a set of mappings between such coreferent individuals.

- *Knowledge base updating.* At this stage the system makes decisions about the resulting state of the merged knowledge base: which statements can be directly inserted into the target knowledge base and which should be modified or removed. In order to perform this subtask the architecture analyses data interdependencies, e.g., inconsistencies caused by contradictory statements.

Each of these subtasks can be performed by different methods, both generic and domain-dependent. In order to support the selection of optimal methods depending on the domain, the architecture considers the basic techniques for each fusion subtask as problem-solving methods. Each method represents a separate module, formally described in terms of its inputs, outputs, and applicability conditions while its internal behaviour is hidden from the system. The main components of the architecture are:

- *A library of problem-solving methods*, containing algorithms dealing with atomic tasks.

- *A fusion ontology*, describing the information needed to guide the fusion process.

The capabilities of the methods are formally described using the fusion ontology, which is itself represented in OWL and provides two kinds of knowledge structures:

- *Descriptors of problem-solving methods, tasks and application contexts.* This information is used to select and configure methods.

- *Intermediate knowledge structures.* These structures represent meta-level descriptors of the methods' inputs and outputs (e.g., mappings between individuals and sets of conflicting statements).

The fusion process is performed by the system as follows. The system receives as its input a source RDF knowledge base, containing new data to be integrated. Then all tasks of the fusion process (coreference resolution and knowledge base updating) are performed in sequence and produce as a result a set of statements to be integrated into the target knowledge base. Execution of each subtask is controlled by a generic workflow, which (i) selects appropriate methods, (ii) invokes the methods and collects their output and (iii) combines the output of methods filtering out redundancies.

The rest of the chapter is organised as follows. In the first part of the chapter we discuss the design approach based on the problem-solving methods and motivate our choice of this approach for the architecture design. Section 3.2 outlines the main requirements, which had to be taken into account when designing the framework. In section 3.3 we discuss the relevant work in the area of problem-solving method libraries, in particular, the principles which we found relevant to the fusion context. Section 3.4 discusses how the requirements have been addressed by applying an approach based on problem-solving methods.

In the second part of the chapter we discuss different aspects of the architecture in more detail. Section 3.5 deals with the decomposition of the generic fusion task into smaller subtasks, which can be executed by problem-solving methods. Section 3.6

describes the main concepts of our *fusion ontology*, which provides the specifications of problem-solving methods and the auxiliary data structures used in the fusion workflow. Section 3.7 describes the standard procedure used by the architecture to tackle fusion subtasks and illustrates the procedure with an example. In section 3.8 we discuss how the problem-solving methods can be configured for specific domains using machine learning and how the architecture exploits the domain ontology to achieve that. Finally, section 3.9 summarises our contribution.

3.2 Challenges and requirements

The task of fusing semantic data imposes a number of requirements, which a fusion architecture has to satisfy.

The first requirement is the *reusability* one. In order to be adjustable to different use cases the architecture has to be reusable across domains: no domain-specific knowledge should be hard-coded in the design of the architecture. Instead, the system must be capable of using two kinds of external knowledge needed to resolve fusion tasks: domain knowledge in the form of a domain ontology and problem-solving knowledge encoded inside problem-solving methods.

Second, efficient selection and configuration of algorithms, and aggregation of their results are necessary. Handling fusion even in a specific domain requires dealing with various types of data and instances of different classes. This motivates the *granularity* requirement: different algorithms can be employed with varying configuration parameters even within one fusion session, when fusing data belonging to a single domain.

Third, developing optimal techniques for each type of data and fine-tuning them for the maximal performance requires significant human effort, which might not be feasible in many scenarios. In order to achieve flexibility, the system has to be able to employ techniques at different levels of generality, using domain-specific methods when available or alternatively adjusting generic methods to a specific domain. We call this the *method-domain adjustment* requirement. The granularity and the method-domain adjustment requirements address the problem granularity limitation of existing systems, which was outlined in section 2.6.

Fourth, there is the *uncertainty management* requirement. The data to be integrated may originate from multiple sources of varying reliability. Some sources may contain incorrect or obsolete data. Additionally, this noise in data may influence automated algorithms handling separate fusion subtasks (e.g., coreference resolution) and cause them to produce incorrect results and introduce additional noise. Hence, in order to evaluate the validity of any fact in the integrated knowledge base, provenance of data has to be taken into account, and, in particular, the degree of uncertainty associated with data statements and problem-solving methods' decisions has to be explicitly represented.

The fifth requirement, closely related to the previous one, is the *dependency processing* requirement. Together, these requirements address the subtasks interdependency limitation mentioned in section 2.6. The methods handling different subtasks of the fusion workflow depend on each other as well as on the quality of the original data sources. At the later stages of the fusion workflow it should be possible to analyse the interdependencies between the data statements and the decisions made at the earlier stages.

Thus, the architecture has to preserve the provenance of information created during its workflow and to allow reasoning about it.

3.3 Related work

In Chapter 2 we discussed the need for a reusable fusion architecture, which could use several alternative methods in combination for handling fusion subtasks. The requirements outlined above include reusability across domains. Problem-solving methods (PSMs) have been proposed as a flexible means of constructing knowledge-based systems, which specifically focuses on these issues [16, 104]. Problem-solving methods are reasoning components that can tackle specific tasks and can be reused across applications [41]. The knowledge modelling approach which describes a knowledge-based system in terms of tasks and methods was proposed in [16]. Informally, "a task defines *what* needs to be achieved (a declarative functional specification), and a PSM *how* it has to be achieved (an operational specification)" [6]. This decoupling allows the system designer to specify alternative ways to solve the same problem by introducing several methods for a single task. Methods can define complete procedures for achieving a task's goal or introduce several lower-level subtasks, which in turn can be performed by their own methods, thus leading to task-subtask hierarchies. Libraries of methods can be used in two ways:

- At design time by a system designer constructing a knowledge-based system for a specific domain.

- At run time by a system, which automatically selects and invokes appropriate methods to perform a task in a specific context.

While initially many approaches primarily focused on the first of these options [104, 88], in our case both are relevant. First, if a system is configured to be used in a specific domain, the designer might need to adjust existing domain-dependent methods to be included into the library or to reconfigure some generic methods' parameters to accommodate domain-specific features. However, the second option is more important in our scenario: automated run-time method selection and invocation. As pointed out in the previous section, even in a single domain the fusion process may need to deal with various types of data, for which different methods may be optimal. Thus, methods have to be selected at run time depending on input data. Automated selection and invocation of problem-solving methods at run time requires these to be described formally in terms of their applicability to tasks, input-output data flow and additional assumptions about data. Benjamins and Pierret-Golbreich [6] outline four types of assumptions:

- *Epistemological assumptions.* These include the domain knowledge needed by a method as its input and in turn can be divided into two subtypes:
  - *Availability assumptions*, referring to existence of facts and instances required to fill input roles in input data.
  - *Property assumptions*, which include features of input data as a whole. For instance, in the data fusion context the ontological reasoner Pellet[1] can be used to detect inconsistencies in the knowledge base under assumption that the knowledge base is compatible with the OWL-DL language standard. Property assumptions cannot be verified by a simple existence check but

[1] http://clarkparsia.com/pellet/

may require more complex reasoning.

- *Pragmatic assumptions.* Requirements related to the external environment, in which the system operates (e.g., a method can be invoked only under Windows, or it can require the Java 1.6 virtual machine).

- *Teleological assumptions.* These assumptions are related to the goal to be achieved. Unlike the previous two types, if these assumptions do not hold, the method can still be applied, but its ability to achieve the goal, the correctness and completeness of its solution are not guaranteed. For instance, a coreference resolution method which compares key properties assumes that all individuals have values for these key properties. If this assumption does not hold, then the solution produced by the method will be incomplete.

In the fusion scenario, all types of assumptions are relevant. For example, an epistemological assumption for a coreferencing algorithm may select only individuals of a particular class as its input. Pragmatic assumptions may concern implementation details (e.g., a method may require a workstation with the Windows operating system and the .NET framework). Given that many relevant algorithms are statistical, it means that the goal may not be achievable for every input and mistakes are possible, which is an example of a teleological assumption. For instance, coreference resolution methods based on string similarity metrics only produce complete and correct solutions if all identical individuals have sufficiently similar values for their datatype properties, and these values are non-ambiguous and represented in the same format. The principle of using specific adaptor structures to refine the method's description and match it to the goal at hand allows these assumptions to be specified in a flexible

way. In our designed architecture we defer implementation-specific issues to actual methods and do not consider pragmatic assumptions at the level of methods. However, epistemological and teleological assumptions need to be taken into account. Another important feature of the PSM library approach is the use of ontologies to describe the capabilities of the system's components and the use of reasoning to guide the method selection and invocation process. These principles have been taken into account when designing the fusion architecture.

There are several proposed problem-solving method libraries, which use formal descriptions of methods and their assumptions to guide the method selection. Here we list a few of them.

`EXPECT` [116] uses a description logic-based representation called Loom to represent methods' capabilities. Capability descriptors define the actions of a method and its input and output roles, thus covering availability assumptions from the above list. Role-filler objects are specified using a shared domain ontology. The ontology defines hierarchical relations between concepts and allows the system to match a generic method with a specific goal.

`PRODIGY` [44] introduces statistical method invocation making use of methods' past performance. The `PRODIGY` system deals with the problem of choosing an appropriate search engine for a given problem. Thus, the time cost of the chosen method is crucial for the task and defines an important teleological assumption: in order to be valuable, the goal must not just be achieved but be achieved within a reasonable time frame. The selection mechanism estimates for each method the expected gain, based on both the expected reward for solving the task at hand and the past performance of the method (percentage of failures and time cost), and then selects a method on the basis

of these metrics.

TMDA (Task-Method-Domain-Application) [83] defines an ontology describing task-method structures and allows the specification of applicability conditions (epistemological assumptions) in the form of logical expressions, which must hold for the method. Both problem-solving method descriptors and domain knowledge are expressed using ontologies and complex reasoning can be performed to validate the expression.

UPML (Unified Problem-solving Method description Language) [42] describes a language and a specific ontology for describing problem-solving method libraries. Special adaptor structures are used to refine the problem-solving method descriptions and match them to tasks. Different kinds of assumptions are defined:

- preconditions, which specify constraints imposed on a method's input (these correspond to epistemological availability assumptions in the list above);
- assumptions, which describe constraints related to all available context knowledge (epistemological property assumptions);
- additional postconditions, which describe properties that apply to a methods' output;

These conditions can be defined both at the level of methods themselves and at the level of adaptors. The system allows the flexible refinement of a method's description depending on the task at hand.

When developing our KnoFuss architecture, we reused several design patterns of problem-solving method library organisation, which were relevant for the semantic data fusion application scenario. Our design was primarily inspired by TMDA [83] and

UPML [42]. The definitions of the main concepts of our fusion ontology (task and method descriptors) were based on the corresponding concepts in TMDA, in particular, we adopted the task decomposition modelling pattern and the relations which associate methods with tasks. Like in TMDA, the fusion ontology uses two properties for determining the applicability of a method: one (*Tackles*) to specify the subtask within the workflow, to which the method can be applied, and another (*Selection criterion*) to provide more fine-grained applicability conditions. The usage of special adaptor structures (the *Application context* concept in the fusion ontology) which associate a method with a specific application domain was adopted from the UPML library description. Also, similar to PRODIGY [44], the KnoFuss architecture takes into account the performance of a method: the estimated reliability of the method's output is used to make decisions at the later stages of the workflow and to resolve conflicts caused by this output. However, when designing the fusion ontology to be used in our KnoFuss architecture, we had to extend the high-level knowledge structures defined in TMDA and UPML, adjust them to the needs of the fusion task, and define additional, fusion-specific knowledge structures. These will be discussed in more detail in section 3.6.

3.4 Design rationale

The primary reason for the choice of problem-solving methods as a foundation for the design of the fusion architecture was the reusability requirements outlined in section 3.2: the architecture needs to be adjustable to different domains and has to handle different types of entities. By definition, problem-solving methods represent the

reasoning components of knowledge-based systems that can be reused across applications. The approach based on problem-solving methods makes it possible to abstract from implementation, domain, and task aspects [41] and to describe these aspects formally, which contributes to the reusability. Reusability across tasks is not relevant in our case because it applies only to high-level methods, which represent generic reasoning strategies suitable for achieving different goals (e.g., the Propose&Revise method has been applied to parametric design [83] and scheduling [95]). While there exist such generic methods, which in particular can be applied to fusion subtasks (e.g., K-nearest-neighbour classification can be refined for the instance coreference resolution task), this study considered a restricted set of tasks and therefore task-independence of methods was outside its scope. Thus, we are interested in two dimensions of method specifications [41]: problem-solving strategies and domain knowledge assumptions.

Complete separation of the two is impossible due to the interaction problem [76]: a problem-solving method and its specific variants cannot be constructed independently of assumptions about the available domain knowledge. These assumptions are not necessarily satisfied by all potential domains. For example, a rule-based coreference resolution method assumes that for individuals of a specific class in the ontology there exists a set of key attributes, which unambiguously describe its identity, while adaptive machine-learning algorithms assume that there exist enough training examples, for which the truth value is known. This issue is also strongly connected with the *granularity* and *method-domain adjustment* requirements: the same method can be reused in different domains, but its assumptions may differ and its behaviour may need tuning. Providing such methods with more information about the domain is

valuable for improving their performance but at the same time restricts their applicability and limits their reusability. Thus, domain and implementation specifications of methods cannot be separated and the architecture has to provide a mechanism for expressing the interdependencies between these two kinds of descriptions. In the KnoFuss architecture the *application context* concept is used for this purpose.

3.5 Task decomposition

Different subtasks which have to be handled in the fusion process were outlined in Chapter 2. Here we describe the task decomposition, which has been realised in the fusion architecture (see Fig. 3.1). As we mentioned in Chapter 2, the high-level distinction between schema-level and instance-level issues has been common in the literature for a long time [66, 81]. In our approach we accept this view and decompose the fusion task into *ontology integration* and *knowledge base integration*. We distinguish three main subtasks of the ontology integration task:

1. *Preprocessing* responsible for syntactic transformation (e.g., from DAML to OWL)

2. *Ontology matching* responsible for the creation of mapping rules

3. *Instance transformation* performing actual translation so that instances in both source and target knowledge bases are structured according to the same ontology

For the core version of the KnoFuss framework, we assume that we are dealing with datasets structured according to the same ontology, so actual ontology integration is

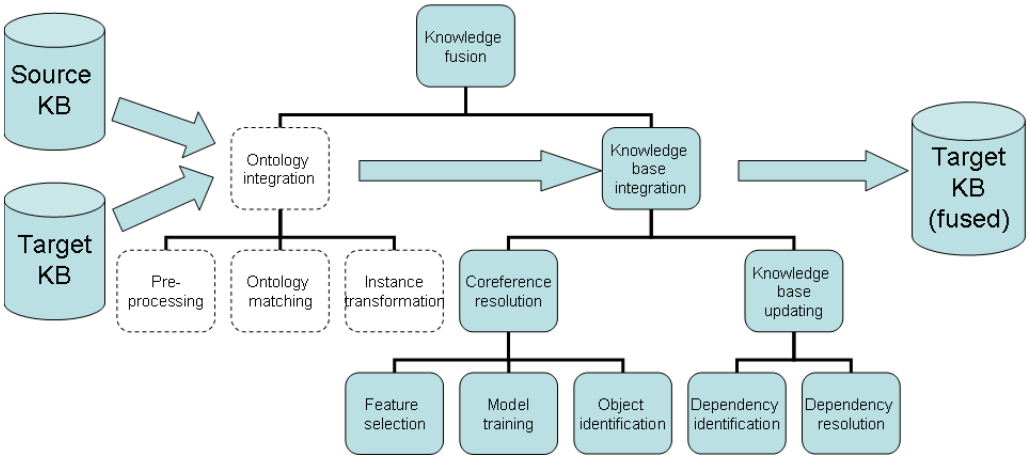

Figure 3.1: Decomposition of the fusion task into subtasks

outside its scope. We will discuss the scenario involving ontological heterogeneity in Chapter 7. The core KnoFuss implementation contains methods covering the second step of the fusion process: *knowledge base integration*. *Coreference resolution*, the first step of the process covers the instance ambiguity issue: identifying individuals referring to the same entity in two knowledge bases. In the classical theory developed for databases [40] this process is performed by comparing sets of attributes attached to individuals. Most existing algorithms [36] follow this theory. A typical coreference resolution algorithm makes its decision about equivalence of two individuals based on (i) a set of attributes (or features), which are considered relevant, and (ii) a decision model, which compares attribute values and produces an answer about whether or not two individuals should be merged. In some trivial cases both these components are known in advance: for instance, for individuals of a certain type a primary key consisting of several properties may be defined. In the general case special data analysis is required to select relevant features and machine-learning algorithms are used to find parameters of decision models. These procedures can be separated

from the actual decision-making regarding the identification of equivalent individuals. Thus, the coreference resolution task is decomposed into three main subtasks: *feature selection*, *model training* and *object identification*. After discovering potential mappings between individuals believed to be identical, the next step is to analyse the resulting knowledge base (assuming that the mappings are applied and mapped individuals are merged) and consider interdependencies between pieces of data and restrictions defined in the ontology. These interdependencies, in particular, include violations of ontological restrictions. Such violations may indicate a disagreement between sources, a wrong piece of data, or an erroneous mapping created by a coreference resolution method. Other ontological axioms may provide positive evidence for reinforcing coreference mappings: e.g., if two individuals linked via a functional property both have potential coreference mappings with another pair of individuals linked via the same property, then both coreference mappings should be reinforced. Analysis of such dependencies is necessary to decide what should be the final state of the integrated knowledge base: which new data statements should be inserted or removed, which coreference mappings should be applied or revoked, and how the degrees of confidence should be modified. This is the main goal of the *knowledge base updating* task. The first step in this process is *dependency identification*. Its goal is to localise sets of data statements and schema-level axioms, which influence each other. The second step is *dependency processing*, where these dependency sets are analysed and refined: e.g., erroneous mappings removed, confidence values associated with beliefs in conflicting statements updated, etc. After the dependency analysis task these refinements and coreference mappings are applied to knowledge bases and the data from the source knowledge base are merged into the main knowledge base.

3.6 Fusion ontology

As described before, the workflow of the architecture involves selecting appropriate methods for each particular task, invoking these methods and handling their results (in particular, combining them where necessary and passing them to the following stages). While the internal implementation of methods is hidden from the architecture, the architecture itself is responsible for method selection and handling results. In order to achieve this, the architecture must be able to compare the capabilities of different methods, invoke them with different parameters, aggregate their results, and pass them to the next stage of the workflow. The fusion ontology defines knowledge structures supporting these functionalities.

The fusion ontology provides two kinds of knowledge structures:

- *Descriptors of problem-solving methods, tasks and application contexts.* This information is used to select and configure methods. Here, the *task* corresponds to a particular stage of the fusion workflow (e.g., coreference resolution). The descriptor of a *method* specifies the task it tackles, the conditions in which it can be applied and the default configuration parameters. The *application context* represents a configuration of a method adjusted for a specific domain. We will discuss these structures in sections 3.6.1, 3.6.2, and 3.6.3 respectively.

- *Auxiliary knowledge structures.* These structures represent meta-level descriptors of the methods' inputs and outputs (e.g., mappings between individuals, sets of conflicting statements). These structures are primarily used to pass methods' results to the following stages of the workflow. We will discuss these in section 3.6.4.

Table 3.1: Task descriptor example

Task	Coreference resolution
Inputs	*SourceKnowledgeBase* :type *KnowledgeBase*; *TargetKnowledgeBase* :type *KnowledgeBase*;
Outputs	*MergeSets* :type list of *MergeSet* - Set of possible mappings between instances of source and target knowledge bases

3.6.1 Tasks

Tasks represent the required capabilities of methods, in particular, the types of inputs and outputs. Each method subscribed to a task has to satisfy the requirements defined in the task description. For example, Table 3.1 shows the descriptor for the coreference resolution task. It specifies that each coreference resolution method should take as input two knowledge bases (row 2) and return a set of mappings between individuals which the method considers identical (row 3).

Some of the tasks can be further decomposed so that each subtask is processed by its own methods. Such complex tasks are tackled using special task decomposition problem-solving methods. These do not refer to any implementation but instead define a sequence of subtasks for a task they handle. The high-level task decomposition is pre-defined in the ontology as shown in Fig. 3.1. However, this task hierarchy can be further expanded by defining task decomposition methods handling tasks at the bottom-level of the hierarchy. Alternative decompositions are possible too: in that case several task decomposition methods will be defined to handle one complex task. At any level of the hierarchy, it is possible that not all tasks need to be handled. For instance, the feature selection subtask of the coreferencing task is not needed in the case where a set of relevant attributes for a particular class is known. Such situations are handled by defining pre-conditions of tasks: if a pre-condition is not satisfied,

then the method selection for the task is not performed and the task is skipped.

3.6.2 Problem-solving methods

Problem-solving methods encapsulate the reasoning steps and the types of knowledge needed to perform a task [42]. As already said, each subtask can be performed by different methods, both generic and domain-dependent (e.g., using key attributes or machine-learning models for coreferencing, hand-tailored rules or formal ontology diagnosis for conflict detection). Existing approaches to building knowledge-based systems distinguish between task decomposition problem-solving methods, which decompose a task into subtasks, and primitive problem-solving methods (or inferences) that perform actual reasoning steps [42, 83, 104]. In our architecture the task decomposition is represented as a hierarchy of tasks (see section 3.6.1).

While the internal reasoning of primitive problem-solving methods is hidden from the architecture, a method descriptor has to provide information needed by the architecture to perform method selection. This information includes the competence of a method (the task it handles and additional constraints restricting its input), the link to its implementation, and the quality of its output (the degree of confidence that the output provided by the method is correct). The inputs and outputs of a method are defined by the task it tackles. Additional restrictions are specified by a selection criterion, which represents the most generic application domain of a method. Since our architecture assumes that the data are represented in RDF, we use a SPARQL query to define the selection criterion. The reliability measure of a problem-solving method represents a degree of confidence that a decision made by the method is correct. It is expressed as a number between 0 (indicating that the output of the algorithm

Table 3.2: Coreference resolution method descriptor example

Method	Label-based Jaro-Winkler matcher
Tackles	Coreference resolution
Selection criterion	SELECT ?uri WHERE { ?uri rdfs:label ?label }
Reliability	0.9
Description	A generic method, which performs matching based on the label similarity measured using Jaro-Winkler metrics.
Implementation	org.xmedia.fusion.objectidentification. JaroWinklerSimMetricsObjectIdentificationMethod
Parameters	
Threshold	0.87
Attributes	*rdfs:label*

does not give any information about the correct answer) and 1 (indicating that the maximal likelihood of a correct decision). The exact interpretation of this value (e.g., as a probability measure or Dempster-Shafer belief measure [106]) is not defined at the level of architecture: it is only used to indicate relative preference. However, depending on the actual methods used, this measure can have precise formal interpretation: e.g., our dependency resolution algorithm described in Chapter 5 operates under the assumption that the reliability value represents a positive belief measure in the Dempster-Shafer theory. Finally, a method descriptor specifies additional parameters required by the method and their default values. An example of such a descriptor of the Jaro-Winkler string-similarity coreferencing method is presented in Table 3.2. First, the descriptor specifies that the method tackles the *coreference resolution* task (row 2). It means that the method takes as its input two knowledge bases and returns a set of mappings between individuals as its output, as was defined in the corresponding task descriptor (Table 3.1). Then, the descriptor contains the most general selection criterion for the method (row 3): a SPARQL query that selects individuals which the method can process. The architecture runs this query

during the method selection stage, and the method is selected for invocation only if the query returns results. In our example, the method is applicable to any individual which has a value for the *rdfs:label* property. The reliability of the method (row 4) denotes the estimated quality of method's output. This score is interpreted as the Dempster-Shafer belief mass [106] assigned to each method's output (in this example, to each coreference mapping produced by the method). The method is invoked using the reference to the method implementation (row 6). Our architecture assumes this to be the name of a Java class available on the system's class path.

Apart from these standard parameters, the method descriptor also contains method-specific parameters. These parameters are not pre-defined in the fusion ontology and are defined by the creator of the method. In our example, the method decides whether two individuals refer to the same entity by measuring average Jaro-Winkler similarity between their attributes. In the most general case it measures the similarity between the labels of individuals (row 9) and compares it with the threshold 0.87 (row 8). It accepts that two individuals are the same if the Jaro-Winkler similarity value between their labels is higher than the threshold.

3.6.3 Application contexts

While some methods (e.g., rule-based ones) are hand-tailored for a specific domain, other methods can be applied to different types of data. However, reusing a method in a different domain might require reconfiguring it. The concept of *application context* represents such a link between the problem-solving method and the domain (similar to the *PSM-Domain* concept in [42] and the *Application* concept in [83]). Application contexts specify the parameters of a method when applied to a specific

Table 3.3: Application context example

Method	Label-based Jaro-Winkler matcher
Selection criterion	SELECT ?uri WHERE { ?uri rdf:type opus:Publication . ?uri rdfs:label ?label . }
Linked to class	*opus:Publication*
Reliability	0.95
Parameters	
Threshold	0.93
Attributes	*rdfs:label*, *opus:year*

domain. These parameters override the default values defined in the method descriptor. Domains are interpreted as class definitions in the domain ontology and described with SPARQL queries (further narrowing the generic applicability range defined in the method descriptor). An example of an application context for the Jaro-Winkler string similarity method applied to individuals of the class *opus:Publication* describing scientific publications in the SWETO-DBLP ontology is given in Table 3.3. In this example the context is related to the label-based Jaro-Winkler matcher described in Table 3.2. However, while the method is applicable to any individual with the *rdfs:label* property, the application context is only relevant for the individuals of the class *opus:Publication* (row 3). The context specifies that the method produces, on average, more reliable mappings for this subset of individuals (row 4) than for the whole set of individuals to which the method can be applied (row 4 in Table 3.2). For *opus:Publication* individuals our example method compares not only labels but also publication years (row 7) and requires that the average Jaro-Winkler similarity for both pairs of attributes exceeds the threshold 0.93 (row 6). In this way, application context objects allow methods to be fine-tuned to different parts of the datasets and address the granularity problem discussed in section 2.6.

3.6.4 Auxiliary structures

Auxiliary structures represent results produced by methods solving intermediate tasks in the workflow. They are used as inputs to other methods together with the factual data contained in the source and the target knowledge bases. They usually contain references to objects from the source and target knowledge bases but are not copied into the main knowledge base. The structures defined in the ontology include:

- *Object context models.* Existing coreferencing algorithms determine whether two individuals represent the same entity based on their *contexts* - sets of relevant attributes. An object context model represents such a set of attributes. The decision models of some coreferencing algorithms (e.g., weighted string similarity) are independent from actual sets of attributes. Also special algorithms for attribute selection (such as information gain or relief [55]) do not require knowledge about the internal reasoning of coreferencing algorithms.

- *Atomic mappings.* These are candidate mappings produced by coreference resolution algorithms. Each atomic mapping contains references to two individuals believed to be identical and the attached belief estimation (a number between 0 and 1).

- *Coreference clusters.* These are sets of multiple individuals believed to be identical, which are created by the architecture as a post-processing of the coreferencing task. Coreference clusters may aggregate several atomic mappings.

- *Dependency sets.* Sets of coreference mappings and data statements may be interdependent with respect to available domain knowledge. Special cases of dependencies are *inconsistencies* occurring in cases, where several statements

violate a restriction, or *functionality dependencies*, where a mapping between a pair of individuals implies a mapping between a pair of related individuals.

- *Training examples.* As pointed out in section 2.2.1, many existing algorithms view coreference resolution as a special case of classification or clustering and use machine-learning techniques to train decision models. Training examples contain available training data for such algorithms. Like an atomic mapping, each training example contains references to two individuals believed to be identical, however, this mapping is accepted as correct. Training examples are stored permanently by the architecture in a separate repository.

With regard to the life-cycle there are two kinds of auxiliary structures: temporary and permanent. Temporary structures are application-dependent, describe the actual data being fused, and are relevant only during a single fusion session. Such structures include atomic mappings, coreference clusters and dependency sets. Temporary instances are deleted after the fusion session is completed. Permanent structures are application-independent and can be relevant to different fusion sessions. However, they can be domain-dependent and contain references to schema-level entities of a domain ontology. Permanent structures include, for instance, object context models and training examples.

3.6.5 Fusion ontology: summary

The fusion ontology represents a fundamental component of the KnoFuss architecture, which allows the system to satisfy the requirements specified in section 3.2 and, consequently, overcome the limitations outlined in section 2.6. Relations between

domain knowledge and problem-solving knowledge are specified at the level of the respective ontologies, which allows them to be easily redefined for a new application scenario and helps to achieve reusability. Application context structures allow methods to be applied with different configuration settings depending on the type of data, which addresses the granularity requirement. In the same way, while method descriptors specify the most generic applicability conditions for methods, application contexts allow these conditions to be narrowed for a specific domain. In this way, the method-domain adjustment requirement is realised. Uncertainty degrees are represented formally via methods' reliability weights and then can be assigned to the methods' output structures such as atomic mappings. In this way, the provenance of the methods' decisions is preserved and can be utilised at the later stages of the workflow. This allows the uncertainty management and dependency processing requirements to be satisfied.

Instantiations of the fusion ontology concepts represent the configuration of the KnoFuss architecture for a specific use case scenario. The architecture utilises this information during its task handling workflow.

3.7 Task handling workflow

The system starts each atomic task by selecting appropriate methods. Method selection is performed in two phases. First, the system pre-selects all methods which can potentially be applied to a domain with the help of *method descriptor* objects defined by the fusion ontology.

All applicable methods are identified by running the SPARQL queries specified as

selection criteria. A method descriptor defines the most general conditions, in which the method can be applied, together with the default configuration parameters. In the example case in Table 3.2 these include just the threshold and the set of relevant attributes; other algorithms may involve more complex decision models.

After the set of applicable methods is selected, the system tries to determine the optimal parameters of the algorithm given the data to which it is applied. It is often the case that the same method can be applied to a wide range of data instances: for example, string similarity coreferencing algorithms are applicable to any individual, which have string properties. However, the performance of an algorithm and its optimal settings may differ when it is applied to individuals of different classes. For instance, when the Jaro-Winkler string similarity algorithm is used to find identical scientific papers, it must have a higher threshold than when it is applied to disambiguate the authors of the papers. Paper titles have generally longer string length and more consistent format, while people's names allow initial abbreviations and titles (like Dr., Prof.), which require the algorithm to be more "tolerant". Thus, as was argued in section 3.2, a more fine-grained method configuration is needed. Application contexts can be organised hierarchically (Fig. 3.2) following the taxonomy defined by the domain ontology. Thus, there can be, for instance, a configuration for journal articles, which use more specific features than the generic publication coreferencing. By default each application context corresponds to one class in the domain ontology.

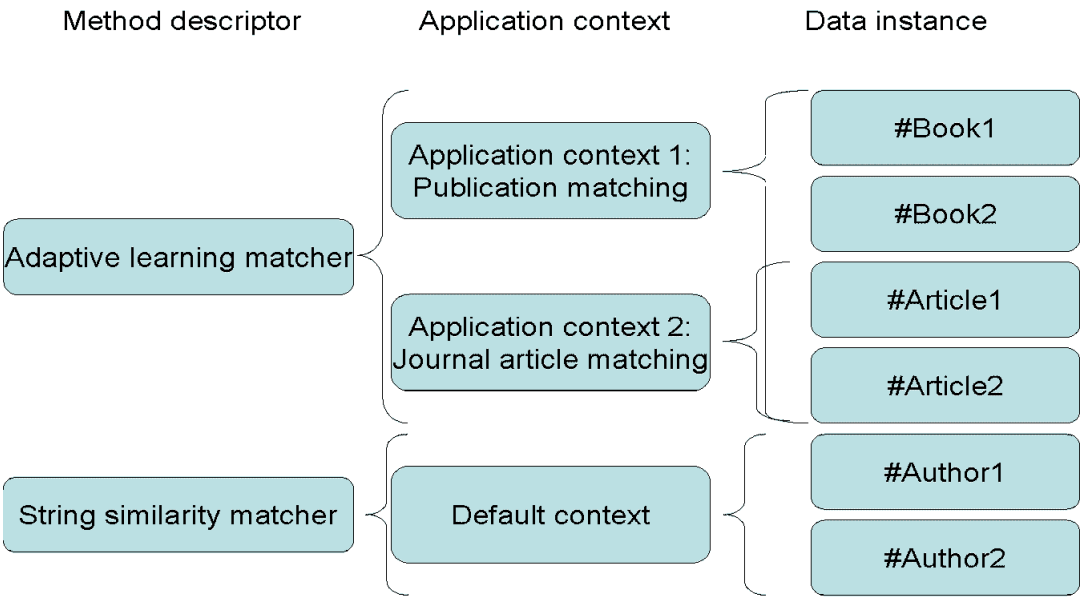

Figure 3.2: Method selection using hierarchical application contexts

3.7.1 Method invocation and handling results

After the system has selected applicable methods and picked the best configuration parameters known for them, it proceeds with method invocation. All methods from the selected set are invoked in sequence, and their results, structured according to the fusion ontology, are added to the source knowledge base. In cases where the results of methods conflict, those produced by the most reliable method are retained. An object representing the result of a method preserves a reference to the method descriptor. Thus, the reliability of the method's output is considered to be the same as the reliability of the method itself in the context in which it was invoked. Then the source knowledge base together with the accumulated intermediate information is passed to the next stage.

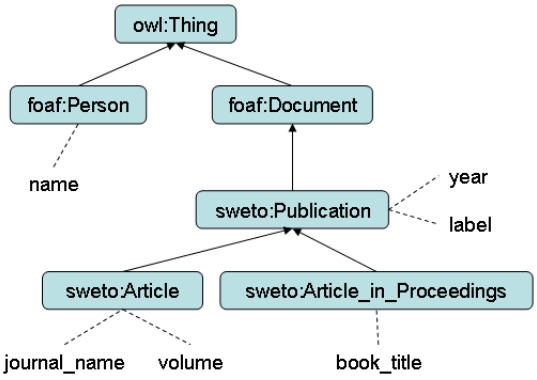

Figure 3.3: Class hierarchy in the SWETO-DBLP ontology

3.7.2 Example

To illustrate, let us consider a simple scenario handling the *Object identification* stage when fusing data from the scientific publication domain structured according to the SWETO-DBLP ontology[2]. This extends the FOAF ontology[3] describing basic data about people and defines additional classes and properties to describe publications, as shown in Fig. 3.3. In this example, we assume that we need to fuse the datasets shown in tables 3.4 and 3.5.

The method library contains the following two methods:

- The rule-based matcher: this matcher considers two instances as equivalent if they have exactly equal values for their key properties. Its parameter is the list of key properties for a class of individuals.

- The Jaro-Winkler matcher: a fuzzy string similarity matcher, which compares property values of two instances using the Jaro-Winkler distance metrics. If more than one property is considered relevant, the result is calculated as an average over all properties. If the resulting similarity is above the threshold,

[2]http://lsdis.cs.uga.edu/projects/semdis/swetodblp/august2007/opus_august2007.rdf
[3]http://xmlns.com/foaf/spec/

Table 3.4: Source dataset for the example scenario

Local name	Type	Properties	
		Name	Value
Person1	foaf:Person	*foaf:name* *foaf:mbox*	Enrico Motta e.motta@open.ac.uk
Person2	foaf:Person	*foaf:name*	Dnyanesh Rajpathak
Pub1	opus:Article	*rdfs:label* *opus:year* *opus:journal_name* *opus:volume* *opus:number* *opus:author*	A Generic Library of Problem Solving Methods for Scheduling Applications 2006 IEEE Transactions on Knowledge and Data Engineering 18 6 Person2 Person1 ...
Pub2	opus:Article_in_Proceedings	*rdfs:label* *opus:year* *opus:book_title* *opus:author*	A Generic Library of Problem Solving Methods for Scheduling Applications 2003 K-CAP Person2 Person1 ...
Pub3	opus:Book	*rdfs:label* *opus:year* *opus:isbn*	Reusable Components for Knowledge Modelling 1999 1-58603-003-5

Table 3.5: Target dataset for the example scenario

Local name	Type	Properties	
		Name	Value
PersonA	foaf:Person	*foaf:name* *foaf:mbox*	Prof. Enrico Motta e.motta@open.ac.uk
PersonB	foaf:Person	*foaf:name*	Dnyanesh Rajpathak
PubA	opus:Article	*rdfs:label* *opus:year* *opus:journal_name* *opus:volume* *opus:number* *opus:author*	A Generic Library of Problem Solving Methods for Scheduling Applications 2006 IEEE Trans. Knowl. Data Eng. 18 6 Person2 Person1 ...
PubB	opus:Article_in_Proceedings	*rdfs:label* *opus:year* *opus:book_title* *opus:author*	A Generic Library of Problem Solving Methods for Scheduling Applications 2003 K-CAP 2003 Person2 Person1
PubC	opus:Book	*rdfs:label* *opus:year* *opus:isbn*	OCML: Reusable Components for Knowledge Modeling: Case Studies in Parametric Design Problem Solving 1999 1-58603-003-5

two individuals are considered equivalent.

All individuals of the class *foaf:Person* have the property *foaf:name*, which contains the full name of a person. However, the full name can be written in different ways and may be not unique (two people may have the same name), thus, it can only be used for fuzzy matching, which reduces precision. In contrast, the *foaf:mbox* property, describing a personal e-mail address, can be used as a key property. However, it does not guarantee good recall: the same person may be described in different datasets with different e-mails and in many cases the e-mail address is not known.

There is a similar case with individuals of the type *opus:Publication*: there is a key property *opus:isbn*, which is not always available. Alternatively, a publication can be identified by the set of its other properties, such as the title (*opus:title*), the publication year (*opus:year*), and the venue (*opus:venue*).

The application contexts for the methods can be defined as shown in Table 3.6. At the method selection step both methods are pre-selected as applicable. The rule-based matcher is applicable because one *Person* individual in the source knowledge base (Person1) has a *foaf:mbox* specified and one of the publications (Pub3) is a book with the ISBN. Both string similarity matchers can be used because *foaf:name* and *rdfs:label* properties are instantiated. However, their configuration settings and reliability will be different. Thus, at the invocation step the rule-based matcher method is invoked twice: once applying application context AC1 for the class *Person* and once for the class *Publication* (application context AC2). The Jaro-Winkler string similarity method is invoked four times, because each application context is relevant. Application context AC3 is used for the individuals Person1 and Person2; AC5 is used for Pub1; AC6 is relevant for Pub2. For Pub3, since there is no context

Table 3.6: Application contexts for the example scenario

No	Descriptor	
AC1	Rule-based matcher - Person	
	Method	Rule-based matcher
	Selection criterion	SELECT ?uri WHERE { ?uri rdf:type foaf:Person . ?uri foaf:mbox ?mbox . }
	Linked to class	*foaf:Person*
	Reliability	1.0
	Parameters	
	Attributes	*foaf:mbox*

AC2	Rule-based matcher - Publication	
	Method	Rule-based matcher
	Selection criterion	SELECT ?uri WHERE { ?uri rdf:type opus:Publication . ?uri opus:isbn ?isbn . }
	Linked to class	*opus:Publication*
	Reliability	1.0
	Parameters	
	Attributes	*opus:isbn*

AC3	Jaro-Winkler matcher - Person	
	Method	Jaro-Winkler matcher
	Selection criterion	SELECT ?uri WHERE { ?uri rdf:type foaf:Person . ?uri foaf:name ?name . }
	Linked to class	*foaf:Person*
	Reliability	0.90
	Parameters	
	Threshold	0.98
	Attributes	*foaf:name*

AC4	Jaro-Winkler matcher - Publication	
	Method	Jaro-Winkler matcher
	Selection criterion	SELECT ?uri WHERE { ?uri rdf:type opus:Publication . ?uri rdfs:label ?label . }
	Linked to class	*opus:Publication*
	Reliability	0.90
	Parameters	
	Threshold	0.97
	Attributes	*rdfs:label*, *opus:year*

AC5	Jaro-Winkler matcher - Article	
	Method	Jaro-Winkler matcher
	Selection criterion	SELECT ?uri WHERE { ?uri rdf:type opus:Article . ?uri rdfs:label ?label . }
	Linked to class	*opus:Article*
	Reliability	0.98
	Parameters	
	Threshold	0.90
	Attributes	*rdfs:label*, *opus:year*, *opus:journal_name*, *opus:volume*, *opus:number*

AC6	Jaro-Winkler matcher - Article_in_Proceedings	
	Method	Jaro-Winkler matcher
	Selection criterion	SELECT ?uri WHERE { ?uri rdf:type opus:Article_in_Proceedings . ?uri rdfs:label ?label . }
	Linked to class	*opus:Article_in_Proceedings*
	Reliability	0.95
	Parameters	
	Threshold	0.92
	Attributes	*rdfs:label*, *opus:year*, *opus:book_title*

Table 3.7: Method invocation results in the example scenario

Method	Context	Output
Rule-based matcher	Person	Person1≡PersonA
	Publication	Pub3≡PubC
Jaro-Winkler	Person	Person2≡PersonB
	Article	Pub1≡PubA
	Article_in_Proceedings	Pub2≡PubB
	Publication	Pub1≡PubB
	Publication	Pub2≡PubA

defined for the class *Book*, the generic context for the class *Publication* (AC4) is used instead. The results obtained after all the methods are invoked are shown in Table 3.7.

The rule-based matcher produced two mappings as its result. The mapping Person1 ≡ PersonA was produced because these individuals had the same value "e.motta@open.ac.uk" for the key property *foaf:mbox*. The mapping Pub3 ≡ PubC was obtained because of identical values for the *opus:isbn* property. The Jaro-Winkler matcher returned one mapping between *foaf:Person* individuals (Person2 ≡ PersonB) which had equivalent names ('Dnyanesh Rajpathak') and four mappings between *opus:Publication* individuals with equivalent titles. The results produced by the different methods complement each other: identical individuals recognised by their primary keys could not be discovered by the Jaro-Winkler matcher because their string comparison scores were below their respective thresholds. However, the Jaro-Winkler method introduced two erroneous mappings as well: Pub1 ≡ PubB and Pub2 ≡ PubA. Given that individuals merged in this way belong to two disjoint classes (*Article* and *Article_in_Proceedings*), these erroneous mappings will lead to an inconsistent knowledge base. Correcting such mistakes will be discussed in Chapter 5.

3.8 Using the class hierarchy to learn optimal method parameters

Context-dependent method configuration allows fine-grained tuning of methods for specific types of data. However, it also requires more effort to assign optimal parameters for all categories of data in the dataset. Assigning them manually is a task requiring significant user effort, especially in cases where the domain ontology contains individuals of many different classes. Moreover, it is possible that individuals of several classes share enough common features to the extent that optimal parameters for them are the same. Thus, the architecture has to include a mechanism for assigning configuration parameters in an optimal way in order to reduce manual user effort and maximise the reusability of the system. In order to achieve reusability, the generated application contexts must cover the widest possible domain while at the same time specify the best possible configuration settings for all subsets of the covered domain.

A common approach to determine optimal parameters of an algorithm automatically involves using machine-learning techniques (e.g., see `APFEL`[35] and `eTuner` [70] for the schema matching task). Machine learning can be used to generate optimal parameters for a method applied to a specific class of individuals and to estimate its reliability. But, in order to train a method to match individuals of a certain class, sufficient training examples are needed. Obtaining these for each ontological class is not always feasible.

In order to maximise the reuse of available training data, the KnoFuss architecture allows the ontological class hierarchy to be exploited for finding optimal parameters

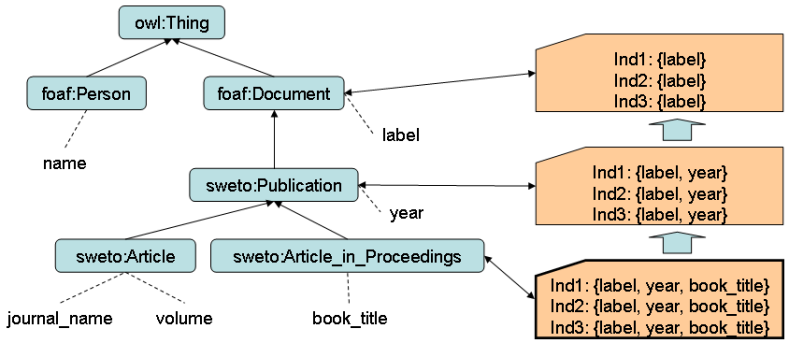

Figure 3.4: Reusing training examples to train a generic model for a superclass.

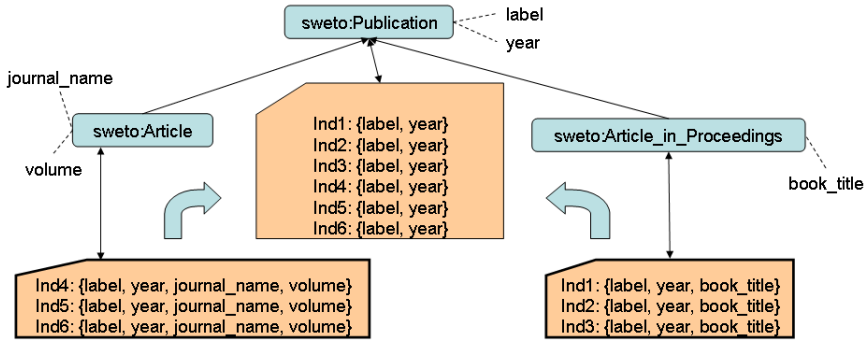

Figure 3.5: Combining training examples to train a model for a superclass.

as well as for selecting appropriate methods. This is especially relevant for the *coreference resolution* stage.

Ontological schemata can be exploited in two ways to manage a limited set of training data and assist the learning mechanism:

- Training examples belonging to a subclass can be used to learn a generic decision model for its superclass (Fig. 3.4).

- Training examples belonging to different subclasses of the same superclass can be combined together (Fig. 3.5).

Let's assume that in the ontology O we have a class C and its subclasses $C_1 ... C_n$:

$$\forall C_i.(C_i \sqsubseteq C)|i = 1...n$$

For each class a subset of individuals D_i is available. In other words, if we have an interpretation $\mathscr{I}$ of the ontology O, then we have:

$$\forall D_i.(D_i \subseteq C_i^{\mathscr{I}})|i = 1...n$$

For some individuals in D_i we also know correct identity relations:

$$T_i \subseteq D_i|i = 1...n$$

Pairs of these individuals constitute training sets S_i:

$$S_i \doteq (T_i \times T_i)|i = 1...n$$

where pairs of coreferent individuals serve as positive examples and pairs of non-coreferent individuals constitute negative examples. Let f_i represent a set of potentially relevant attributes for each class C_i (e.g., this may include all properties within a certain range r). The learning algorithm takes as input a set of training examples S and relevant attributes f and produces a decision model $h_i(S_i, f_i)$:

$$h_i(S_i, f_i) : (x; y) \rightarrow P(x \equiv y)$$

where x and y belong to the class C_i and $P(x \equiv y)$ is a probability that the individuals x and y are equivalent.

Now, suppose, we only have training instances for a subset of classes $C_1...C_n$, i.e.,

$$\forall S_j.(|S_j| > 0)|j = 1...m, m < n$$

$$\forall S_k.(|S_k| = 0)|k = m + 1...n$$

It means that it will not be possible to generate decision models for a range of subclasses $C_k \sqsubseteq C|k = m + 1...n$, because of the lack of training examples.

During the configuration phase we train the learning algorithm to produce $m+1$ decision models: one model $h_j(S_j, f_j)$ for each C_j where $j \leq m$ and one model for the superclass C:

$$h_{overall}(\bigcup_{j=1}^{m} S_j, \bigcap_{i=1}^{n} f_i)$$

The learning algorithm for the superclass C will take as input the union of all training sets $S = \bigcup_{j=1}^{m} S_j$. The set of relevant features will only contain the features of the class C: $f = \bigcap_{i=1}^{n} f_i$. Obtained decision models are linked to relevant application context descriptors.

Thus, during the execution phase the architecture will be able to select the decision model for instances of a specific class in the following way:

- Use either $h_{overall}$ or h_j for individuals of C_j where $j = 1...m$
- Use $h_{overall}$ for individuals of C_k where $k = m...n$

The dilemma in the first case for each class C_j concerns the choice between the model $h_{overall}$ and the model h_j. In the general case, the model $h_{overall}$ is produced having more training examples (because $|\bigcup_{j=1}^{m} S_j| \geq |S_j|$) but less features (because $|\bigcap_{i=1}^{n} f_i| \leq |f_i|$).

This choice can be made based on the estimated accuracy of each decision model. In the case of coreference resolution task, the accuracy can be measured using the standard F-measure which combines precision p and recall r ($F = \frac{(1+\alpha)pr}{\alpha p + r}$). Each learned model is evaluated on a set of test examples. The algorithm is included into the library of matching methods and each learned model is described as a separate application context. The reliability of the algorithm in each context is assigned according to the achieved accuracy on the test set. In the case where two models

$h_{overall}$ and h_j were applied and produced conflicting results, the result produced by a model with a greater reliability is chosen.

This approach, however, can be used only under the assumption that each property is used with the same meaning for all subclasses of its domain class. For example, a generic property *rdfs:label* can be used for individuals of any class (i.e., *owl:Thing*, if we consider an OWL knowledge base) but have different meaning in each case: person's full name, document's title or the name of a geographic object. Despite using the same property as the predicate, each of these attributes has its specific representation features and format. An algorithm trained for the most generic class (*owl:Thing* in OWL knowledge bases) using only training examples belonging to one subclass (e.g., people) is likely to suffer from overfitting: it may not be able to generalise and to identify correctly coreferent individuals belonging to classes not represented in the training data.

The experiments we performed to validate this procedure of using an ontological class hierarchy for method configuration are described in Chapter 6.

3.9 Summary

In this chapter we presented the design of the architecture we developed to handle the fusion process. The material presented in this chapter was published in [87].

The architecture allows several different methods to be combined for the same task and methods for different subtasks to be combined into a workflow, which was the goal specified in the research question 1: "How should algorithms performing fusion subtasks be used in combination to implement the semantic data fusion workflow?"

Combination of algorithms is used in many existing integration systems developed both in the database and the Semantic Web communities (see section 2.5), however these systems do not take into account specific requirements of the semantic data fusion task:

- Schema matching systems do not have the required granularity and do not allow different methods and different configuration parameters to be selected for subsets of the knowledge base being considered.

- Database integration systems do not take into account features of semantic data and axioms defined in domain ontologies

Our solution to the problem was an approach based on problem-solving methods. Different algorithms handling subtasks of the fusion process are represented as problem-solving methods and have their capabilities and configuration parameters described formally using the fusion ontology. The architecture selects appropriate methods and combines them into a workflow based on these descriptors.

The second research question considered was question 2: "How can we support the reusability of fusion algorithms across domains?" In order to maximise reusability of methods, and the system as a whole, application contexts are defined as adaptor structures linking a method and a domain. Thus, for the same method different configuration parameters, selection criteria and estimated reliability can be defined depending on the domain to which it is applied.

To enhance the process of assigning these parameters the domain ontology is used. Ontological subsumption relations are exploited to adjust method parameters depending on the type of data, to which a method is applied. Class hierarchy defined

in the domain ontology is used both for determining appropriate parameters of a method using training data and selecting an appropriate context for method invocation depending on input data. In this way ontological axioms are exploited to assist fusion algorithms, which was the focus of the question 3: "How can we exploit axioms defined in domain ontologies to improve the performance of fusion algorithms?"

Chapter 4

Inconsistency resolution using Dempster-Shafer belief propagation

This chapter describes our approach to the inconsistency resolution problem. Data to be fused are extracted from several sources with varying reliability. Data statements can contradict each other and render the fused knowledge base inconsistent. These inconsistencies need to be resolved. Our approach uses the Dempster-Shafer theory of evidence with description logic reasoning to localise and resolve inconsistencies. The chapter first presents a review of related work motivating the choice of the Dempster-Shafer theory of evidence as an uncertainty representation framework. Then, we discuss three stages of our inconsistency handling algorithm: (i) an inconsistency detection procedure using Reiter's hitting set tree algorithm, (ii) a set of rules translating an inconsistent OWL subontology into a belief propagation network, and (iii) a belief propagation procedure.

The material presented in this chapter contributed to a workshop paper which was later extended to a book chapter:

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2007). Using the Dempster-Shafer theory of evidence to resolve ABox inconsistencies. Workshop: Uncertainty Reasoning for the Semantic Web, 6th International Semantic Web Conference (ISWC 2007), Busan, Korea.

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2008). Using the Dempster-Shafer theory of evidence to resolve ABox inconsistencies. In: Uncertainty Reasoning for the Semantic Web I, Eds: Paulo C.G. da Costa, Claudia d'Amato, Nicola Fanizzi, Kathryn B. Laskey, Ken Laskey, Thomas Lukasiewicz, Matthias Nickles, Mike Pool.

4.1 Introduction

Automatic information extraction algorithms do not produce 100% correct output, which may lead to inconsistencies in the extracted knowledge base. Errors can also be introduced by human editors: e.g., a publication venue of a paper given on an author's web-site can be mistyped. Information extracted from different sources can be genuinely contradictory. Finally, when information from different sources is fused together, the identity problem has to be resolved: individuals referring to the same real-world entities must be linked or merged. Automatic matching algorithms can also produce errors, which lead to knowledge base inconsistencies. Processing inconsistencies during fusion is important not only because these affect logical reasoning, but also because each inconsistency normally indicates either a possible error in the data or a divergence of views between information sources. For example, the popular DBLP[1] web-site maintaining information about research publications in the computer science domain, contains a record describing the following publication:

> Diana Maynard, Valentin Tablan, Kalina Bontcheva, Hamish Cunningham: Rapid customization of an information extraction system for a surprise language. ACM Trans. Asian Lang. Inf. Process. 2(3): 295-300 (2003).

This paper was produced within the context of the AKT project[2]. However, the web-site of the AKT project itself gives the following information about the paper:

> Maynard, Dr Diana and Tablan, Mr Valentin and Bontcheva, Dr Kalina and Cunningham, Dr Hamish (2004) Rapid customization of an Information Extraction system for surprise languages. Special issue of ACM Transactions on Asian Language Information Processing.

If the publication year is defined as a functional property in the ontology, then fusing these two records will lead to an inconsistent knowledge base. Different publication

[1] http://dblp.uni-trier.de

[2] http://www.aktors.org/akt/

years may indicate that either one of the sites provides incorrect information or even that there are two papers with the same title publishing in different years. Knowing how the data was constructed by both web-sites is helpful for resolving this conflict: in the case of DBLP information is gathered from the original electronic records of publication venues (journals and proceedings), while the authors manually entered data into the AKT web-site. The possibility of a human error such as mistype in the second case makes AKT data less reliable. An optimal inconsistency resolution decision in the fused knowledge base will be to remove the value from the AKT web-site (2004). Because both data and fusion methods' decisions can be imperfect, the fusion process has to deal with uncertain information. In order to analyse the causes of a conflict and propose a solution (in a simple variant, a ranking of conflicting statements, or a set of statements which should be considered "wrong"), it is important to measure this uncertainty and reason about it. There are different ways to obtain the necessary uncertainty measurements, for example:

- Extraction algorithms can often estimate the reliability of their output by attaching confidence values to the generated statements [59] (an example of a large-scale dataset providing these measures is YAGO [3]).

- Degrees of trust can be assigned to different information sources based on experts' evaluations.

- The quality of the output of automatic matching algorithms can be measured using precision/recall measures over some gold standard evaluation set.

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

Each case of inconsistency occurring during fusion can be caused by several uncertain factors concerning both the data statements themselves and/or fusion decisions. Hence, techniques are needed both to represent these uncertainty values in a consistent quantitative way, and also to reason about these factors in an integrated way. There are several existing formalisms for uncertainty reasoning which propose solutions to these problems.

In particular, most of the ongoing research in the field of applying uncertainty reasoning to the Semantic Web focuses on fuzzy logic and probabilistic approaches [24]. Of the defective aspects of information listed in section 2.1 (*ambiguity*, *uncertainty*, *imprecision*, *vagueness*, *inconsistency*), fuzzy logic was designed to deal with the representation of *vagueness* and *imprecision*. In our data fusion scenario we are dealing with knowledge bases structured using standard OWL ontologies: i.e., the classes and properties are not vague. The main issue is the one of uncertainty, where we need to assess the likelihood that a statement is true or false. The probabilistic approach is more appropriate for dealing with such problems, however, it still has a limitation. As stated in [56], axioms of the probability theory are implied by seven properties of belief measures. One of them is *completeness*, which states that "a degree of belief can be assigned to any well-defined proposition". However, this property cannot be ensured when dealing with confidence degrees assigned by extractors, because they do not always carry information about the probability of a statement being *false*. The Dempster-Shafer theory of evidence [106] presents a formalism that helps to overcome this problem. It allows belief measurements to be assigned to sets of propositions, thus explicitly specifying degrees of ignorance. In order to apply this formalism to the task of semantic data fusion we need to cover the gap between

the logic reasoning over OWL axioms, which operates with description logic knowledge bases, and the uncertainty reasoning, which operates with belief propagation networks. Our approach includes three components:

1. An inconsistency detection and localisation procedure, which uses crisp logical reasoning and the diagnosis algorithm described in [96].

2. An approach to uncertainty reasoning using valuation networks [109].

3. A set of translation rules for building valuation networks from OWL-DL sub-ontologies, which we consider our primary contribution.

In the following sections we describe the algorithm for resolving conflicts using the Dempster-Shafer belief propagation approach in more detail.

4.2 Related work

There are several studies dealing with inconsistency handling in OWL ontologies, among others [53] and [64]. The general algorithm for the task of repairing inconsistent ontologies consists of two steps:

- *Ontology diagnosis*: finding sets of axioms, which contribute to inconsistency;

- *Repairing inconsistencies*: changing/removing the axioms most likely to be erroneous.

Choosing the relevant axioms for change and removal is a non-trivial task. Existing algorithms working with crisp ontologies (e.g., [64]) utilise criteria such as syntactic relevance (how often each entity is referenced in the ontology), impact (the impact

on the ontology from removing the axiom should be minimised), and provenance (preferring the removal of axioms originating from less reliable sources). The latter criterion is especially interesting for the automatic ontology population scenario since extraction algorithms do not extract information with 100% accuracy. A study described in [54] specifies an algorithm which utilises the confidence value assigned by the extraction algorithm. The strategy employed there is to order the axioms according to their confidence values and to add them incrementally, starting from the most certain one. If adding an axiom leads to an inconsistency, then a minimal inconsistent subontology is determined and the axiom with the lowest confidence is removed from it. A disadvantage of such a technique is that it does not take into account the impact of an axiom (i.e., when an axiom violates several restrictions, it has higher impact and so this factor should be taken into account during the ranking process). Moreover, it does not consider the importance of redundancy: if the same statement was extracted from several sources, this should increase its reliability. Using an established formalism for uncertainty representation and reasoning would provide a more sound approach to capture these factors, rank potentially erroneous statements, and resolve inconsistencies.

As we said earlier, in the Semantic Web domain the studies on uncertainty reasoning are mostly focused on two formalisms: probability theory and fuzzy logic. Existing implementations of fuzzy description logic [113, 114] are based on the notion of a fuzzy set representing a vague concept. The uncertainty value in this context denotes a membership function $\mu_F(x)$ which specifies the degree to which an object x belongs to a fuzzy class F. Probabilistic adaptations of OWL-DL include Bayes OWL [27] and PR-OWL [25]. However, as we discuss below, neither of these formalisms

fully reflects the properties of the problems we are dealing with in the fusion scenario.

A framework for choosing an appropriate uncertainty handling formalism was presented in [56]. The framework is based on the seven properties of belief measurements:

1. *Clarity*: Propositions, to which belief a degree of belief can be attached, are well-defined (non-vague).

2. *Scalar continuity*: In order to represent a degree of belief, a single real number is both necessary and sufficient.

3. *Completeness*: Any well-defined proposition can be assigned a degree of belief.

4. *Context dependency*: The belief assigned to a proposition can be influenced by the belief in other propositions.

5. *Hypothetical conditioning*: The belief in a conjunction of propositions can be calculated from the belief in one proposition and the belief in the other proposition using some function which assumes that the first proposition is true.

6. *Complementarity*: The belief in a proposition is a monotonically decreasing function of the belief in a negation of the proposition.

7. *Consistency*: The belief in propositions which have the same truth value must be equal.

As discussed in [56], it has been proven that accepting all seven properties logically necessitates the axioms of probability theory. Alternative formalisms allow some properties to be weakened. Fuzzy logic deals with the case in which the *clarity*

property does not hold, i.e., when concepts and relations are vague. In our fusion scenario we are dealing with OWL ontologies which contain crisp concepts and properties. Thus, at the data level the corresponding class and property instantiation statements can be either true or false: a confidence value attached to a type assertion *ClassA(Individual1)* denotes a degree of belief that the statement is true in the real world rather than the degree of inclusion of the entity *Individual1* into a fuzzy concept *ClassA*. A special case involves instance equivalence and distinction relations: *owl:sameAs* and *owl:differentFrom*. Vagueness can be present for such relations even if both subject and object individuals are described in terms of crisp ontologies. For example, in the knowledge base containing cooking recipes it is possible to have two recipes for the same dish which differ in the quantity of one ingredient. Such individuals can be seen as "equivalent to a certain degree", which would violate the *clarity* property. However, given that we are dealing with a common shared ontology, we assume that there exists a common individual identity criterion among the users of this ontology. This criterion may be represented in the ontology or exist as an implicit user agreement. In other words, we assume that, when given complete information about two individuals, it is possible to say with certainty whether these individuals represent the same entity or two distinct entities. With this assumption, we can state that the clarity property holds in our scenario and the fuzzy interpretation is inappropriate for our case.

Although a probabilistic interpretation of the extraction algorithm's confidence is more suitable, it may lead to a potential problem. If we interpret the confidence value c attached to a statement returned by an extraction algorithm as a Bayesian probability value p, we, at the same time, introduce a belief that the statement is false

with a probability $1 - p$. However, the confidence of an extraction algorithm reflects only the belief that the document supports the statement and does not itself reflect the probability of a statement being false in the real world. Moreover, while statistical extraction algorithms [73] are able to assign a degree of probability to each extracted statement, rule-based algorithms [19, 129] can only assign the same confidence value to all statements extracted by the same rule based on the rule's performance on some evaluation set. Any extraction produced by a rule with a low confidence value in this case will serve as negative evidence rather than simply lack of evidence. This issue becomes particularly important if the reliability of sources is included in the analysis: it is hard to assign the conditional probability of a statement being false given that the document supports it. It means that the *completeness* property does not always hold.

The Dempster-Shafer theory of evidence [106] allows weakening of the completeness property. Belief can be assigned to sets of alternative options rather than only to atomic elements. In the case of binary logic, it means that the degree of ignorance can be explicitly represented by assigning a non-zero belief to the set {true;false}. On the other hand, the theory still allows the Bayesian interpretation of confidence to be used, when it is appropriate (in this case the belief assigned to the set {true;false} is always set to 0). This chapter presents an algorithm for resolving inconsistencies by translating the inconsistency-preserving subset of an ontology into the Dempster-Shafer belief network and choosing the axioms to remove on the basis of their *plausibility* value.

Alternative approaches to uncertainty representation, which have not been applied so far to ontological modelling, include probability intervals [26] and higher-order

probability [45]. However, the first of these approaches uses *min* and *max* operators for aggregation, which makes it hard to represent cumulative evidence, while the second focuses on resolving different kinds of problems (namely expressing probability estimations of other probability estimations). There are also other approaches to belief fusion in the Semantic Web (e.g., [99] and [85]). These studies deal with the social issues associated with representing trust and provenance in a distributed knowledge base and focus on the problem of establishing the certainty of statements asserted by other people. These approaches, however, do not focus on resolving the inconsistencies and just deal with direct conflicts (i.e., statement A is true vs statement A is false). Moreover, they do not take into account ontological inference and mutual influence of statements in the knowledge base. Hence, they can be considered complementary rather than alternative to the approach described here.

4.3 The Dempster-Shafer belief theory

The Dempster-Shafer theory of evidence differs from Bayesian probability theory because it allows beliefs to be assigned not only to atomic elements but to sets of elements as well. The basis of Dempster's belief distribution is the *frame of discernment* (Ω) - an exhaustive set of mutually exclusive alternatives. A *belief distribution function* (also called *mass function* or *belief potential*), $m(A)$, represents the influence of a piece of evidence on subsets of Ω and has the following constraints:

- $m(\oslash) = 0$ and
- $\sum_{A \subseteq \Omega} m(A) = 1$

$m(A)$ defines the amount of belief assigned to the subset A. When $m(A) > 0$, A is referred to as a *focal element*. If each focal element A contains only a single element, the mass function is reduced to a probability distribution. Mass can also be assigned to the whole set Ω. This represents the uncertainty of the piece of evidence about which of the elements in Ω is true. In our case each mass function is defined on a set of variables $D = \{x_1, ..., x_n\}$ called the domain of m. Each variable is Boolean and represents an assertion in the knowledge base. For a single variable we can represent the degree of *support* $Sup(x) = m(\{true\})$ and the degree of *plausibility* $Pl(x) = m(\{true\}) + m(\{true; false\})$. *Plausibility* specifies how likely it is that the statement is not false and aggregates the degree of *support* with the degree of ignorance ($m(\{true; false\})$). Using *plausibility* it is possible to select from a set of statements the one to be removed.

4.4 Description of the algorithm

In this section we present our algorithm for handling data-level inconsistencies. The algorithm assumes (i) that the data structure is defined using an OWL-DL ontology, (ii) that the ontological schema is consistent and (iii) that belief scores are available for data statements.

The algorithm consists of four steps:

1. Inconsistency detection.

   At this stage a subontology is selected containing all axioms contributing to an inconsistency. This step is performed using Reiter's diagnosis algorithm [96].

2. Constructing a belief network.

At this stage the subontology found at the previous step is translated into a belief network. We developed a set of rules, which translate OWL-DL axioms into corresponding nodes of the belief network.

3. Assigning mass distributions.

   At this stage mass distribution functions are assigned to nodes. We assign initial belief distributions for nodes based on the restrictions imposed by the corresponding OWL-DL axioms.

4. Belief propagation.

   At this stage uncertainties are propagated through the network and the confidence degrees of ABox statements are updated. This propagation is performed using the axioms for belief propagation in valuation networks formulated in [109]

The updated confidence degrees can be used to make decisions about each case of inconsistency. There are several possible options: e.g., resolve the inconsistency by removing the statement with the lowest plausibility (or support), or preserve all conflicting statements but use the confidence degrees to provide their ranking to the user.

4.4.1 Illustrating example

In order to illustrate our algorithm, we develop an example from the banking domain. We consider an ontology describing credit card applications, which defines two disjoint classes of applicants: reliable and risky. In order to be reliable, an applicant

has to have UK citizenship and evidence that (s)he was never bankrupt in the past. For example, the TBox contains the following axioms:

T1: $RiskyApplicant \sqsubseteq CreditCardApplicant$

T2: $ReliableApplicant \sqsubseteq CreditCardApplicant$

T3: $RiskyApplicant \sqsubseteq \neg\ ReliableApplicant$

T4: $ReliableApplicant \equiv \exists\ wasBankrupt.False \sqcap \exists\ hasCitizenship.UK$

T5: $\top \sqsubseteq \leq 1\ wasBankrupt$ ($wasBankrupt$ is functional)

The ABox contains the following axioms (with attached confidence values):

A1: *RiskyApplicant(Ind1)*: 0.7

A2: *wasBankrupt(Ind1, False)*: 0.6

A3: *hasCitizenship(Ind1, UK)*: 0.4

A4: *wasBankrupt(Ind1, True)*: 0.5

As given, the ontology is inconsistent: the individual *Ind1* belongs to both classes *RiskyApplicant* and *ReliableApplicant*[4], which are disjoint, and the functional property *wasBankrupt* has two different values. If we choose to remove the axioms with the lowest confidence values, it will require removing A3 and A4. However, inconsistency can also be repaired by removing a single statement A2, and the difference between the confidence degrees of A2 and A4 is small. The fact that A2 leads to the violation of two ontological constraints should increase the likelihood that it is incorrect. In order to select between these options (A2 vs A3 and A4), we need to capture adequately the mutual impact of confidence degrees of conflicting statements and to decide whether the *plausibility* value of a combination A3+A4 is higher than the *plausibility* value of A2.

[4]The latter is inferred from A2, A3, and T4

4.4.2 Inconsistency detection

The task of the inconsistency detection step is to retrieve all *minimal inconsistent subontologies (MISO)* of the ontology and combine them. As defined in [53], an ontology O' is a minimal inconsistent subontology of an ontology O, if (i) $O' \subseteq O$ and O' is inconsistent and (ii) for all O'' such that $O'' \subset O' \subseteq O$, O'' is consistent.

A general algorithm for diagnosis was proposed by Reiter in [96]. The algorithm deals with diagnosing first-order logic systems. A *system* is described as

$$(SD, COMPONENTS, OBS)$$

where

- SD, the *system description*, is a set of first-order sentences;
- $COMPONENTS$, the *system components*, is a finite set of constants;
- OBS, an *observation* of the system, a set of first-order sentences describing the state of the system.

If the system is behaving incorrectly, it means that the observation OBS contradicts the desired system behaviour, which assumes that all components behave correctly ($AB(c_i)$ - unary predicate interpreted as "abnormal"):

$$SD \cup \{\neg AB(c_1), \cdots \neg AB(c_n)\} | c_i \in COMPONENTS$$

In other words, the following expression is inconsistent:

$$SD \cup \{\neg AB(c_1), \cdots \neg AB(c_n)\} \cup OBS$$

By definition [96], "a diagnosis for (SD, $COMPONENTS$, OBS) is a minimal set $\Delta \subseteq COMPONENTS$ such that

$SD \cup OBS \cup \{AB(c)|c \in \Delta\} \cup \{\neg AB(c)|c \in COMPONENTS - \Delta\}$ is consistent."

Thus, a diagnosis is the smallest set of such components that the assumption about their abnormal behaviour together with the assumption about the correct behaviour of other components will be consistent with both the system description and the observation. A diagnosis can be found by identifying *minimal conflict sets* within the system. A conflict set for (SD, $COMPONENTS$, OBS) is a set $\{c_1, \cdots c_k\}$ such that

$$SD \cup OBS \cup \{\neg AB(c_1), \cdots, \neg AB(c_k)\} \text{ is inconsistent.}$$

A conflict set is minimal if none of its subsets is a conflict set for (SD, $COMPONENTS$, OBS). A system may have more than one conflict set: either there are several components causing abnormal behaviour or a single one, which leads to several conflicts. Thus, to resolve all conflicts, a diagnosis needs to contain at least one component from each conflict set. Such a set containing elements from several other sets is called a *hitting set*. For a collection of sets C a hitting set for C is a set

$$H \subseteq \bigcup_{S \in C} S$$

such that

$$H \cap S \neq \oslash \text{ for each } S \in C.$$

Reiter's algorithm [96] is aimed at finding all diagnoses by building a hitting set tree. The nodes of the tree are labeled with conflict sets. A node n labeled with a non-empty set Σ has outgoing edges labeled with elements of the set $\sigma \in \Sigma$ connecting the node n with its successors n_σ. The label for n_σ is such a conflict set S that

$$S \cap H(n_\sigma) = \oslash,$$

where $H(n_\sigma)$ contains all edge labels on the path from the root node to n_σ. In other words, a node is labeled by a random conflict set, which remains in the system if the components from $H(n_\sigma)$ were removed from it. So, eventually, in the process of tree building, each conflict set becomes resolved (by removing one of its elements), and the system becomes consistent (leaf nodes are marked with empty sets). Each path from the tree root to one of its leaves represents a diagnosis for the system. The tree is built breadth-first and the algorithm defines several rules which control pruning of the tree and optimise the diagnosis process. Reiter's algorithm ensures that all diagnoses and all minimal conflict sets are found.

Considering a specific case of ABox conflicts in a DL knowledge base, OWL axioms play the role of components of the system. It is easy to see that MISOs, as defined above, represent minimal conflict sets for the knowledge base. Implementing the diagnosis procedure for a DL knowledge base requires using a specific reasoner capable of identifying a minimal conflict set in an inconsistent ontology. The OWL reasoner Pellet [64] is able to return an inconsistent subontology for the first encountered inconsistency in the ontology. To calculate all MISOs $O'_1, ..., O'_n$ in the ontology we employ Reiter's hitting set tree algorithm as described above.

After all conflict sets were identified, the next step involves constructing belief networks from each set. If there exist two subontologies O'_i and O'_j such that $O'_i \cap O'_j \neq \oslash$, then these two subontologies are replaced with $O' = O'_i \cup O'_j$. For our illustrating example, the conflict detection algorithm is able to identify two conflict sets in this ontology: the first, consisting of {T3, T4, A1, A2, A3} (individual *Ind1* belongs to two disjoint classes), and the second {T5, A2, A4} (individual *Ind1* has two instantiations of a functional property). The statement A2 belongs to both sets

and therefore the sets are merged.

4.4.3 Constructing belief networks

The networks for propagation of Dempster-Shafer belief functions (also called valuation networks) have been described in [109]. By definition the valuation network is an undirected graph represented as a 5-tuple:

$$\{\Psi, \{\Omega_X\}_{X\in\Psi}, \{\tau_1, ..., \tau_n\}, \downarrow, \otimes\}$$

where

- Ψ is a set of variables,
- $\{\Omega_X\}_{X\in\Psi}$ is a collection of state spaces,
- $\{\tau_1, ..., \tau_n\}$ is a collection of valuations (belief potentials of nodes),
- $\downarrow$ is a marginalisation operator,
- $\otimes$ is a combination operator (see subsection 4.4.5 for the description of operators).

In our case Ψ consists of ABox assertions, and every $\{\Omega_X\}_{X\in\Psi} = \{0; 1\}$ and $\{\tau_1, ..., \tau_n\}$ are created using the rules given below. The operators are used for propagation of beliefs and are described in the following subsections. The network contains two kinds of nodes: *variable nodes* corresponding to explicit or inferred ABox assertions and *valuation nodes* representing TBox axioms. Variable nodes contain only one variable, while valuation nodes contain several variables.

To construct a valuation network we developed a set of rules for translating an inconsistent subontology into a belief propagation network (Table 4.1). We consider this

set of rules to be a novel contribution. OWL-DL TBox axioms specify how the truth value of one statement influences the truth values of other statements. This influence can be either positive (if it is true that an individual belongs to a subclass then it is true that it belongs to a superclass as well) or negative (if an individual already has one value for a functional property, it cannot have a different one). Translation rules capture these mutual influences and represent them as a set of network nodes and links between them. Each rule has as its precondition the existence of a certain OWL-DL axiom and the existence of other nodes in the belief network. Each rule corresponds to one particular type of OWL-DL axioms: for instance, rule 2 corresponds to a property instantiation statement and rule 1 represents a functional property restriction. The output of each rule includes adding one or more new nodes into the network and adding links between nodes.

Rules 1 and 2 directly translate each ABox statement into a variable node. Other rules process TBox axioms and create two kinds of nodes: one valuation node to represent the TBox axiom and one or more variable nodes to represent inferred statements. Such rules only fire if the network already contains variable nodes for ABox axioms, which are necessary to make the inference. For example, a rule processing the class equivalence axiom (Rule 4) is interpreted as the following: 'if there is a node N_1 representing the type assertion $I \in X$ and an *owl:equivalentClass* axiom $X \equiv Y$, then:

- Create a node N_2 representing the assertion $I \in Y$;
- Create a node N_3 representing the axiom $X \sqsubseteq Y$;
- Create links between N_1 and N_3 and between N_3 and N_2.'

Table 4.1: Belief network construction rules

N	Pre-conditions	Nodes to create	Links to create
1	$I \in X$	$N_1 : I \in X$	
2	$R(I_1, I_2)$	$N_2 : R(I_1, I_2)$	
3	$N_1 : I \in X$, $X \sqsubseteq Y$	$N_2 : I \in Y$, $N_3 : X \sqsubseteq Y$	(N_1,N_3),(N_3,N_2)
4	$N_1 : I \in X$, $X \equiv Y$	$N_2 : I \in Y$, $N_3 : X \equiv Y$	(N_1,N_3),(N_3,N_2)
5	$N_1 : I \in X$, $X \sqsubseteq \neg Y$	$N_2 : I \in Y$, $N_3 : X \sqsubseteq \neg Y$	(N_1,N_3),(N_3,N_2)
6	$N_1 : I \in X$, $X \sqcap Y$	$N_2 : I \in X \sqcap Y$, $N_3 : X \sqcap Y$, $N_4 : I \in Y$	(N_1,N_3),(N_4,N_3), (N_3,N_2)
7	$N_1 : I \in X$, $X \sqcup Y$	$N_2 : I \in X \sqcup Y$, $N_3 : X \sqcup Y$, $N_4 : I \in Y$	(N_1,N_3),(N_4,N_3), (N_3,N_2)
8	$N_1 : I \in X$, $\neg X$	$N_2 : I \in \neg X$, $N_3 : \neg X$	(N_1,N_3),(N_3,N_2)
9	$\top \sqsubseteq \leq 1R$, $N_1 : R(I, o_1)$, $N_2 : R(I, o_2)$	$N_3 : \top \sqsubseteq \leq 1R$	(N_1,N_3),(N_2,N_3)
10	$\top \sqsubseteq \leq 1R^-$, $N_1 : R(I_2, I_1)$, $N_2 : R(I_3, I_1)$	$N_3 : \top \sqsubseteq \leq 1R^-$	(N_1,N_3),(N_2,N_3)
11	$R \equiv R^-$, $N_1 : R(I_1, I_2)$	$N_2 : R \equiv R^-$, $N_3 : R(I_2, I_1)$	(N_1,N_2),(N_2,N_3)
12	$R \equiv Q$, $N1 : R(I_1, I_2)$	$N_2 : R \equiv Q$,$N_3 : Q(I_1, I_2)$	(N_1,N_2),(N_2,N_3)
13	$R \sqsubseteq Q$,$N_1 : R(I_1, I_2)$	$N_2 : R \sqsubseteq Q$, $N_3 : Q(I_1, I_2)$	(N_1,N_2),(N_2,N_3)
14	$R \equiv Q^-$,$N1 : R(I_1, I_2)$	$N_2 : R \equiv Q^-$, $N_3 : Q(I_2, I_1)$	(N_1,N_2),(N_2,N_3)
15	$Trans(R)$, $N_1 : R(I_1, I_2)$, $N_2 : R(I_2, I_3)$	$N_3 : Trans(R)$, $N_4 : R(I_1, I_3)$	(N_1,N_3),(N_2,N_3), (N_3,N_4)
16	$\leq 1.R$, $N_1 : R(I_1, o_1)$, $N_2 : R(I_1, o_2)$	$N_3 :\leq 1.R$, $N_4 : I \in \leq 1.R$	(N_1,N_3),(N_2,N_3), (N_3,N_4)
17	$\geq 1.R$, $N_1 : R(I_1, o_1)$, $N_2 : R(I_1, o_2)$	$N_3 :\geq 1.R$, $N_4 : I \in \geq 1.R$	(N_1,N_3),(N_2,N_3)
18	$= 1.R$, $N_1 : R(I_1, o_1)$, $N_2 : R(I_1, o_2)$	$N_3 : I \in = 1.R$	(N_1,N_3),(N_2,N_3)
19	$\forall R.X$, $N_1 : R(I_1, I_2)$, $N_2 : I_2 \in X$	$N_3 : \forall R.X$, $N_4 : I_1 \in \forall R.X$	(N_1,N_3),(N_2,N_3), (N_3,N_4)
20	$\forall R.o_1$, $N_1 : R(I_1, o_1)$,	$N_2 : \forall R.o_1$, $N_3 : I_1 \in \forall R.o_1$	(N_1,N_2),(N_2,N_3)
21	$\forall R.o_1$, $N_1 : R(I_1, o_2)$,	$N_2 : \forall R.o_1$, $N_3 : I_1 \in \forall R.o_1$	(N_1,N_2),(N_2,N_3)
22	$\exists R.X$, $N_1 : R(I_1, I_2)$, $N_2 : I_2 \in X$	$N_3 : \exists R.X$, $N_4 : I_1 \in \exists R.X$	(N_1,N_3),(N_2,N_3), (N_3,N_4)
23	$\exists R.o_1$, $N_1 : R(I_1, o_1)$,	$N_2 : \exists R.o_1$, $N_3 : I_1 \in \exists R.o_1$	(N_1,N_2),(N_2,N_3)
24	$\exists R.\top \sqsubseteq X$, $N_1 : R(I_1, I_2)$, $N_2 : I_1 \in X$	$N_3 : \exists R.\top \sqsubseteq X$	(N_1,N_3),(N_2,N_3)
25	$\top \sqsubseteq \forall R.X$, $N_1 : R(I_1, I_2)$, $N_2 : I_2 \in X$	$N_3 : \top \sqsubseteq \forall R.X$	(N_1,N_3),(N_2,N_3)

If a rule requires creating a node, which already exists in the network, then the existing node is used.

In our example, we can apply the rules in the following order:

1. Rule 1 to the statement A1 to create the variable node N_1: (*RiskyApplicant(Ind1)*)

2. Rule 2 to the statements A2, A3, A4 to create the variable nodes N_2: *hasCitizenship(Ind1, UK)*, N_3: *wasBankrupt(Ind1, False)*, N_4: *wasBankrupt(Ind1, True)*

3. Rule 5 to the node N_1 and the axiom T3 to create the variable node N_5: *ReliableApplicant(Ind1)* and the valuation node N_6: $RiskyApplicant \sqsubseteq \neg\ ReliableApplicant$

4. Rule 12 to the node N_5 and the axiom T4 to create the valuation node N_7: $ReliableApplicant \equiv \exists\ wasBankrupt.False \sqcap \exists\ hasCitizenship.UK$ and the variable node N_8: $Ind1 \in \exists\ wasBankrupt.False \sqcap \exists\ hasCitizenship.UK$

5. Rule 23 to the node N_2 and the axiom T4 to create the valuation node N_9: $hasCitizenship.UK$ and the variable node N_{10}: $Ind1 \in hasCitizenship.UK$

6. Rule 23 to the node N_3 and the axiom T4 to create the valuation node N_{11}: $wasBankrupt.False$ and the variable node N_{12}: $Ind1 \in wasBankrupt.False$

7. Rule 6 to the node N_{10} and the axiom T4 to create the valuation node N_{13}: $\exists\ wasBankrupt.False \sqcap \exists\ hasCitizenship.UK$ and add connections (N_{10}, N_{13}) and (N_{12}, N_{13}) (already existing nodes are reused instead of creating new connections)

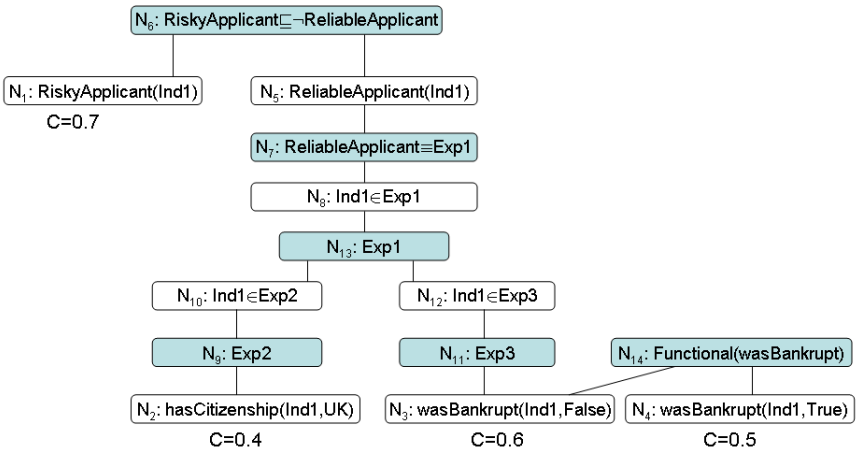

Figure 4.1: Belief network example (*Exp1*= ∃ *wasBankrupt.False*⊓∃ *hasCitizenship.UK*, *Exp2*=∃ *hasCitizenship.UK*, *Exp3*=∃ *wasBankrupt.False*)

8. Rule 9 to nodes N_3 and N_4 to create the valuation node N_{14}: *Functional (wasBankrupt)* and connections (N_3, N_{14}) and (N_4, N_{14})

The resulting valuation network is shown in Fig. 4.1

4.4.4 Assigning mass distributions

After the nodes are combined into the network, the next step is to assign the mass distribution functions to the nodes. There are two kinds of variable nodes: (i) nodes representing statements supported by the evidence and (ii) nodes representing inferred statements. Initial mass distribution for the nodes of the first type is assigned based on their extracted confidence. If a statement was extracted with a confidence degree c, it is assigned the following mass distribution: $m(True) = c, m(True; False) = 1 - c$. It is possible that the same statement is extracted from several sources. In this case, multiple pieces of evidence have to be combined using Dempster's rule of combination.

Nodes created artificially during network construction are only used for propagating beliefs from their neighbours and do not contain their own mass assignment.

Table 4.2: Belief distribution functions for valuation nodes

N	Node type	Variables	Mass distribution
1	$X \sqsubseteq Y$	$I \in X$, $I \in Y$	m({0;0}, {0;1}, {1;1})=1
2	$X \equiv Y$	$I \in X$, $I \in Y$	m({0;0},{1;1})=1
3	$X \sqsubseteq \neg Y$	$I \in X$, $I \in Y$	m({0;0},{0;1},{1;0})=1
4	$X \sqcap Y$	$I \in X$, $I \in Y$, $I \in X \sqcap Y$	m({0;0;0},{0;1;0},{1;0;0},{1;1;1})=1
5	$X \sqcup Y$	$I \in X$, $I \in Y$, $I \in X \sqcup Y$	m({0;0;0},{0;1;1},{1;0;1},{1;1;1})=1
6	$\neg X$	$I \in X$, $I \in \neg X$	m({0;1},{1;0})=1
7	$\top \sqsubseteq \leq 1R$	$R(I, o_1)$, $R(I, o_2)$	m({0;0},{0;1},{1;0})=1
8	$\top \sqsubseteq \leq 1R^-$	$R(I_2, I_1)$, $R(I_3, I_1)$	m({0;0},{0;1},{1;0})=1
9	$R \equiv R^-$	$R(I_1, I_2)$, $R(I_2, I_1)$	m({0;0},{1;1})=1
10	$R \equiv Q$	$R(I_1, I_2)$, $Q(I_1, I_2)$	m({0;0},{1;1})=1
11	$R \sqsubseteq Q$	$R(I_1, I_2)$, $Q(I_1, I_2)$	m({0;0},{0;1},{1;1})=1
12	$R \equiv Q^-$	$R(I_1, I_2)$, $Q(I_2, I_1)$	m({0;0},{1;1})=1
13	$Trans(R)$	$R(I_1, I_2)$, $R(I_2, I_3)$, $R(I_1, I_3)$	m({0;0;0},{0;0;1},{0;1;0},{0;1;1}, {1;0;0},{1;0;1},{1;1;1})=1
14	$\leq 1.R$	$R(I_1, o_1)$, $R(I_1, o_2)$, $I_1 \in \leq 1.R$	m({0;0;1},{0;1;1},{1;0;1},{1;1;0})=1
15	$\geq 1.R$	$R(I_1, o_1)$, $R(I_1, o_2)$, $I_1 \in \geq 1.R$	m({0;0;0},{0;1;1},{1;0;1},{1;1;1})=1
16	$= 1.R$	$R(I_1, o_1)$, $R(I_1, o_2)$, $I_1 \in = 1.R$	m({0;0;0},{0;1;1},{1;0;1},{1;1;0})=1
17	$\forall R.X$	$R(I_1, I_2)$, $I_2 \in X$, $I_1 \in \forall R.X$	m({0;0;1},{0;1;1},{1;0;0},{1;1;1})=1
18	$\forall R.o_1$	$R(I_1, o_1)$, $I_1 \in \forall R.o_1$	m({0;0},{1;0},{1;1})=1
19	$\forall R.o_1$	$R(I_1, o_2)$, $I_1 \in \forall R.o_1$	m({0;0},{0;1},{1;0})=1
20	$\exists R.X$	$R(I_1, I_2)$, $I_2 \in X$, $I_1 \in \exists R.X$	m({0;0;1},{0;1;1},{1;0;0},{1;1;1})=1
21	$\exists R.o_1$	$R(I_1, o_1)$, $I_1 \in \exists R.o_1$	m({0;0},{1;1})=1
22	$\exists R.\top \sqsubseteq X$	$R(I_1, I_2)$, $I_1 \in X$	m({0;0}, {0;1}, {1;1})=1
23	$\top \sqsubseteq \forall R.X$	$R(I_1, I_2)$, $I_2 \in X$	m({0;0}, {0;1}, {1;1})=1

Valuation nodes represent the TBox axioms, specify how variable nodes influence each other, and are used to propagate beliefs through the network. For crisp OWL ontologies only mass assignments of 0 and 1 are possible. The principle for assigning masses is to assign the mass of 1 to the set of all combinations of variable sets allowed by the corresponding axiom. Table 4.2 shows the mass assignment functions for OWL-DL T-Box axioms[5]. These rules together with the rules from Table 4.1 constitute the core part of our approach.

[5]For nodes allowing multiple operands (e.g., intersection or cardinality) only the case of two operands is given. If the node allows more than two children, then the number of variables and the distribution function are adjusted to represent the restriction correctly

In our example we assign distributions to the variable nodes representing extracted statements based on the extractor's confidence values:

- N_1:(m(1)=0.7, m({0;1})=0.3)
- N_2:(m(1)=0.4, m({0;1})=0.6)
- N_3:(m(1)=0.6, m({0;1})=0.4)
- N_4:(m(1)=0.5, m({0;1})=0.5)

The valuation nodes obtain their distributions according to the rules specified in Table 4.2:

- N_6: m({0;0},{0;1},{1;0})=1 (rule 3, variables N_1, N_5)
- N_7: m({0;0},{1;1})=1 (rule 2, variables N_5, N_8)
- N_9: m({0;0},{1;1})=1 (rule 21, variables N_2, N_{10})
- N_{11}: m({0;0},{1;1})=1 (rule 21, variables N_3, N_{12})
- N_{13}: m({0;0;0},{0;1;0},{1;0;0},{1;1;1})=1 (rule 4, variables N_{10}, N_{12}, N_8)
- N_{14}: m({0;0},{0;1},{1;0})=1 (rule 7, variables N_3, N_4)

4.4.5 Belief propagation

The axioms for belief propagation have been formulated in [109]. The basic operators for belief potentials are marginalisation $\downarrow$ and combination $\otimes$. Marginalisation takes a mass distribution function m on the domain D and produces a new mass distribution on the domain $C \subseteq D$.

$$m^{\downarrow C}(X) = \sum_{Y^{\downarrow C}=X} m(Y)$$

For instance, if we have the function m defined on the domain $\{x, y\}$ as $m(\{0;0\}) = 0.2$, $m(\{0;1\}) = 0.35$, $m(\{1;0\}) = 0.3$, $m(\{1;1\}) = 0.15$ and we want to find a marginalisation on the domain $\{y\}$, we will get $m(0) = 0.2 + 0.3 = 0.5$ and $m(1) = 0.35 + 0.15 = 0.5$. The combination operator is represented by Dempster's rule of combination:

$$m_1 \otimes m_2(X) = \frac{\sum_{X_1 \cap X_2 = X} m_1(X_1) m_2(X_2)}{1 - \sum_{X_1 \cap X_2 = \oslash} m_1(X_1) m_2(X_2)}$$

Belief propagation is performed by passing messages between nodes according to the following rules:

1. Each node sends a message to its inward neighbour (towards the root of the tree). If $\mu^{A \to B}$ is a message from A to B, $N(A)$ is a set of neigbours of A and the potential of A is m_A, then the message is specified as a combination of messages from all neighbours except B and the potential of A:

$$\mu^{A \to B} = (\otimes \{\mu^{X \to A} | X \in (N(A) - \{B\}) \otimes m_A\})^{\downarrow A \cap B}$$

2. After a node A has received a message from all its neighbours, it combines all messages with its own potential and reports the result as its marginal.

As the message-passing algorithm assumes that the graph is a tree, it is necessary to eliminate loops. All valuation nodes constituting a loop must be replaced by a single node with the mass distribution equal to the combination of mass distributions of its constituents. Once the graph does not contain loops, any node can be selected as the root one. The marginals obtained after propagation for the nodes corresponding to initial ABox assertions will reflect updated mass distributions. After the propagation we can remove the statement with the lowest plausibility from each of the MISOs found at the diagnosis stage.

In our test example we can select the node N_6 as a root (any other node would be appropriate as well), then process inward messages and finally produce marginals starting from the root node and proceeding towards the leaves of the tree. The first steps of the process will look like the following:

- $N_1 \rightarrow N_6$: variable N_1, belief (m(1)=0.7, m({0;1})=0.3)
- $N_4 \rightarrow N_{14}$: variable N_4, belief (m(1)=0.5, m({0;1})=0.5)
- $N_{14} \rightarrow N_3$:
  - extending the domain of the message from N_4 to the domain of N_{14}: variables N_3, N_4, belief (m({0;1},{1;1})=0.5, m({0;0},{0;1},{1;0},{1;1})=0.5)
  - combining N_{14} own distribution with the message from N_4: variables (N_3, N_4), belief m({0;1})=0.5$\cdot$ 1=0.5, (m({0;0},{0;1},{1;0})=0.5$\cdot$ 1=0.5
  - obtaining the marginal for N_3: variable N_3, belief $m(0) = 0.5$, $m(\{0; 1\}) = 0.5$
- $N_3 \rightarrow N_{11}$: variable N_3, belief m(0)=$\frac{0.4 \cdot 0.5}{1-0.6 \cdot 0.5} \approx 0.29$, m(1)=$\frac{0.6 \cdot 0.5}{1-0.6 \cdot 0.5} \approx 0.43$, m{0;1}=$\frac{0.4 \cdot 0.5}{1-0.6 \cdot 0.5} \approx 0.29$
- $N_{11} \rightarrow N_{12}$: variable N_{12}, belief m(0)$\approx$ 0.29, m(1)$\approx$ 0.43, m{0;1}$\approx$ 0.29
- $N_{12} \rightarrow N_{13}$: variable N_{12}, belief m(0)$\approx$ 0.29, m(1)$\approx$ 0.43, m{0;1}$\approx$ 0.29
- $N_2 \rightarrow N_9$: variable N_2, belief m(1)=0.4, m({0;1})=0.6
- $N_9 \rightarrow N_{10}$: variable N_10, belief m(1)=0.4, m({0;1})=0.6
- $N_{10} \rightarrow N_{13}$: variable N_{10}, belief m(1)=0.4, m({0;1})=0.6

Table 4.3: Support and plausibility values for the example scenario

Statement	Before propagation		After propagation	
	Support	Plausibility	Support	Plausibility
A1: *RiskyApplicant(Ind1)*	0.70	1.0	0.66	0.94
A2: *wasBankrupt(Ind1, False)*	0.60	1.0	0.35	0.58
A3: *hasCitizenship(Ind1, UK)*	0.40	1.0	0.32	0.80
A4: *wasBankrupt(Ind1, True)*	0.50	1.0	0.33	0.65

- ...

Continuing in this way, we obtain the following Dempster-Shafer plausibility values for ABox statements: Pl(A1)=0.94, Pl(A2)=0.58, Pl(A3)=0.8, Pl(A4)=0.65 (see Table 4.3). In order to make the ontology consistent, it is sufficient to remove from both conflict sets an axiom with the lowest plausibility value (A2). In this example, we can see how the results using Dempster-Shafer belief propagation differ from the Bayesian interpretation. Bayesian probabilities, in this case, are calculated in the same way as Dempster-Shafer support values. If we use confidence values as probabilities and propagate them using the same valuation network, we will obtain the results: P(A1)=0.66, P(A2)=0.35, P(A3)=0.32 and P(A4)=0.33. In this scenario, we would remove A3 and A4 because of the negative belief bias. Also we can see that all three statements A2, A3 and A4 will be considered wrong in such a scenario (the resulting probability is less than 0.5). The Dempster-Shafer approach provides more flexibility by making it possible to reason about both support ("harsh" queries) and plausibility ("lenient" queries).

4.4.6 Implementation

We have implemented the algorithm described above to be used within the KnoFuss architecture. The inconsistency detection stage was implemented as a method for

the *dependency identification* task in the KnoFuss workflow (Fig. 3.1). The network construction and belief propagation stages constituted a single method for the *dependency resolution* task. Both methods were implemented in Java[6] using the OWL-API library[7] to work with OWL axioms and the Pellet reasoner[8] to perform ontological diagnosis for the inconsistency detection. Due to the time complexity of the procedure, its use is primarily intended for the scenarios where fusion is performed off-line, as we assumed in Chapter 1. As described in section 4.4, the algorithm includes four steps:

- Inconsistency detection.
- Constructing a belief network.
- Assigning mass distributions.
- Belief propagation.

Inconsistency detection is performed using Reiter's diagnosis algorithm and the ontological reasoner Pellet, which computes atomic conflict sets. The latter is based on the tableaux algorithm with a number of optimisations[9]. However, in the general case this algorithm still has an exponential complexity. The Reiter's algorithm implements breadth-first tree search which also has an exponential time complexity with respect to the depth of the tree. Constructing a belief network and assigning mass distributions involve the application of rules to OWL axioms and have linear complexity. The belief propagation procedure in valuation networks also has linear

[6]http://java.sun.com

[7]http://owlapi.sourceforge.net/

[8]http://clarkparsia.com/pellet/

[9]http://www.mindswap.org/papers/PelletJWS.pdf

complexity with respect to the number of edges of the graph: each message passes each edge once in each direction. However, Dempster's rule of combination, which combines beliefs at each node, also has an exponential time complexity with respect to the size of the frame of discernment Ω. Again, in the worst-case scenario when all nodes are included in loops, they have to be replaced with a single node with the frame of discernment including all elements from initial nodes. Thus, although in most real use cases we can expect conflict sets to be of small size (e.g., only three axioms in a common case of a functional property value conflict), in the worst-case scenario (e.g., involving many transitive properties) the effects of the exponential complexity can be significant.

4.5 Summary

In this chapter, we described how the Dempster-Shafer theory of evidence can be used for dealing with ABox-level inconsistencies produced by inaccurate information extraction and human errors. Based on our analysis of related approaches (section 4.2) we selected the Dempster-Shafer formalism as an answer for the research question 4: "What kind of uncertainty management framework is suitable for fusion?"

Then, we described the algorithm, which uses the structure defined by the ontology to propagate and update initial beliefs of ontological statements (ABox assertions) and, subsequently, resolve inconsistencies by selecting assertions most likely to be incorrect. This algorithm contributes to answering the research question 5: "How can we exploit uncertainty and provenance information to improve the fusion performance?"

Chapter 5

Refining instance coreferencing results using belief propagation

In Chapter 3 we outlined the fusion workflow (section 3.5) and formulated the *dependency processing* requirement (section 3.2), which stated the need to analyse interdependencies between data statements and decisions made at the earlier stages of the workflow. The *knowledge base updating* task in the workflow is aimed at performing this analysis. This chapter describes a novel method we have developed for tackling this task. The method uses the Dempster-Shafer theory of evidence for reasoning about and refining the results of the coreference resolution stage of the workflow. Hence, this method further develops the belief propagation approach described in Chapter 4. The method takes as input a set of candidate mappings between individuals produced by coreference resolution methods and refines this set, deleting some mappings likely to be incorrect and inferring new mappings implied by existing ones. Specifically, we take into account coreference mappings, data statements,

and additional information about the features of sources and mutual influence between mappings when constructing a belief network, and we re-estimate the validity of mappings using belief propagation.

This chapter is based on the following publication:

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2008). Refining instance coreferencing results using belief propagation. 3rd Asian Semantic Web Conference (ASWC 2008), Bangkok, Thailand.

5.1 Introduction

In Chapter 3 we described the KnoFuss framework and outlined its workflow, which focuses on two main fusion problems: coreference resolution and inconsistency resolution. In Chapter 4 we presented a method based on the valuation network framework [108], which uses belief propagation to resolve inconsistencies occurring when two knowledge bases are fused. This method takes the results of the coreference resolution stage as input, identifies cases when inconsistency occurs, updates the belief values of ABox statements contributing to the conflict, and selects and deletes the least reliable statements. The scope of this initial method was limited by its task: it only considers situations in which there is a logical inconsistency, and it can only provide negative evidence, where one statement indicates that some other statements are incorrect. This negative evidence can be used to delete some statements from the resulting knowledge base. In this chapter we present a generalised method which extends the belief propagation framework and uses additional information to refine and improve the results of the coreference resolution stage. We consider three relevant factors:

- *Ontological schema restrictions.* Constraints and restrictions defined by the schema (e.g., functionality relations) may provide both negative and positive evidence. In the latter case one statement reinforces other statements providing further information that they are correct. For instance, having two individuals as objects of a functional property with the same subject should reinforce a mapping between these individuals. The reverse also applies: the fact that two potentially identical individuals belong to two disjoint classes should be

considered as negative evidence.

- *Coreference mappings between other entities (context mappings).* Even if there is no explicit functionality restriction defined for an ontological property, relations between individuals may still reduce ambiguity: the fact that two similar individuals are both related to a third one may strengthen the confidence of the mapping.

- *Provenance data.* Knowledge about the quality of data may be used to assign confidence values to class and property assertions. This is important when we need to judge whether a mapping, which violates the domain ontology, is incorrect or the conflict is caused by a data statement. Knowledge about the "cleanness" of a source (e.g., whether duplicates occur in a given source) provides additional evidence about potential mappings.

In the previous chapter we considered the uncertainty degrees of the data statements in merged knowledge bases. The new factors listed above also require dealing with uncertainty. Mappings are created by attribute-based matching algorithms, which do not provide 100% precision. Different ontological relations have different impact as evidence for mappings: if two similar *foaf:Person* individuals are both connected to a *sweto:Publication* individual via a *sweto:author* relation, this provides much stronger evidence supporting an identity mapping, than if they were related to a *tap:Country* individual *#USA* via a *#citizenOf* relation. All these kinds of uncertainty have to be captured and managed. The basic algorithm presented in chapter 4 has to be significantly extended to handle a more general task.

5.2 Refining coreference mappings

The method receives as its input a set of candidate mappings between individuals in source and target knowledge bases. In order to perform belief propagation, these mappings, along with relevant parts from both knowledge bases, must be translated into valuation networks. Building a large network from complete knowledge bases is both computationally expensive and unnecessary, as not all triples are valuable for analysis. Hence, we select only relevant triples considering only statements and axioms which mutually influence each other. First, as in Chapter 4, these include the statements which, when taken together, lead to a conflict. Moreover, we include into the analysis the statements which can provide positive evidence and reinforce the belief values of some *owl:sameAs* mappings: these include instantiations of explicitly defined functional and inverse functional properties, but also other "influential" properties which provide implicit evidence (we will discuss such properties in section 5.2.2). Conflicts are detected by selecting all statements in the neighbourhood of potentially mapped individuals and checking their consistency with respect to the domain ontology (to this purpose we use the Pellet OWL reasoner with the explanation service). If the reasoner finds an inconsistency, all statements which contribute to it are considered relevant.

Then, belief networks are constructed by applying the rules defined in [86] and the extended set described in section 5.2.1, and initial beliefs are assigned to variable nodes. For each *owl:sameAs* variable node a belief value is determined according to the precision of the corresponding coreferencing algorithm which produced the *owl:sameAs* mapping. A coreference resolution algorithm can produce two kinds of

mappings: "probably correct" ones, exceeding the optimal similarity threshold for the algorithm (the one, which maximised the algorithm's F-measure performance), and "possibly correct" ones with similarities below the optimal threshold, but achieving at least 0.1 precision. Each variable node representing a class or property assertion receives its initial belief score based on its attached provenance data: the reliability of its source and/or its extraction algorithm. After that the beliefs are updated using the belief propagation procedure described in Chapter 4, and for each mapping a decision about its acceptance is taken.

The most significant part of the algorithm is network construction. At this stage we exploit the factors listed in section 5.1. In the following subsections we describe how this is done in more detail.

5.2.1 Exploiting ontological schemata

Logical axioms defined by a schema may have both positive and negative influence on mappings. First, some OWL axioms impose restrictions on the data. If creating an *owl:sameAs* relation between two individuals violates a restriction, the confidence of the mapping should be reduced. Second, object properties defined as *owl:FunctionalProperty* and *owl:InverseFunctionalProperty* allow us to infer equivalence between individuals. Finally, having an *owl:sameAs* relation between a pair of individuals may directly contradict an *owl:differentFrom* relation between the same individuals. When we developed our initial algorithm, we considered the situation after the instance coreference resolution decisions have been applied and tried to find and resolve inconsistencies in an already fused ontology. Thus, the original set of rules described in Chapter 4 does not allow reasoning about uncertain coreference

Table 5.1: Extended belief network construction rules

N	Axiom	Pre-conditions	Nodes to create	Links to create
1	*sameAs*	$I_1 = I_2$	$N_1 : I_1 = I_2$ (variable)	
2	*differentFrom*	$I_1 \neq I_2$	$N_1 : I_1 \neq I_2$ (variable)	
3	*sameAs*	$N_1 : I_1 = I_2$ (variable), $N_2 : R(I_2, I_3)$	$N_3 : I_1 = I_2$ (valuation), $N_4 : R(I_1, I_3)$	(N_1,N_3),(N_2,N_3), (N_3,N_4)
4	*differentFrom*	$N_1 : I_1 = I_2$ (variable), $N_2 : I_1 \neq I_2$ (variable)	$N_3 : I_1 \neq I_2$ (valuation)	(N_1,N_3),(N_2,N_3)
5	*Functional Property*	$\top \sqsubseteq \leq 1R$, $N_1 : R(I_3, I_1)$, $N_2 : R(I_3, I_2)$, $N_3 : I_1 = I_2$	$N_4 : \top \sqsubseteq \leq 1R$	(N_1,N_4),(N_2,N_4), (N_3,N_4)
6	*InverseFunctional Property*	$\top \sqsubseteq \leq 1R^-$, $N_1 : R(I_1, I_3)$, $N_2 : R(I_2, I_3)$, $N_3 : I_1 = I_2$	$N_4 : \top \sqsubseteq \leq 1R^-$	(N_1,N_4),(N_2,N_4), (N_3,N_4)

resolution decisions and is insufficient for reasoning about coreference relations. To resolve this issue, we developed an extended set of rules, which allow us to process the subontologies containing uncertain *owl:sameAs* relations and reason about them. In this section, we present this novel set of additional rules. Table 5.1 contains the translation rules and Table 5.2 lists the additional belief assignment functions for corresponding valuation nodes.

The *owl:sameAs* axiom and its opposite, *owl:differentFrom*, are special in comparison with others, which we considered in Chapter 4. While they belong to the ontology ABox and represent relations between instances, they also define schema-level rules, which allow new statements to be inferred. To capture this situation, each *owl:sameAs* statement leads to the creation of two types of nodes. First, for an *owl:sameAs* statement we create a variable node in the belief propagation network and assign its support value (usually, according to the estimated precision of the algorithm which produced it) (rule 1). Then, this node is used as a pre-condition for inferring new statements: if one individual from a pair linked by the *owl:sameAs*

relation is a subject or an object of a statement, then the statement holds for another individual as well. This is captured by rule 3, which creates a *owl:sameAs* valuation node and a variable node corresponding to the inferred statement. *owl:differentFrom* creates a variable node as well (rule 2), but it is connected to the network only if there is already a *owl:sameAs* variable node corresponding to the link between the same individuals.

Another change concerns *owl:FunctionalProperty* and *owl:InverseFunctionalProperty* axioms. Their interpretation is changed in comparison with the interpretation in section 4.4.3. Chapter 4. After applying coreference resolution decisions to a knowledge base, when an individual had different property values for a functional property, only two possibilities were considered: that either one of the statements is incorrect. However, when reasoning about coreferencing results, a third option can be considered: that two values are in fact the same. Rules 5 and 6 refer to this issue. To avoid spurious *owl:sameAs* relations being inferred, we use a restriction: valuation nodes corresponding to functional and inverse functional property axioms can only be linked to already existing *owl:sameAs* nodes. In this way, they can only increase similarity between individuals, which were already considered potentially equal. Otherwise the functionality node is treated as in Chapter 4: as a strict constraint violated by two property assertion statements. Belief distribution functions are assigned using the same principle as in Chapter 4: all belief mass (1) was assigned to a set of all possible combinations of variable truth values, except for those forbidden by the corresponding ontological axiom.

To illustrate the work of the algorithm we will use an example with datasets from the citations domain. One dataset (DBLP) contains an individual *Ind1* describing

Table 5.2: Extended belief distribution functions for valuation nodes

N	Axiom	Node type	Variables	Mass distribution
1	*sameAs*	$I_1 = I_2$	$I_1 = I_2$, $R(I_1, I_3)$, $R(I_2, I_3)$	m({0;0;0}, {0;0;1}, {0;1;0}, {0;1;1}, {1;0;0}, {1;1;1})=1
2	*differentFrom*	$I_1 \neq I_2$	$I_1 = I_2$, $I_1 \neq I_2$	m({0;1},{1;0})=1
3	*Functional Property*	$\top \sqsubseteq \leq 1R$	$R(I_3, I_1)$, $R(I_3, I_2)$, $I_1 = I_2$	m({0;0;0}, {0;0;1}, {0;1;0}, {1;0;0}, {1;1;1})=1
4	*Inverse Functional Property*	$\top \sqsubseteq \leq 1R^-$	$R(I_1, I_3)$, $R(I_2, I_3)$, $I_1 = I_2$	m({0;0;0}, {0;0;1}, {0;1;0}, {1;0;0}, {1;1;1})=1

the following paper:

> D. Corsar, D. H. Sleeman. Reusing JessTab Rules in Protege. Knowledge-Based Systems 19(5). (2006) 291-297.

Another one (EPrints) also contained a paper *Ind2* with the same title:

> Corsar, Mr. David and Sleeman, Prof. Derek. Reusing JessTab Rules in Protege. In Proceedings The Twenty-fifth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (2005), pages pp. 7-20, Cambridge, UK.

This illustrates a common case where the same group of researchers first publishes their research results at a conference and then submits the extended and revised paper to a journal. An attribute-based coreferencing algorithm (Jaro-Winkler similarity applied to the title) with a good overall performance on the dataset (precision about 0.92 and F-measure about 0.94) incorrectly considered these two papers identical. However, a mapping between these individuals violated two restrictions: the individual belonged to two disjoint classes simultaneously and had two different values for the functional property *year*. The inconsistencies were detected by the conflict detection algorithm, which produced two sets of relevant statements: {*owl:sameAs(Ind1,*

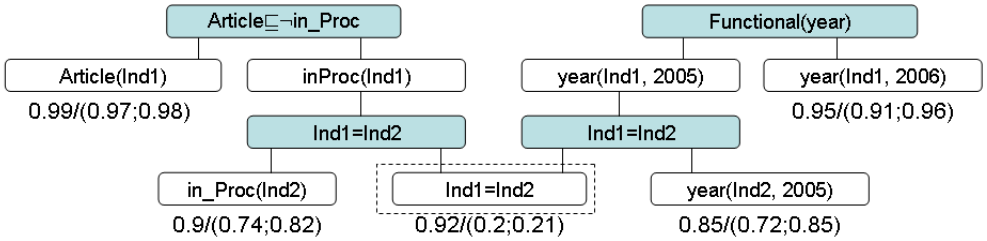

Figure 5.1: Example of a belief network including a coreference relation. The numbers show the support before propagation and, in brackets, support and plausibility after propagation. *Ind1* and *Ind2* represent two papers with the same title: the journal article and the conference paper respectively. The *owl:sameAs* link between them (*Ind1=Ind2*) has a high initial support (0.92), but violates two ontological restrictions: disjointness between journal and conference articles (left part of the graph) and functionality of the publication year property (right). As a result, after propagation it receives both low support and low plausibility.

Ind2); *Article(Ind1)*; *Article_in_Proceedings(Ind2)*; *owl: disjointWith(Article, Article_in_Proceedings)*} and {*owl:sameAs(Ind1, Ind2)*; *year(Ind1, 2006)*; *year(Ind2, 2005)*; *owl: FunctionalProperty(year)*}. Since these sets share a common statement (*sameAs* link), they are translated into a single valuation network (Fig. 5.1). Although in our example the initial support of the mapping was higher than the support of both statements related to Ind2 (*Article_in_Proceedings(Ind2)* and *year(Ind2, 2005)*), after belief propagation the incorrect *owl:sameAs* mapping was properly recognised and received the lowest plausibility (0.21).

5.2.2 Influence of context mappings

Belief propagation for properties explicitly defined as functional is a trivial case, which can be handled by direct application of the rules defined in tables 5.1 and 5.2. However, properties which allow multiple values are also valuable as a means to narrow the context of matched individuals and increase similarity between them. We have to estimate the impact of the relation and model this in the network. As

shown in Table 5.2 (row 1), by default the valuation node for the *owl:sameAs* relation is defined in such a way that the belief in $I_1 = I_2$ is completely independent from a strong belief for both $R(I_3, I_1)$ and $R(I_3, I_2)$. The functionality axiom represents an opposite scenario: having a belief 1.0 for both $R(I_3, I_1)$ and $R(I_3, I_2)$ implies the belief 1.0 for $I_1 = I_2$. The actual strength of influence for a property may lie between these extreme cases. In order to utilise such links the network construction algorithm receives for each relevant property a vector $< n_1, n_2 >$, where n_1 and n_2 determine the impact of the link in the forward (subject to object) and reverse (object to subject) directions respectively. The impact in the two directions may be different: having two people as first authors of the same paper strongly implies their equivalence, while having the same person as the first author of two papers with a similar title does not increase the probability of two papers being the same. The *owl:sameAs* valuation node, combining variables $I_1 = I_2$, $R(I_2, I_3)$, $R(I_1, I_3)$ will receive two belief assignments instead of one: m({0;0;0}, {0;0;1}, {0;1;0}, {1;0;0}, {1;1;1})=n_1 and m({0;0;0}, {0;0;1}, {0;1;0}, {0;1;1}, {1;0;0}, {1;1;1})=$1 - n_1$. One possible way to determine coefficients $< n_1, n_2 >$ is to learn them from training data.

Also, some relevant relations may be implicit and undefined in the ontology. For instance, the same group of people may be involved in different projects. If the link between a project and a person is specified using a property *akt:has-project-member*, when two knowledge bases describing two non-overlapping sets of projects are combined, the relations between people cannot be utilised. In order to capture these implicit relations we can add artificial properties, which connect individuals belonging to the same sets, into the ontology. Co-authorship analysis, commonly used in the citation matching domain, is a special case of this scenario. In Fig. 5.2a

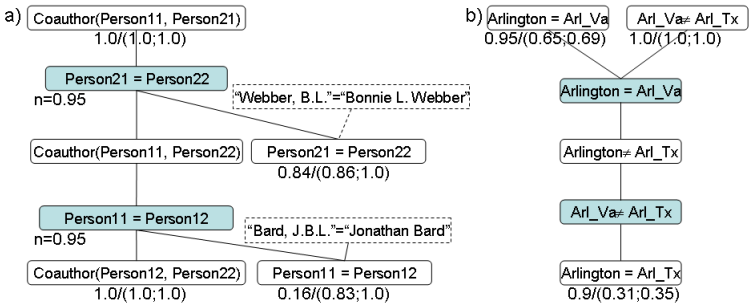

Figure 5.2: Examples of belief networks illustrating different evidence types: (a) the usage of artificial set membership relations (section 5.2.2) and (b) processing competing mappings knowing that a source does not contain duplicates (section 5.2.3). The numbers show the belief before propagation and belief and plausibility after propagation.

an example of the use of co-authorship relations is shown. Two people, "Webber, B. L." and "Bard, J. B. L.", are co-authors of a paper mentioned in the source knowledge base. The same people are also mentioned as co-authors of a different paper in the target knowledge base, but under different names ("Bonnie L. Webber" and "Jonathan Bard"). Although the string similarity-based coreference resolution method was able to find an equivalence between "Webber, B. L." and "Bonnie L. Webber" with a high degree of confidence (0.84), it assigned a low confidence to the mapping between "Bard, J. B. L." and "Jonathan Bard" (0.16). However, in our tests, co-authorship relation had a strong impact ($n_1 = 0.95$, $n_2 = 0.95$), and we were able to reinforce the mapping between "Bard, J. B. L." and "Jonathan Bard": resulting support value was 0.84.

5.2.3 Provenance data

The estimated reliability of a source is directly used at the starting stage when initial beliefs are assigned to variable nodes representing class and property assertions. Thus, if a violation of a functional restriction is caused by a property assertion with a low belief, its impact will be insufficient to break the *owl:sameAs* link. Another important factor is the knowledge about duplicate individuals in a knowledge base. For instance, one knowledge base (AGROVOC) contains an individual *fao:arlington*. If we match AGROVOC against the UTexas geographical ontology, which contains two individuals "arlingtonVa" and "arlingtonTx", two pairs will be matched. Although the similarity of one pair is slightly greater than another one, both values are above the threshold and both these individuals can potentially be matched to the first individual. However, knowing that this particular knowledge base does not contain duplicates, allows us to add a corresponding *owl:differentFrom* variable node into the network (Fig.2b). Updating beliefs allows us to reject one of the two competing options.

5.3 Summary

In this chapter we described the generalisation of the Dempster-Shafer belief propagation algorithm to the problem of refining coreference resolution results. The goal of the algorithm is to capture and utilise the interdependencies between pieces of data in order to make decisions about what information should be put into the knowledge base. These decisions are made based on the measured and updated uncertainty of pieces of data. The initial version of the algorithm presented in Chapter 4 took into

account only one type of interdependencies (logical inconsistency) and only one source of uncertainty (confidence assigned to data statements). Statements contributing to a logical inconsistency provided mutual negative evidence. In this chapter we have shown how the algorithm can be extended by considering other patterns of interdependencies in data and another source of uncertainty: decisions made by algorithms at the earlier stages of the fusion workflow (coreference resolution). This method further contributes to answering the research question 5: "How can we exploit uncertainty and provenance information to improve the fusion performance?"

Chapter 6

Evaluation

In this chapter we describe the results of experiments we have performed to test the framework and the algorithms described in previous chapters. First, the use of application contexts to aid machine learning algorithms for coreference resolution is tested (see Chapter 3). Then, the results of these experiments are used to test the inconsistency resolution approach described in Chapter 4 and the more generic belief propagation approach described in Chapter 5.

6.1 Introduction

Experiments were carried out to demonstrate the applicability of the following aspects:

- The architecture and the problem-solving method selection approach, in particular, using the class hierarchy for method configuration as described in section 3.8. This approach was proposed as a contribution to addressing research question 3: "How can we exploit axioms defined in domain ontologies to improve the performance of fusion algorithms?"

- The applicability of the belief propagation algorithm for inconsistency resolution and coreference refinement. First, these tests were needed to check the suitability of the Dempster-Shafer formalism as an answer to research question 4: "What kind of uncertainty management framework is suitable for fusion?". Second, we wanted to verify the appropriateness of belief propagation based on valuation networks as a means to address question 5: "How to exploit uncertainty and provenance to improve fusion performance?"

A common way of measuring the quality of data fusion is calculating precision and recall measures by comparing the results obtained automatically with a manually constructed gold standard. So selecting appropriate datasets was the primary issue when we prepared the evaluation setup. In order to test the aspects we were interested in, the datasets had to possess several features.

First, the datasets had to cover the same domain and have overlapping sets of instances. From these overlapping sets we needed to construct the actual gold standards to measure precision and recall. At the same time, the datasets need to use

different URI assignment schemes and, preferably, use different literal formats to describe individuals referring to the same entities, so that the coreference resolution stage did not become trivial.

Second, we needed a domain ontology with certain properties to be used by the datasets. In particular, in order to perform the first part of the tests we needed to have at least two classes with a common superclass and both of these classes needed to have individuals attached to them. To perform the second part of the tests, the ontology needed to contain restrictions which could lead to inconsistencies and be used to detect them.

Third, there was the need for initial belief assignments to test inconsistency resolution and coreference refinement using belief propagation. These were not directly important for the first part of our tests: usage of the class hierarchy for method configuration. However, these tests involved the coreference resolution stage of the fusion process and the results obtained at this stage could be reused for the coreference refinement stage. For this reason, it made sense to use the same datasets for both parts of the tests. Because the goal of belief propagation is to reveal incorrect statements, the datasets needed to contain sufficient errors to provide illustrative examples.

Finally, to create the gold standard, we needed to be able to determine ourselves which statements were incorrect, so some highly specialist datasets were not suitable. Also, the identity of individuals had to be non-ambiguous and it had always to be possible to decide whether any two individuals were equivalent or not.

Because of this last requirement, we have chosen the domain of scientific publications: on each conflict it is relatively simple to find out manually, which statement or which coreferencing link is incorrect. We tried to select datasets constructed in

different ways so that we could rank the reliability of their data. Thus, one of our datasets was constructed automatically from reliable sources (DBLP), one was automatically created using information extraction from text (Rexa) and another one was manually created by the authors of papers (AKT EPrints). Initial belief functions were assigned according to this provenance information as well as manual checking of annotations. These datasets contained publications of two different disjoint categories (journal articles and conference papers), which had a common superclass representing generic publications. We translated all these datasets according to a common domain ontology (SWETO-DBLP) and used them in both parts of our tests.

Additionally, for the second part of our tests we used a common Cora dataset, created as a benchmark in the database community. However, this does not satisfy all of the above requirements: different publication subtypes are not distinguished and the provenance of different statements is not preserved. Hence, we had to assume that all statements have the same reliability. Thus, this dataset could only be used to test the coreference refinement procedure.

In the following sections we describe both parts of our tests. First, section 6.2 covers the tests of the context-dependent method configuration technique. Then, section 6.3 discusses the evaluation of our Dempster-Shafer belief propagation approach. Each section contains a description of the experimental setup and a discussion of the results.

6.2 Context-dependent method configuration

When designing our architecture described in Chapter 3, we formulated the *granularity* requirement, which stated that a method could be invoked with different parameters for different types of data. The goal of this was to allow the method's settings to be optimised for a specific subset of data from the knowledge base. One of the ways to obtain optimal parameters is the use of machine learning techniques. In section 3.8 we discussed how to use the class hierarchy defined in the domain ontology in order to manipulate the set of available training instances and produce optimal method configuration parameters. Our experiments described in this section were performed to validate this method configuration procedure, in particular, combining training instances from different subclasses to generate a generic decision model for a superclass.

6.2.1 Experimental setup

In our tests we compared the performance of coreference resolution algorithms when applied to instances of classes at different levels of the class hierarchy. In particular, we were interested in the quality of coreference resolution as well as the robustness of the obtained decision models. In order to test the system, we used the following datasets from the domain of scientific publications:

- AKT EPrints archive[1]. This dataset contains information about papers produced within the AKT research project.

[1] http://eprints.aktors.org/

- Rexa dataset[2]. The dataset extracted from the Rexa search server, which was constructed at the University of Massachusetts using automatic IE algorithms.

- SWETO DBLP dataset[3]. This is a publicly available dataset listing publications from the computer science domain.

The SWETO-DBLP dataset was originally represented in RDF. Two other datasets (AKT EPrints and Rexa) were extracted from the HTML sources using specially constructed wrappers and structured according to the SWETO-DBLP ontology (Fig. 6.1). The ontology describes information about scientific publications and their authors and extends the commonly used FOAF ontology[4]. Authors are represented as individuals of the *foaf:Person* class and a special class *sweto:Publication* is defined for publications as an extension of the class *foaf:Document*. We consider two relevant datatype properties used for *sweto:Publication* individuals: the standard *rdfs:label* property expressing the title of a publication and *sweto:year* denoting the publication year. Several subclasses are defined for the generic class *sweto:Publication* to describe different types of publications. We considered two of them as they were represented in all three datasets: *sweto:Article*, which includes journal publications, and *sweto:Article_in_Proceedings*, which includes publications in conference proceedings. Additional properties were defined for individuals of these classes to store the publication venue: *sweto:journal_name* and *sweto:volume* for journal articles and *sweto:booktitle* for articles in proceedings.

Experiments were performed with the following matching algorithms:

[2]http://www.rexa.info/

[3]http://lsdis.cs.uga.edu/projects/semdis/swetodblp/august2007/opus_august2007.rdf

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

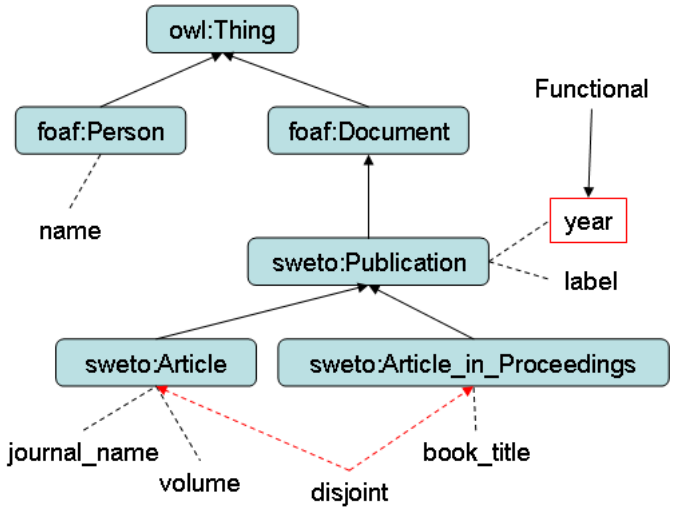

Figure 6.1: SWETO-DBLP ontology: class hierarchy and restrictions

- Jaro-Winkler metrics directly applied to the label.
- L2 Jaro-Winkler applied to the label.
- Average L2 Jaro-Winkler over the properties of the class *Publication*.
- Average L2 Jaro-Winkler over all available properties.
- Adaptive learning clustering algorithm employing TF-IDF and N-gram metrics [97].

The Jaro-Winkler algorithm was used as a representative of string matching algorithms: it outperformed Levenshtein on our dataset during the preliminary tests and was specifically designed to work with peoples' names, which were part of our data. L2 Jaro-Winkler is a mixture of string similarity and set similarity measures: it tokenises both compared values, then each pair of tokens is compared using the standard Jaro-Winkler algorithm and the maximal total score is selected. It is able to work in cases when the order of words in a multi-word string value is not important for establishing their identity (e.g., "Enrico Motta" and "Motta, Enrico"). We assumed

that the algorithms did not have any domain-specific knowledge, so such common techniques as analysing co-authors to disambiguate a person were not used. The links between papers and authors were not exploited either. The process was structured as follows. First, we trained each algorithm to recognise matching individuals of each direct class (*Person*, *Article* and *Article_in_Proceedings*). For this, a subset of available individuals of each class was selected as the training set, and a decision model was learned from this training set (for the string similarity techniques the only parameter of the model was the optimal threshold). Then, we ran the algorithms with the learned parameters and measured their performance. Then, we combined the individuals of classes *Article* and *Article_in_Proceedings* and performed the same tests for their superclass *Publication*. In this case, the training set was larger, but only the properties common for individuals of both classes were available.

6.2.2 Experimental results

The results of our tests are given in Table 6.1. As a performance metric we used the commonly employed F1 measure, which combines precision and recall. In the coreference resolution task, as in the generic classification task, the precision is defined as $p = \frac{tp}{tp+fp}$, where tp is the number of *true positives* (correct mappings returned by the algorithm), and fp is the number of *false positives* (mappings returned by the algorithm, but not correct). Accordingly, the recall is defined as $p = \frac{tp}{tp+fn}$, where fn is the number of false negatives (correct mappings which were not discovered by the algorithm). The F1 measure calculated as $F1 = \frac{2pr}{p+r}$.

We performed tests for each class and each method 5 times, each time randomly selecting 1/3 of individuals belonging to a class as a training set. In Table 6.1 we give

Table 6.1: Test results: coreference resolution.

No	Datasets	Article		Article_in_ Proceedings		Publication		Person	
		F1	σ	F1	σ	F1	σ	F1	σ
AKT/Rexa									
1	Direct Jaro-Winkler (label)	0.92	0.09	0.85	0.03	0.87	0.01	0.29	0.01
2	L2 Jaro-Winkler (label)	0.89	0.07	0.9	0.01	0.9	0.01	0.84	0.01
3	L2 Jaro-Winkler (label+year)	0.9	0.06	0.92	0.02	0.93	0.03		
4	L2 Jaro-Winkler (all)	0.48	0.11	0.74	0.03				
5	Clustering	0.69	0.39	0.85	0.043	0.82	0.045		
AKT/DBLP									
6	Direct Jaro-Winkler (label)	0.87	0.05	0.94	0.01	0.93	0.02	0.10	0.04
7	L2 Jaro-Winkler (label)	0.66	0.1	0.52	0.03	0.55	0.01	0.63	0.03
8	L2 Jaro-Winkler (label+year)	0.88	0.07	0.88	0.01	0.89	0.02		
9	L2 Jaro-Winkler (all)	0.24	0.06	0.54	0.03				
10	Clustering	0.75	0.16	0.9	0.03	0.83	0.09		
Rexa/DBLP									
11	Direct Jaro-Winkler (label)	0.9	0.04	0.91	0.01	0.92	0.01	0.90	0.03
12	L2 Jaro-Winkler (label)	0.7	0.03	0.7	0.01	0.7	0.01	0.72	0.04
13	L2 Jaro-Winkler (label+year)	0.93	0.02	0.88	0.05	0.89	0.01		
14	L2 Jaro-Winkler (all)	0.89	0.02	0.89	0.02				
15	Clustering	0.86	0.04	0.89	0.01	0.89	0.03		

the average value of the F1 measure for 5 tests and its standard deviation. While the average value of the F1 measure represented the quality of the output produced by the algorithm, the standard deviation indicated the robustness of the decision model. Given the same amount of training data, a more robust learning algorithm is more likely to produce a decision model which achieves a certain quality of the output mappings. Smaller standard deviation indicated that the number of training individuals was sufficient for the learning algorithm to converge and the model was closer to the optimal one. Robustness is an important feature of the model because it allows estimating the quality of the method's output when the method is reused. Thus, in case of similar precision/recall performance of two models, a more robust model is preferable.

In all our sets the majority of the *Publication* individuals belonged to the class *Article_in_Proceedings*, while there were fewer *Article* individuals. The results indicate that a decision model over the combined dataset (*Publication*) was usually more robust (had smaller standard deviation) in comparison with the class with few

training instances (*Article*). As expected, the gain in robustness was greater for the more sophisticated machine learning clustering algorithm (rows 5, 10, 15) because it required more training examples to learn an optimal decision model. Two factors influenced the performance of the generic model learned from the combined set of training examples. On the one hand, analysing more training data allowed better classification accuracy to be achieved. Again, this factor was more relevant when the class *Article* did not have sufficient individuals for the model to converge (rows 5, 10). On the other hand, sometimes the performance was lower due to spurious mappings between instances belonging to different classes, when an *Article* individual was mapped to an *Article_in_Proceedings* one. This latter problem can be handled at the dependency processing stage, because such mappings lead to an inconsistency.

In general, we can summarise that the method configuration mechanism outlined in section 3.8 provides more flexibility in comparison with the generic scenario (one set of parameters for the whole knowledge base, as in schema matching tools): method configuration can be optimised for each specific type of data, and for each class more appropriate methods and configuration settings can be selected. For example, it is possible to use the L2 Jaro-Winkler metric for *Person* individuals and the direct Jaro-Winkler one for publications, and to select between a generic model for all publications and two different models for *Article* and *Article_in_Proceedings* individuals.

However, there are also important factors which limit the methods' reuse. First, in order to reuse coreferencing methods between classes linked into a hierarchy we have to assume that the properties significant for identifying the objects are inherited. While this is a common pattern, which holds in our scenario, it may not be the case in

all ontologies. Second, encoding mismatches between different datasets (mentioned in section 7.2) can limit the reuse of a method if they are not known in advance. For instance, a pair of labels “Sleeman, Derek” in EPrints and “Derek Sleeman” in DBLP could not be captured by the direct Jaro-Winkler algorithm, which performed best for the pair Rexa/DBLP.

6.3 Exploiting data interdependencies using belief propagation

6.3.1 Experimental setup

In order to test the belief propagation mechanism the same datasets were reused together with the output of coreference resolution algorithms. Three constraints were added into the SWETO-DBLP ontology:

- Classes *opus:Article* and *opus:Article_in_Proceedings* were stated as disjoint.
- The property *opus:year* (the year of publication) was defined as functional.
- The object property *author* connecting a publication with a set of authors was defined as functional.

Also, two variations of another common testbed dataset called Cora were included into consideration. These were:

1. Cora(I) dataset[5]. A citation dataset used for machine learning tests.
2. Cora(II) dataset. Another version of the Cora dataset used in [32].

The Cora(I) dataset was created at the University of Massachusetts for the purpose of testing machine-learning clustering algorithms. It contains 1295 references and is deliberately noisy: e.g., the gold standard contains some obviously wrong mappings[6].

[5]http://www.cs.utexas.edu/users/ml/riddle/data/cora.tar.gz

[6]For instance, two papers by N. Cesa-Bianchi et al. "How to use expert advice. 25th ACM Symposium on the theory of computing (1993) 382-391" and "On-line prediction and conversion strategies. Eurocolt'93 (1993) 205-216" were considered the same in Cora(I).

We translated this dataset into RDF using the SWETO-DBLP ontology. The authors of Cora(II) [32] translated the data from Cora(I) into RDF according to their own ontology and cleaned the gold standard by removing some spurious mappings, so the results achieved on Cora(I) and Cora(II) are not comparable. The data and the gold standard mappings in Cora(II) are significantly cleaner than in Cora(I). Also in Cora(II) all *Person* individuals were initially considered different while in Cora(I) individuals with exactly the same name were assigned the same URI, which led to a significant difference in the number of individuals (305 vs 3521) and, consequently, in performance measurements. In our tests we tried to merge each pair of datasets from AKT/Rexa/DBLP and to find duplicates in the Cora datasets (i.e., the same dataset played the role of both the source and the target knowledge base). Given that neither Cora dataset distinguishes between journal and conference articles, we represented publication venues (specific journal issues or conference proceedings) as individuals, and defined relations between papers and venues as functional. Also the Cora(II) ontology described pages as two integer properties *pageFrom* and *pageTo*, which allowed us to add a functionality restriction on them as well.

A priori belief masses were assigned based on information available about the datasets. It was known that the AKT EPrints archive was created by the authors themselves, who entered the data about their publications manually. The Rexa dataset was extracted using automatic IE algorithms (the authors reported extraction accuracy of 0.97 and coreferencing accuracy "in the 90s"[7].) However, sometimes the information was incorrectly reported in the sources (e.g., when it was extracted from a citation in a third-party publication), which lowers the actual quality of the

[7]http://www.rexa.info/faq

data. The DBLP dataset was primarily constructed using the data reported in the proceedings and journal contents, which makes it a more reliable source.

A priori mass assignment was performed in the following way. First, the data-sources were ranked according to our confidence in their quality (DBLP > Rexa > EPrints). As a rough clue to ranking the sources according to quality, the coreference quality of the papers' authors was used. Cases where the same author was referred to in the same dataset using different labels were considered as an error and the percentage of correct individuals was calculated (for EPrints this percentage was 0.46, for Rexa 0.63 and for DBLP 0.93). Then, we treated the class assignments as more reliable than datatype property assignments because the IE algorithms used by Rexa and the HTML wrappers sometimes made errors by assigning the wrong property (e.g., venue instead of year) or by assigning the borders of the value incorrectly (e.g., dropping part of the paper's title). Finally, for the Rexa dataset we had additional information: each paper record had a number of citations indicating the number of sources referring to the paper. We estimated the dependency between the reliability of records and the number of citations by randomly selecting a subset of paper records and manually counting the number of "spurious" records, which contained some obvious error (e.g., like assigning the name of the conference as paper title). We randomly selected 400 records for each value of the *hasCitations* property from 0 to 5 and counted the number of spurious records. If the total number of papers for some interval was lower than 400, then we selected all available records in the interval. Based on these reliability assignments, we adjusted the reliability of datatype property assignments for the Rexa dataset.

Table 6.2: Initial belief mass assignment

Dataset	Class assertions	Datatype assertions
DBLP	0.99	0.95
Rexa	0.95	0.81 (<2 citations) 0.855 (>2 citations)
EPrints	0.9	0.85
Cora(I & II)	N/A	0.6

For Cora datasets we did not have the provenance of sources, from which each citation was extracted. Publications in the Cora datasets were not further classified into journal and conference articles, so class assertions were not relevant. Knowing that the data in the Cora datasets were noisy, beliefs were assigned in such a way that a disagreement on a single property value was not sufficient to break the mapping (initial support of 0.6 was used). This led us to assign belief masses to the statements from each source as shown in Table 6.2. Of course, such confidence estimation was subjective, but we cannot expect it to be precise in most real-life scenarios, unless the complete gold standard data is available in advance. Additionally, in order to test the conflict resolution performance of the algorithm in a situation where the quality of one of the data sources is low and to see the patterns of the inconsistency resolution process, we introduced additional noise into our datasets. We did it in the following way: for one of the datasets in the pair (the smaller one, EPrints or Rexa depending on the case) we randomly mutated 40% of *rdf:type* assertions (changing from *Article* to *Article_in_Proceedings* and vice versa) and *opus:year* assertions (by + or -1). The support belief mass of all statements in the dataset was proportionally reduced: the values in the rows 2 and 3 in Table 6.2 were multiplied by 0.6. We did not perform these tests with Cora(I) and Cora(II) because for them we only had the gold standard for coreference resolution and did not know for sure which data

statements were correct in case of a data value conflict.

For attribute-based coreferencing we used string similarity measures applied to a paper title or person's name. In particular, we used Jaro-Winkler and Monge-Elkan metrics applied to whole strings or tokenised strings (L2 Jaro-Winkler). An initial belief mass distribution for each *owl:sameAs* relation was assigned according to the precision of the algorithm which produced it.

In the following two subsections we discuss the results of these tests. First, we focus on the capability of the belief propagation algorithm to find the correct resolution of the conflict causing inconsistency. Thus, we only considered AKT/Rexa/DBLP datasets for which we knew not only the correct coreference mappings, but also correct and erroneous data statements. This allowed us to measure how well the algorithm was able to resolve conflicts and how often it was able to determine which statements or coreference mappings were incorrect. Moreover, we only considered the cases of an ontological inconsistency, where some disjointness or functionality restriction was violated. As a result, we only took into account individuals of the class *sweto:Publication*, for which these restrictions had been defined.

Second, we considered the capability of the algorithm to improve the quality of the coreference resolution stage and only focused on the quality of coreference mappings before and after the refinement. At this stage we were able to consider both Cora datasets and individuals of the class *foaf:Person*: for these individuals the belief propagation algorithm could exploit positive evidence (inferring new mappings), rather than negative evidence (violation of ontological restrictions). Thus, subsection 6.3.3 also subsumes the results reported in subsection 6.3.2, but focuses on different aspects and describes more tests.

Table 6.3: Test results: inconsistency resolution

No	Matching algorithm	Total clusters	Matching precision (before)	Conflicts found	Conflict resolution accuracy	Matching precision (after)
	EPrints/Rexa					
1	Jaro-Winkler	88	0.95	9	0.61	0.97
2	L2 Jaro-Winkler	96	0.88	13	0.73	0.92
3	Jaro-Winkler (mutated dataset)	88	0.95	55	0.92	0.95
	EPrints/DBLP					
4	Jaro-Winkler	122	0.92	12	0.83	0.99
5	L2 Jaro-Winkler	217	0.39	110	0.9	0.84
6	Jaro-Winkler (mutated dataset)	122	0.92	84	0.91	0.92
	Rexa/DBLP					
7	Jaro-Winkler	305	0.9	21	0.73	0.94
8	L2 Jaro-Winkler	425	0.55	149	0.87	0.82
9	Jaro-Winkler (mutated dataset)	305	0.9	213	0.94	0.9

6.3.2 Experimental results: inconsistency resolution

We measured the quality of inconsistency resolution by comparing the resulting ranking produced after the belief propagation with the "correct" conflict resolution. A conflict was considered correctly resolved if the genuinely incorrect statements got the lowest belief after propagation. In cases where a conflict set contained several incorrect statements and only some of them were correctly recognised, we assigned a reduced score to such a conflict (e.g., 0.33 if only one statement out of three was properly recognised as wrong). Incorrect statements which received low support but were plausible counted as 0.5 of a correct answer. The results we obtained are given in Table 6.3. Since the inconsistency resolution procedure only utilised negative types of evidence, in the table we only give the changes of precision. Since in our use case the conflicts occurred because of incorrect data, applying high-precision matching

algorithms to the original datasets resulted in very small numbers of conflicts. Thus, the results obtained in such experiments (rows 1, 2, 4, 7) only illustrate common cases rather than provide reliable quantitative evaluation. As was expected, the algorithm's performance was better in "trivial" cases when the wrong statement had an a priori support significantly lower than the statements with which it was in conflict. This was the most frequent pattern in the experiments with low-precision matching algorithms (rows 5, 8) and artificially distorted datasets (rows 3, 6, 9). In more complex cases, if the conflict set contained a correct statement with a lower support than the actual wrong statement, the algorithm was still able to resolve the conflict correctly if additional evidence was available. One typical cause of such a conflict was the situation mentioned in section 5.2.1 when the same authors first presented a paper in a conference and after that published its extended version in a journal. For instance, a belief network built for such a case is shown in Fig. 6.2a. The individual *Ind1* represents a journal article published in 2006, and the individual *Ind2* represents a conference paper published in 2005. The Jaro-Winkler coreference resolution algorithm, which achieved a precision of 0.92, identified these papers as potentially equal, so we added an *owl:sameAs* link between *Ind1* and *Ind2*. This link allows two new statements to be inferred: *InProc(Ind1)* stating that *Ind1* is a paper in proceedings and *year(Ind1, 2005)* stating that its publication year is 2005. Thus, the *owl:sameAs* mapping leads to the violation of two ontological restrictions: disjointness between journal and conference articles. While each conflict separately would be resolved by removing the assertions related to *Ind2* (*In_Proc(Ind2)* and *year(Ind2, 2005)* in the Fig. 6.2a, because they have lower initial support than *Ind1=Ind2*), cumulative evidence allowed the algorithm to recognise the actual incorrect *sameAs*

link (*Ind1=Ind2*). A similar situation occurred when one instance (*Ind1*) was considered similar to two others (*Ind2* and *Ind3*), but only one of the *sameAs* links (e.g., *Ind1=Ind2*) led to the inconsistency (e.g., disjoint axiom violation). In that case the existence of the correct *sameAs* link (*Ind1=Ind3*) increased the support of the corresponding class assertion and again caused the wrong link (*Ind1=Ind2*) to be removed. However, in cases where the incorrect statement was considered a priori more reliable than the conflicting ones and the evidence was not sufficient, the algorithm made a mistake. For instance, the conflict in Fig. 6.2a was resolved wrongly when the dataset containing *Ind2* was artificially distorted. Although the statements involved in the conflict were not affected by the distortion, the initial support of the *Ind2* assertions was significantly lower (0.51 instead of 0.9), which was insufficient to break the *sameAs* link.

The capabilities of the Dempster-Shafer representation were important in cases where the a priori support of some statements was low. For instance, using L2 Jaro-Winkler similarity for the EPrints/DBLP datasets achieved a very low precision (0.39). In such cases the plausibility allows us to distinguish the cases where a statement is considered unreliable because of insufficient evidence from those where there is sufficient evidence against it. For instance, Fig. 6.2b shows such a case. A record in the EPrints dataset describing a conference paper was linked to two different papers in the DBLP dataset. One of the links was incorrect. After belief propagation the support values of both links were still below 0.5. However, the evidence against the incorrect link was significantly stronger, so its plausibility was low (0.02) while the plausibility of the correct link remained high (0.99).

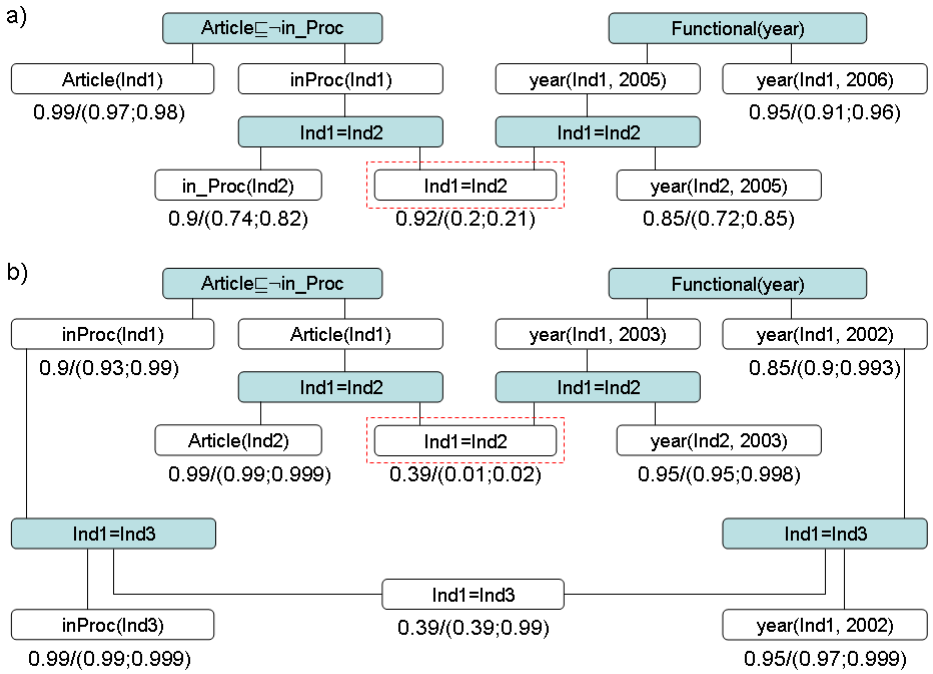

Figure 6.2: Examples of belief networks constructed during evaluation. The numbers show the support before propagation and (in brackets) support and plausibility after propagation.
a) An incorrect *sameAs* mapping *Ind1=Ind2* violates two restrictions: disjointness between journal and conference articles (left part of the graph) and functionality of the publication year property (right).
b) Influence of the Dempster-Shafer plausibility: correct *sameAs* relation *Ind1=Ind3* (bottom) has low support but high plausibility because it does not contribute to the inconsistency while an incorrect one (*Ind1=Ind2*) with the same initial support receives both low support and low plausibility after propagation.

Table 6.4: Test results: coreference refinement

Dataset	No	Matching algorithm	*sweto:Publication*					
			Before			After		
			Precision	Recall	F1	Precision	Recall	F1
EPrints/Rexa	1	Jaro-Winkler	0.950	0.833	0.887	0.969	0.832	0.895
	2	L2 Jaro-Winkler	0.879	0.956	0.916	0.923	0.956	0.939
EPrints/DBLP	3	Jaro-Winkler	0.922	0.952	0.937	0.992	0.952	0.971
	4	L2 Jaro-Winkler	0.389	0.984	0.558	0.838	0.983	0.905
Rexa/DBLP	5	Jaro-Winkler	0.899	0.933	0.916	0.944	0.932	0.938
	6	L2 Jaro-Winkler	0.546	0.982	0.702	0.823	0.981	0.895
Cora(I)	7	Monge-Elkan	0.735	0.931	0.821	0.939	0.836	0.884
Cora(II)	8	Monge-Elkan	0.698	0.986	0.817	0.958	0.956	0.957
			foaf:Person					
EPrints/Rexa	9	L2 Jaro-Winkler	0.738	0.888	0.806	0.788	0.935	0.855
EPrints/DBLP	10	L2 Jaro-Winkler	0.532	0.746	0.621	0.583	0.921	0.714
Rexa/DBLP	11	Jaro-Winkler	0.965	0.755	0.846	0.968	0.876	0.920
Cora(I)	12	L2 Jaro-Winkler	0.983	0.879	0.928	0.981	0.895	0.936
Cora(II)	13	L2 Jaro-Winkler	0.999	0.994	0.997	0.999	0.994	0.997

6.3.3 Experimental results: coreference refinement

In these tests we measured the quality of coreference resolution before and after belief propagation. The results of the tests are shown in the Table 6.4. As expected, in almost all cases the refinement procedure led to an improvement in overall performance expressed by the F1-measure. For *sweto:Publication* instances (rows 1, 2, 4, 6, 7, 8) the recall has decreased: the algorithm incorrectly resolved some inconsistencies, which in fact occurred due to incorrect data statements. The decrease was slight for AKT/Rexa/DBLP datasets and more significant for Cora where the degree of noise was higher. However, in all cases this decrease was accompanied by an increase in precision. For *foaf:Person* individuals, the effect of belief propagation primarily influenced recall: links between instances reinforced the potential mappings, which would otherwise be rejected. Because Cora(II) contained less formatting errors than Cora(I), there were very few "dubious" mappings produced during initial coreferencing, and belief propagation was not able to catch them.

Considering our results in context, we can compare the results obtained for Cora (I) and Cora(II) datasets with the results reported by the authors of other coreference

resolution tools. We found that our resulting F1 measure obtained for Cora(I) publications (row 7) was higher than the results reported in [8] (0.867) and [90] (0.87), but lower than in [97] (0.93). The main cause for the difference with the latter case was the limited set of properties used: in order to minimise the number of attributes processed by basic coreferencing methods, we only used the title comparison for determining candidate individuals in our tests. When we trained the algorithm described in [97] using only the *title*, *year*, and *venue* attributes, it achieved the F1 measure value of 0.88.

For Cora(II), the F-measure was similar to that reported for [32]: slightly higher for publications (0.957 vs 0.954) while slightly lower for people (0.997 vs 0.999). The difference is due to the fact that the authors of [32] used better similarity measures for the initial coreference resolution stage (reported precision for publications 0.985 without exploiting links), while exploiting data uncertainty in our approach increased recall by allowing the impact of different types of contradictory statements to be captured: e.g., if two papers were considered identical, but had different publication years, this inconsistency was not sufficient to break the mapping if there was agreement on the venue name and pages.

Although there are several other publications where tests with the Cora dataset were reported, we found that their results are not comparable with ours. This concerns the algorithms proposed in [111] and [17], and [101], which used different versions of the Cora dataset where, in particular, more mappings were removed from the gold standard so that the dataset contained 132 clusters [111] rather than 125 in Cora(II), and papers with the same title and year were considered identical [101].

However, the main advantage of our technique in comparison with existing techniques is the ability to reason about uncertainty degrees of coreference mappings in combination with the uncertainty values attached to data statements. This capability could not be fully utilised in the tests with the Cora(I) and Cora(II) datasets, because the provenance of data statements was not preserved, and we we could not distinguish between "more reliable" and "less reliable" data statements. However, this capability was valuable for the AKT-Rexa-DBLP testbed where the knowledge about relative reliability of different data sources allowed the majority of conflicts to be resolved correctly (Table 6.3 and rows 1-6 of Table 6.4).

6.4 Summary

Experiments described in this chapter consisted of two parts. First, we wanted to check the usefulness of one of the aspects of the architecture: using the class hierarchy to configure problem-solving methods.

When designing the architecture our intention was to make flexible method selection and configuration possible. Given that even within the same domain, depending on the type of data, different methods can be optimal and optimal parameters for the same method may vary, we needed to allow fine-grained configuration of methods to be defined. This was achieved by specifying application contexts as adapters between a method and an application domain and linking methods' selection and configuration parameters to corresponding application contexts (chapter 3). However, the need to define optimal parameters for each method and type of data for each dataset, to

which the method is applied, would make the system too complex and require potentially excessive human effort. In order to reduce this effort, the architecture exploits axioms defined in the domain ontologies, in particular the class hierarchy defined by the *rdfs:subClassOf* axioms. Our tests have shown (i) how the class hierarchy can be used in combination with machine-learning techniques to automatically determine optimal parameters for coreferencing resolution method applied to instances of different classes and (ii) how it helps to find an optimal trade-off between specificity of the data and scarcity of training examples. In particular, this has allowed the methods to produce more robust decision models by combining training instances from different subclasses. In this way, these findings contribute to answering the research question 3: "How can we exploit axioms defined in domain ontologies to improve the performance of fusion algorithms?".

Moreover, we performed tests to examine the capabilities of our belief propagation algorithm exploiting the Dempster-Shafer uncertainty representation and the valuation networks framework for uncertainty propagation. In Chapter 4, we proposed to use the Dempster-Shafer formalism because it was more appropriate for the specific needs of the knowledge fusion task than popular alternative options. Our tests have shown how specific representation capabilities of the Dempster-Shafer approach were useful in the conflict resolution task (see subsection 6.3.2). In particular, using two belief values (support and plausibility) to express the confidence degree of a statement helped to distinguish between insufficient positive evidence and strong negative evidence and thus be able to judge about correctness of statements with low initial support (e.g., example in Fig. 6.2b). These results justified our choice of the formalism as an answer to question 4: "What kind of uncertainty management framework

is suitable for fusion?"

The tests have shown that the belief propagation algorithm was able to resolve correctly the majority of conflicts resulting from coreferencing errors and spurious data (subsection 6.3.2) and improved the quality of the set of coreference mappings produced by the architecture (subsection 6.3.3). Thus, the algorithm provides a solution to research question 5: "How can we exploit uncertainty and provenance to improve the fusion performance?" Our approach exploits uncertainty and provenance in combination with ontological knowledge and mutual interdependencies between data statements. One feature of the approach is the possibility to reason about uncertain data and uncertain coreference resolution results in combination. This feature justifies our choice to represent uncertainty at the architecture level and express the estimated reliability of methods together with their configuration parameters.

Chapter 7

Fusion in a multi-ontology environment

In the previous chapters we discussed the problem of semantic data integration assuming that the data to be integrated are structured according to a single shared ontology. This assumption holds in some scenarios, like corporate knowledge management, but is not valid on the Web scale. So, in this chapter we target the case where the knowledge bases are structured according to different ontologies. First, we describe our extension of the KnoFuss architecture aimed at incorporating ontology matching techniques into the fusion workflow and the tests we performed with existing schema-matching tools. Then, we present our schema matching approach aimed at enhancing the data-level coreference resolution in a special case of a multi-ontology environment: the network of Linked Data repositories.

This chapter is based on the following publications:

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2009). Towards data fusion in a multi-ontology environment. Workshop: Linked Data on the

Web (LDOW 2009), 18th International World Wide Web Conference (WWW 2009), Madrid, Spain.

- Andriy Nikolov, Victoria Uren, Enrico Motta, Anne de Roeck (2009). Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. 4th Asian Semantic Web Conference (ASWC 2009), Shanghai, China.

7.1 Introduction

As was discussed before (section 2.2), the data integration process has to deal with two top-level problems: resolving the schema-level and the data-level conflicts. Our work has focused on the data-level issues, and the algorithms discussed in previous chapters assumed that schema-level problems had already been resolved and the knowledge bases to be integrated were structured according to the same ontology. A significant amount of research has been carried out in the ontology matching area, and many algorithms have been produced to resolve the schema-level integration problem automatically (see section 2.4). On the Web at large, in particular, in the Linked Data environment, the semantic heterogeneity of data is inevitable, which makes it necessary for a data integration system to use results of automatic ontology matching techniques. Although the usage of common schema ontologies such as FOAF, SKOS, or Dublin Core is encouraged [10], existing datasets often employ their own schemas or, in other cases, terms of common ontologies do not provide full information about the data they describe and many important usage patterns remain implicit: e.g., in DBLP[1] a generic *foaf:Person* class in fact refers only to people related to computer science.

These techniques do not guarantee 100% accuracy, and errors produced by them may influence the quality of the data fusion stage. In this chapter we consider two complementary approaches aimed at performing fusion in the presence of schema heterogeneity that we have investigated.

The first approach extends the KnoFuss architecture making it possible to employ automatic schema matching algorithms to process knowledge bases structured using

[1] http://www4.wiwiss.fu-berlin.de/dblp/

different ontologies. This extension was implemented for the following reasons:

- To enable the usage of the KnoFuss architecture in a multi-ontology environment.
- To discover specific issues arising in multi-ontology data fusion in comparison with the single-ontology case.
- To test the possible impact caused by inaccuracies introduced at the schema-matching stage on the data-level fusion process.
- To research ways of reducing the impact of ontological heterogeneity.

As our second contribution, we propose an approach which performs schema matching in order to improve instance coreference resolution. A novel feature of the approach is its use of existing data-level coreference links defined in third-party Linked Data repositories as background knowledge for schema matching techniques. Using our algorithm, we were able find a substantial number of new coreference links well beyond the set of original links which are published in Linked Data repositories.

The chapter is structured in the following way. In the first part of the chapter we discuss additional challenges to the fusion process caused by schema heterogeneity and describe our extension of the KnoFuss architecture aimed at dealing with it. In section 7.2 we discuss how the problem of data integration in the presence of schema heterogeneity is traditionally viewed and what kinds of schema mismatches are described in the literature. In section 7.3 we discuss how these schema-level mismatches influence the instance-level integration task. In section 7.4 we describe how we extend the KnoFuss architecture presented in Chapter 3 to process data repositories structured using different ontologies. Then, in section 7.5 we describe the

experiments we conducted with the extended architecture using third-party ontology matching tools as methods.

In the second part of the chapter we describe our proposed schema-matching approach aimed at facilitating instance-level coreference resolution between Linked Data repositories. Section 7.6 provides an overview of the schema heterogeneity issues in the Linked Data environment and discusses how third-party repositories can be utilised as background knowledge to resolve them. In sections 7.7 and 7.8 we describe the stages of our algorithm in more detail, and in section 7.9 we describe the experiments we conducted to validate our approach. Section 7.10 concludes the chapter.

7.2 Related work: ontology mismatches classification

The situation where the source and the target knowledge bases use different ontologies makes it impossible for data integration methods to use the semantic data structure: information in a knowledge base which uses an unknown ontology becomes meaningless. Only a limited set of methods, which do not use data structures, can be applied directly to such datasets. These include, for instance, object identification methods that originated in the natural language processing domain, which compare word sets using set similarity measures. Most of the methods, including those considered in our experiments, however, require a uniform view over data in both knowledge bases.

The problem of providing such a view originated in the database domain under the term of *data integration*: combining data residing in different sources, and providing

the user with a unified view of this data [71]. A typical data integration architecture (e.g., assumed in [71], [81]) includes a global reference schema and a set of sources, each structured according to its specific local schema. The global schema is connected to each local schema via a set of mappings: correspondences which allow query reformulation. There are two main approaches to mapping specification: Global-as-View (GAV) and Local-as-View (LAV). In the GAV approach each construct in the global schema is mapped to a view (query) over local schema constructs. The LAV approach, in contrast, specifies each local schema construct as a query over global schema constructs. Then, an inferencing mechanism is used to re-define atoms in the global schema according to each local schema's terminology. Thus, the GAV approach is more efficient with regard to query processing: no logical inferencing is needed, and a query specified in terms of the global schema is reformulated by replacing the global schema terms with appropriate mapping views. On the other hand, it is more demanding from the mapping construction point of view: the mapping views contain references to all relevant sources and have to be changed each time a source is changed.

Obtaining an adequate representation of mappings which allows correct data transformation is a non-trivial problem due to ontology mismatches. A classification framework of different types of mismatches between overlapping ontologies is given in [67]. Assuming that ontologies are represented in the same language, the framework distinguishes:

- *Conceptualisation mismatches*, which concern:
  - *Scope*, when two classes seemingly representing the same concept do not

contain the same instances (e.g., the class *PoliticalOrganization* in TAP ontology includes terrorist groups, while in SWETO it is meant to represent only legal organisations).

- *Model coverage and granularity*, when parts of the domain in one ontology are not covered in another or covered with a different level of detail (e.g., in SWETO the class *Company* does not have subclasses while TAP and DBPedia 3.2 distinguish between different types of companies).

- *Explication mismatches*, which are divided into:

  - *Modelling style mismatches*, when the same domain is modeled using different paradigms (e.g., point vs interval logic for time representation) or concept specification (e.g., splitting the subclasses of the same class in a hierarchy according to different criteria).

  - *Terminological mismatches*, when different terms are used to represent the same entity (synonymy) or the same term represents different entities (homonymy).

  - *Encoding mismatches*, when the values at the data level have different formats. This one has to be dealt at the data-level stage, so we do not consider it in this chapter.

To represent the correspondences between ontologies correctly and overcome these mismatches, mappings of varying degrees of complexity are required. In [102] common correspondence patterns are introduced to represent such mappings (see Fig. 7.1). For the most part, mapping patterns represent description logic relations. However, current automatic ontology matching algorithms can only deal with a subset of the

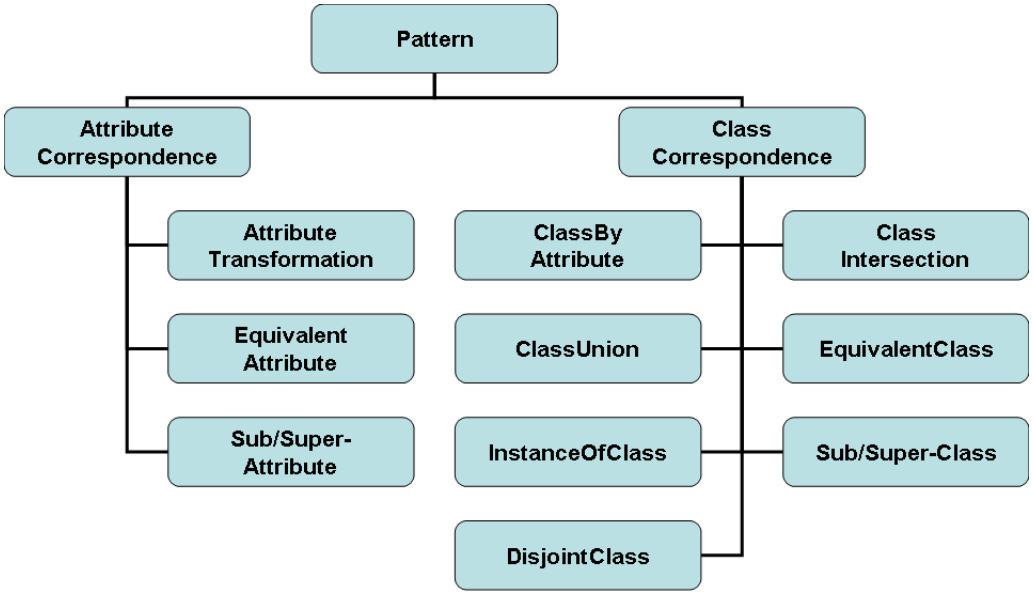

Figure 7.1: Correspondence patterns of ontology matching according to [102] (fragment). A commonly used *DisjointClass* pattern is included.

various types of mismatches that may occur (primarily terminological). Indeed, most of them are limited to one-to-one mappings [38]. Conceptualisation mismatches can be partially represented using the *Sub/Super-Class* mappings instead of *Equivalent-Class*. Some types of mismatches such as using different point vs interval logic time representation paradigms cannot be handled automatically by any existing tool, to our knowledge.

Given the limited capabilities of the ontology matching tools, we can expect that some of the ontology mismatches will remain unresolved or partially unresolved at the data integration stage. In the next section we discuss the impact of such mismatches during the data integration process.

7.3 Data-level impact of ontology mismatches

The first type of mismatches in the classification presented in [67] concerns conceptualisation mismatches. For the coreference resolution stage, shared conceptualisation allows the system to:

- consider individuals belonging to the same class as candidates for matching;
- estimate the likelihood of individuals being equivalent given available evidence (e.g., having two people with the same name belonging to a specific class *SemanticWebResearcher* is much stronger evidence of equivalence than if they only had a generic class *Person* in common).

The conceptualisation mismatches between two ontologies (in particular, scope mismatches) may reduce both recall and precision of coreference resolution algorithms. For example, the class *Company* in SWETO does not include financial organisations while its counterpart in TAP includes them. Thus, when the system tries to find, for each company in TAP, coreferent individuals in SWETO having only the equivalence relation between these classes, it will not find matching pairs for financial organisations because they belong to a different class in SWETO. This factor decreases recall. On the other hand, the class *ComputerScientist* in TAP contains only world-famous computer scientists while most researchers are classified according to their place of work (e.g., *CMUPerson*, *W3CPerson*). *ComputerScienceResearcher* in SWETO, which automatic tools often consider equivalent to *tap:ComputerScientist*, has much wider coverage and includes everybody who contributed to a computer science paper mentioned in the knowledge base. On the whole, labels in SWETO are much more ambiguous than in TAP, and the danger of matching two unrelated

individuals increases, which may affect precision negatively. The same happens when there is no equivalence between classes but a *Sub/Super-Class* relation: the same degree of similarity between individuals may provide much weaker evidence, which makes it hard to estimate adequately the reliability of methods' output. Another area of impact involves disjointness relations. Disjointness between classes can be used as evidence to consider some coreference mappings incorrect and delete them. The scope mismatches can lead to errors when classes considered disjoint in one ontology are overlapping in another one (like in the case with *PoliticalOrganization* and *TerroristOrganization* above): correct mappings can be deleted if they are perceived as causing inconsistency. The granularity mismatches do not allow ontological constraints defined for classes at the lower levels of the hierarchy to be used if the other ontology does not distinguish between these classes.

Among the explication mismatches, modelling style differences are the hardest to solve automatically. Translation between paradigms is very much a domain-specific problem, and common correspondence patterns are often not sufficient to align two ontologies. In a simple example, if one ontology represents colours using a set of pre-defined labels (red, yellow, black) and another one uses RGB encoding, it is very hard to find similar values automatically: a hand-tailored matching procedure is necessary. To our knowledge, no existing automatic ontology matching tool is capable of dealing with different paradigms. For the case where subclasses of the same class in two ontologies are split according to different criteria, no useful DL relations can be established between them (apart from the fact that there may be some overlap). Such differences can make any automatic data integration procedures intractable. If these mismatches occur at the lower levels of the hierarchy, methods can operate

only with information defined at a higher level. In the KnoFuss architecture terms it means using the application contexts defined for the generic classes (section 3.6.3).

Terminological mismatches are the primary focus of most existing ontology matching tools [38], which makes them the simplest to handle. They can be solved by creating *EquivalentClass* and *EquivalentAttribute* correspondences.

Finally, encoding problems are not specific to the schema level and may occur even if knowledge bases share the same ontology. These mismatches cause coreference resolution problems as mentioned in the previous chapters. Thus, such problems do not require additional attention when moving from single-ontology to multi-ontology semantic data fusion.

7.4 Extending the KnoFuss architecture

The core KnoFuss architecture described in Chapter 3 focuses on the second stage of the fusion process: knowledge base integration. The *ontology integration* step was divided into three subtasks, which were assumed to be performed outside the scope of the architecture: *preprocessing*, *ontology matching* and *instance transformation* (see Fig. 3.1).

When extending the functionality of the architecture to cover the ontology integration stage we still assume that the knowledge bases to be integrated are represented in the same common language (RDFS or OWL), which makes the *preprocessing* step unnecessary. The two remaining subtasks, however, must be covered by the framework (Fig. 7.2) and are explained in the following subsections.

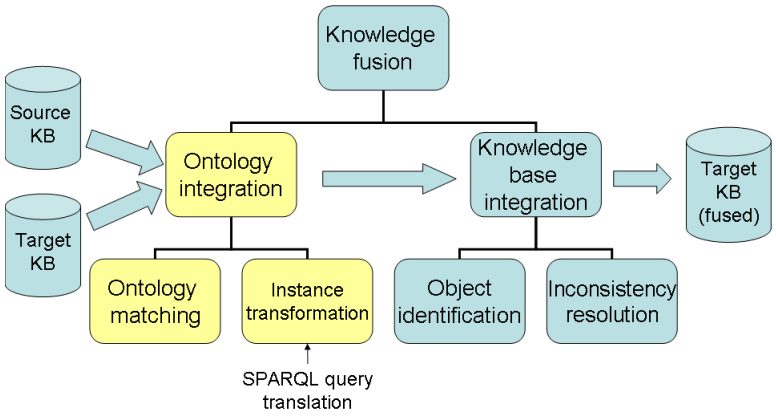

Figure 7.2: Fusion task decomposition incorporating schema matching.

7.4.1 Ontology matching

The *ontology matching* task involves the creation of mapping rules or alignments: sets of correspondences between two ontologies [38]. The choice to make here concerns the suitable mechanism for applying mappings (Global-as-View vs Local-as-View). In our scenario, approximate mappings are produced automatically, which makes the GAV approach preferable (no manual construction of mappings is required). In KnoFuss terms, the target ontology serves as the global schema and the source ontology as the local one in each fusion session. Thus, the system needs mappings to represent each single term in the target ontology in the source ontology terms. The descriptor of the ontology matching task is given in Table 7.1.

Considering correspondence patterns, data fusion needs both correspondences between concepts (*ClassCorrespondence*) and correspondences between properties (*Attribute-Correspondence*). Class mappings allow the relevant method application contexts to be translated into the terms of the source ontology if they were initially defined in terms of the target ontology. The attribute correspondences are needed in order to retrieve properties relevant for coreference resolution in both knowledge

Table 7.1: Ontology matching task descriptor

Task	Ontology matching
Inputs	*SourceOntology* :type *Ontology*; *TargetOntology* :type *Ontology*;
Outputs	*Alignment* :type list of *MergeSet* - Set of possible equivalence, subsumption and disjointness mappings between terms of source and target knowledge base schema ontologies

bases. The equivalence and subsumption relations allow relevant concepts and properties in the source ontology to be found. The disjointness relations between concepts are usable for the *knowledge base updating* stage, providing evidence for inconsistency resolution. The architecture assumes that the ontology matching methods provide their output in the standard Alignment API format [37]. After the results of the ontology matching methods are obtained, the system utilises the *DisjointClass* mappings. The system uses a simple algorithm to search for contradictory mappings: it finds situations where two classes in different ontologies are connected via a *Sub/Super-Class* mapping (created by ontology matching methods or inferred) and at the same time are disjoint (again, directly or via inference). Such mappings are considered conflicting. If the *DisjointClass* mapping has a higher confidence value, then the contradictory *Sub/Super-Class* mapping (or the mapping from which it was inferred) is removed from consideration.

7.4.2 Instance transformation

After the schema-level mappings are obtained, the next stage is *instance transformation*. At this stage the mappings are applied in order to allow the methods for the *coreference resolution* and *knowledge base updating* stages to process the data from

the source and the target repositories in a uniform way, as if they were structured using the same ontology. In the KnoFuss architecture SPARQL queries are used as a primary means of retrieving data: method applicability ranges, application contexts, sets of relevant attributes are expressed in SPARQL. For example, in order to check whether two individuals, one from the source knowledge base and one from the target one, are coreferent, a coreference resolution method needs to retrieve for each individual a set of relevant property values to compare (e.g., *rdfs:label* and *sweto:year* for scientific publications in our experiments in Chapter 6). If both knowledge bases share the same schema, this can be done by posting the same selection query to both the source and the target knowledge bases. However, if the source ontology is different from the target one, then the original query defined in terms of the target ontology has to be reformulated to be applied to the source knowledge base. The queries are translated into the terms of the source ontology using the available schema-level mappings. In a simple case, if all mappings represent equivalence relations, and we only have one-to-one mappings, this is done by replacing the target ontology terms mentioned in the WHERE clause of a query with equivalent terms from the source ontology. For example, if we use TAP as the source knowledge base and SWETO testbed as the target one, and we have a method which is applicable to individuals of the class *sweto:City*, the corresponding selection query will be the following:

SELECT *?uri* **WHERE** {

?uri rdf:type sweto:City }

In a trivial case, if we have an *EquivalentClass* mapping with the class *tap:City*, the query to the source ontology will be translated as

SELECT *?uri* **WHERE** {

?uri rdf:type tap:City }

Sometimes a term in the target ontology may correspond to several terms in the source ontology, and the mappings may not cover existing schema mismatches adequately. This happens when there are several candidate *EquivalentClass* mappings provided by one or several ontology matching tools. In such situations we combine these mappings and consider them as a single *ClassUnion* mapping. For instance, when we consider the query

SELECT *?uri* **WHERE** {

?uri rdf:type sweto:Computer_Science_Researcher }

the system tries to find all *ClassCorrespondence* mappings which include the class *sweto:Computer_Science_Researcher*. In our example with the CIDER tool (see below) these included *EquivalentClass* mappings with the classes *tap: CMUPerson*, *tap:ComputerScientist* and *tap:MedicalScientist*.

Such a variety of potentially corresponding classes is caused by several existing mismatches between ontologies, in particular, terminological mismatches (*Computer_Science_Researcher* vs *ComputerScientist*), modelling style mismatches (*tap: CMUPerson* includes computer science researchers who worked in the CMU), and conceptualisation scope mismatches (*tap: ComputerScientist* represents only a subset of "world-famous" researchers and *tap:Medical-Scientist* includes authors of medical AI expert systems). From the strictly logical point of view, the only correct mapping would be a

Sub-Super-Class mapping *tap:ComputerScientist* $\subseteq$ *sweto: Computer_Science_Researcher*. However, excluding other mappings would remove from consideration many TAP individuals which have their equivalent SWETO counterparts. In reality, rather than strict logical relations, the data integration system needs information about partial alignments between concepts to select individuals which may potentially be coreferent. We can call this the *OverlapClass* correspondence pattern.[2]

Thus, the query from our example is translated into:

SELECT *?uri* **WHERE**

{ {*?uri rdf:type tap:CMUPerson*}

UNION {*?uri rdf:type tap:Computer_Scientist*}

UNION {*?uri rdf:type tap:Medical_Scientist*}}

These pairs of queries assumed to be equivalent are then available at the later stages of the workflow, which allows the system to operate in the same way as in a single ontology case.

[2]In existing ontologies an example of a relation implementing this pattern is *umbel:isAligned* defined in the UMBEL ontology (http://www.umbel.org/). However, by definition, it is only applicable to internally defined UMBEL classes (*umbel:SubjectConcept*).

7.5 Experiments with schema-level ontology matching techniques

To test the KnoFuss architecture in a multi-ontology scenario, we used three public knowledge bases:

- TAP [52]
- SWETO testbed [3]
- DBPedia 3.2[3]

The DBPedia dataset was automatically extracted from Wikipedia articles and, in version 3.2, was structured according to a specially designed ontology. All three knowledge bases contain instances from several topics, and their schema ontologies are shallow and cross-domain. We selected several overlapping domains of these ontologies such as people, organisations, and geographic entities (Fig. 7.3, 7.4, 7.5), and considered individuals belonging to classes describing these domains.

For the ontology schema matching, we used two third-party tools: CIDER [50] and Lily [121]. Both have participated in the OAEI 2008 ontology alignment contest. The functionality of CIDER includes a term similarity service: calculating similarity metrics for a pair of concepts belonging to two different ontologies. We used this service to calculate similarities between all pairs of concepts and then selected mappings which exceeded the chosen threshold value. Lily was used as a "black box" tool: two ontologies were provided as its input, the tool produced potential mappings, and we filtered them further using a threshold. Thus, the alignment extraction stage [38] was

[3] www.dbpedia.org

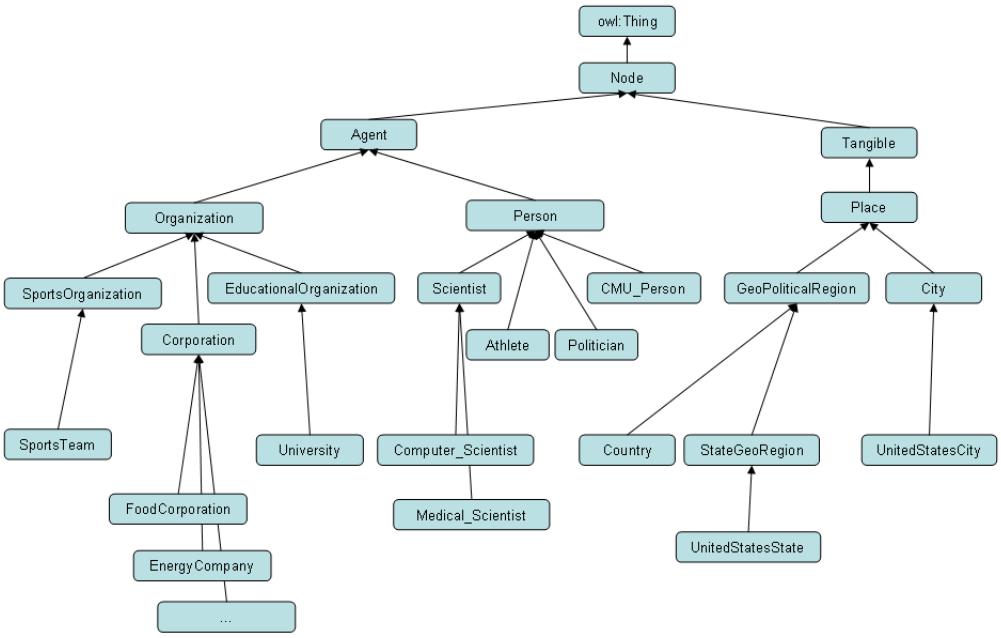

Figure 7.3: Class hierarchy of the relevant subset of the TAP dataset.

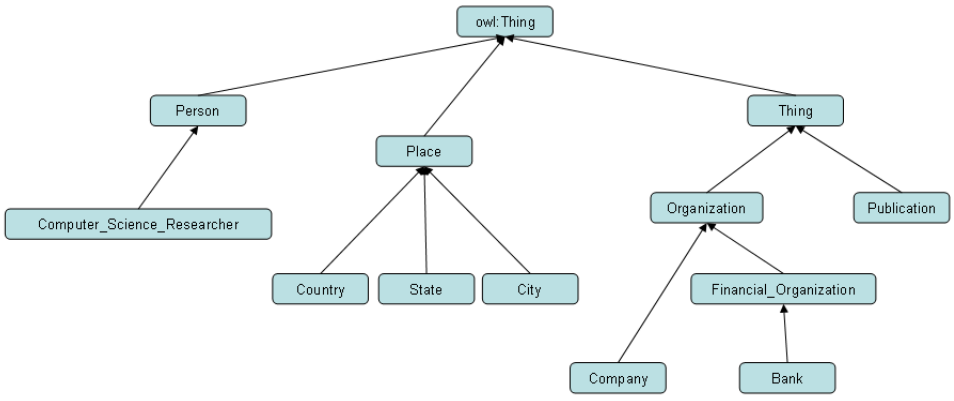

Figure 7.4: Class hierarchy of the relevant subset of the SWETO dataset.

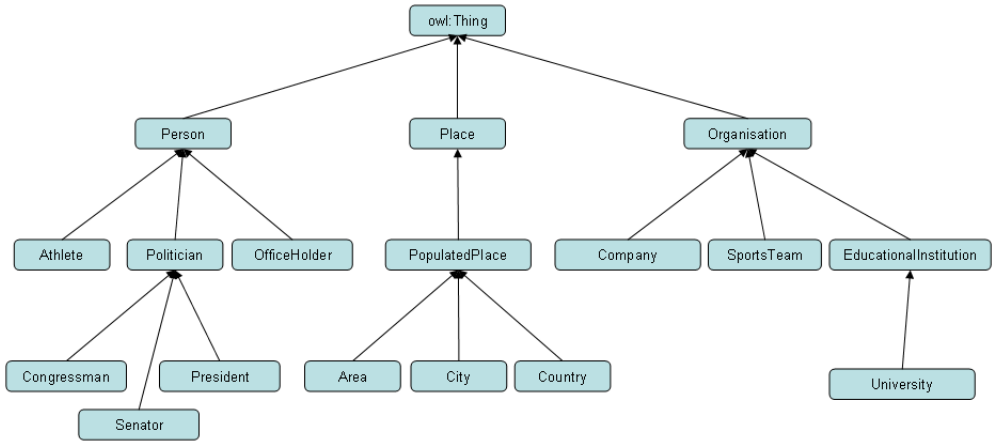

Figure 7.5: Class hierarchy of the relevant subset of the DBPedia dataset.

performed internally by Lily. The most important consequence of this was that Lily applied a special filter ensuring that all correspondences are one-to-one, while from CIDER we received a set of possible mappings for each class. In addition to those, we used SCARLET [100] as a method for generating *DisjointClass* mappings. The SCARLET service takes as its input two terms and tries to find relations between them using other publicly available ontologies as background knowledge. Disjointness was used to filter out conflicting equivalence relations with a low reliability. As coreference resolution methods for instances, we used the same string similarity techniques as in our single ontology scenario experiments (Jaro-Winkler and L2 Jaro-Winkler).

7.5.1 Test results

The test results are given in tables 7.2, 7.3 and 7.4. For each chosen class in the target knowledge base (column 2) we performed instance coreference resolution trying to find equivalent instances in the source knowledge base. For each class we performed three tests using the same instance coreference resolution method (Jaro-Winkler or L2 Jaro-Winkler), but three different schema alignments: one generated by CIDER,

Table 7.2: Test results for instance coreferencing (TAP vs SWETO)

N	Class (Target ontology)	Schema matcher	Instance matcher	Prec.	Rec.	F1	aligned concepts
1	Person	actual	L2 JW	0.29	0.92	0.44	Person
		CIDER		-"-	-"-	-"-	-"-
		Lily		-"-	-"-	-"-	-"-
2	Computer_Science_ Researcher	actual	L2 JW	0.62	0.93	0.75	Computer_Scientist, Medical_Scientist CMU_Person
		CIDER		0.62	0.93	0.75	-"-
		Lily	N/A				Computer_Science_Research_Project
3	Organization	actual	JW	0.80	0.56	0.66	Organization
		CIDER		-"-	-"-	-"-	-"-
		Lily		-"-	-"-	-"-	-"-
4	Company	actual	JW	0.76	0.66	0.71	Corporation
		CIDER		0.82	0.56	0.66	Corporation, Country, Conifer, ClothingBrand, ComputerScientist
		Lily		1	0.002	0.003	SoftwareCompany
5	City	actual	JW	0.87	0.93	0.90	City
		CIDER		0.87	0.93	0.90	City, Territory
		Lily		0.87	0.93	0.90	City
6	State	actual	JW	0.98	0.98	0.98	StateGeoRegion
		CIDER		0.98	0.98	0.98	UnitedStatesState, UnitedStatesCity
		Lily	N/A				
7	Country	actual	JW	0.95	0.92	0.93	Country
		CIDER		0.87	0.88	0.88	Country, Continent, Corporation
		Lily		0.95	0.92	0.93	Country
8	Place	actual	JW	0.79	0.92	0.85	Place
		CIDER		0.79	0.92	0.85	Place, Plant, Employee
		Lily	N/A				Continent
Avg		actual		0.76	0.85	0.78	
		CIDER		0.76	0.84	0.76	
		Lily		0.49	0.42	0.37	

one produced by Lily, and the gold standard obtained manually. Moreover, we added the *DisjointClass* mappings produced by SCARLET to the alignments generated by CIDER and Lily. The precision, recall and F1 measures for instance coreference resolution in three cases are provided in columns 5, 6, and 7[4], and the list of source ontology classes aligned to the target ontology class is given in column 8.

As could be expected, errors during schema matching are propagated and can lead to significant distortions during instance coreference resolution. For instance, in many tests involving CIDER, one target concept was aligned with several source concepts, some of them irrelevant. This had a negative impact on precision when individuals from non-overlapping classes were compared and sometimes found identical

[4]The "-"-" sign means that the value is the same as in the previous row

Table 7.3: Test results for instance coreferencing (TAP vs DBPedia)

N	Class (Target ontology)	Schema matcher	Instance matcher	Prec.	Rec.	F1	aligned concepts
9	Company	actual	JW	0.91	0.63	0.75	Corporation
		CIDER		0.93	0.60	0.73	Organization
		Lily		1.00	0.002	0.003	EnergyCompany
10	University	actual	JW	0.98	0.86	0.92	University
		CIDER		-"-	-"-	-"-	-"-
		Lily		-"-	-"-	-"-	-"-
11	Organisation	actual	JW	0.98	0.76	0.86	Organization
		CIDER		0.98	0.76	0.86	Organization, Brand, Magazine
		Lily	N/A				
12	Athlete	actual	L2 JW	0.95	0.93	0.94	Athlete
		CIDER		-"-	-"-	-"-	-"-
		Lily		-"-	-"-	-"-	-"-
13	OfficeHolder	actual	L2 JW	0.92	0.95	0.93	Politician
		CIDER	N/A				OfficeProduct
		Lily	N/A				CMUPerson
14	Person	actual	L2 JW	0.78	0.93	0.85	Person
		CIDER		0.77	0.93	0.84	Person, OperaOrganization, Cartoon, Perennial, Fern
		Lily		0.78	0.93	0.85	Person
15	City	actual	JW	0.82	0.69	0.75	City
		CIDER		0.82	0.68	0.74	City, Country Continent, Cuisine
		Lily		0.82	0.69	0.75	City
16	Country	actual	JW	0.98	0.94	0.96	Country
		CIDER		0.90	0.94	0.92	City, Country, Continent Corporation, ComicStrip, Conifer
		Lily		0.98	0.94	0.96	Country
17	Area	actual	JW	1.00	1.00	1.00	StateGeoRegion
		CIDER	N/A				Garden
		Lily	N/A				
18	Place	actual	JW	0.48	0.86	0.61	Place
		CIDER		0.48	0.86	0.61	Place, CMUFace, CyberPlace
		Lily	N/A				
Avg		actual		0.88	0.86	0.68	
		CIDER		0.68	0.66	0.62	
		Lily		0.55	0.44	0.44	

Table 7.4: Test results for instance coreferencing (SWETO vs DBPedia)

N	Class (Target ontology)	Schema matcher	Instance matcher	Prec.	Rec.	F1	aligned concepts
19	Company	actual	JW	0.61	0.56	0.58	Company, Bank
		CIDER		0.33	0.23	0.27	Complex, Company, Country
		Lily		0.61	0.30	0.40	Company
20	Country	actual	JW	0.98	0.91	0.95	Country
		CIDER		0.83	0.91	0.86	City, Country, Conference, Company
		Lily	N/A				County
21	City	actual	JW	0.96	0.74	0.84	City
		CIDER		0.96	0.74	0.84	City, Country
		Lily		0.96	0.74	0.84	City
22	Area	actual	JW	0.83	0.84	0.83	State
		CIDER		0.17	0.84	0.28	Place
		Lily	N/A				
23	Place	actual	JW	0.47	0.81	0.60	Place
		CIDER		0.47	0.81	0.60	Place
		Lily		0.47	0.81	0.60	Place
Avg		actual		0.77	0.77	0.76	
		CIDER		0.55	0.71	0.57	
		Lily		0.41	0.37	0.44	

due to label similarity (e.g., some companies had names derived from country names). In other cases, recall dropped when an *EquivalentClass* mapping was produced instead of a *ClassUnion* mapping and some relevant concepts were omitted: e.g., in row 19 (Table 7.4) Lily considered the classes *sweto:Company* and *dbpedia:Company* as equivalent while *sweto:Bank* instances were omitted from consideration. In general, we can see that the one-to-one mapping restriction imposed by Lily improves the precision of the schema matching stage but leads to an "all-or-nothing" outcome at the instance level: either two precisely corresponding classes were found, or completely spurious pairs were produced, which did not allow instance coreferencing (e.g., *ComputerScienceResearcher* and *ComputerScienceResearchProject*). CIDER, in contrast, had lower schema matching precision but better recall and, usually, led to better performance at the data level: on average for all tests, the F1-measure dropped by 13% when using CIDER and by 41% when using Lily, in comparison with manual schema matching.

7.5.2 Discussion

As a means to repair such errors, ontological constraints are extremely valuable in the *Coreference resolution* task. Apart from the widely used *owl:FunctionalProperty* and *owl:InverseFunctionalProperty*, which allow non-ambiguous instance identification, restrictions providing negative evidence are also valuable for filtering out incorrect mappings. These constraints include disjointness and datatype properties with cardinality constraints. For example, knowing that *Company* is disjoint with *Country* (or inferring that) would repair the schema matching problem in rows 4, 7, 16, 19 and 20.

In the SWETO ontology, all individuals belong to only one class, thus making sibling classes disjoint. In the TAP/SWETO tests we used this to produce disjointness restrictions and filter out some spurious schema mappings produced by the CIDER tool. As a result, for these tests the average precision drop obtained when using CIDER was only 3% (from 78% to 75%). However, most ontologies do not define disjointness relations explicitly, and even involving background knowledge did not help to discover them: the SCARLET service in our tests discovered only one relevant *DisjointClass* mapping (between the concepts *University* and *Company*), which could be used for filtering out incorrect mappings for the class *dbpedia:Organisation*. Having a special high-level reference ontology accessible on the Web where these constraints are specified would be a significant source of information.

It is well recognised that label comparison cannot be considered sufficiently reliable evidence for coreference resolution. However, more complex algorithms utilizing context data (additional properties and links between individuals) can only be applied to datasets containing sufficiently overlapping data. It can be expected that many data integration tasks on the Web scale will only be able to rely on instance names and thus can only provide suggestions rather than generate *owl:sameAs* statements carrying strong implications.

Since errors are inevitable in automatic coreferencing, provenance information must be stored together with the produced coreference mappings, so that the user application can decide whether to rely on them or not. One possible way to do so is to extend the coreference bundles approach [60] to include for each URI a measure of the system's confidence in its inclusion into the set.

Although semantic heterogeneity (different meaning attached to similar resources)

is primarily a schema-level knowledge modelling issue, it can cause problems at the instance level as well. For instance, the TAP ontology contains a single individual "Coca-Cola" for the Coca-Cola company, while SWETO contains several individuals describing Coca-Cola branches in different countries. Clearly, these should not be mapped as equivalent.

Finally, the use of instance-based ontology matching techniques [38] can be particularly promising in the context of the data fusion task. Hence, in the rest of the chapter we will describe the schema matching approach we developed to assist data-level coreference resolution of Linked Data repositories, which employs instance-based ontology matching.

7.6 Facilitating coreference resolution in Linked Data repositories: an instance-based approach

Instance-based ontology matching techniques operate in a "bottom-up" way and infer schema-level mappings from the available instance coreference links. The advantage of instance-level methods is their ability to provide valuable insights into the contents and meaning of schema entities from the way they are used. This makes them suitable for the data fusion scenario, because here there is a need to capture the actual usage pattern of an ontological term rather than how it was intended to be used by an ontology designer. In particular, this capability would help the system to deal with conceptualisation mismatches: e.g., to recognise such relations as the one between *dbpedia:Company* and *sweto:Bank* from the example above. At the same time, the major obstacle for the use of instance-based ontology matching techniques is the

need to have initial instance-level coreference mappings in the first place. Given that in the data fusion scenario schema mappings are primarily needed to facilitate instance-level coreference resolution, this dependency can lead to a "chicken and egg" problem. However, in certain scenarios this problem can be resolved if a partial set of instance-level mappings is available or can be easily obtained (e.g., using primary key values). One such scenario concerns data fusion in the Linked Data environment, where partial sets of instance-level links are indeed available.

Our approach focuses on inferring schema-level mappings between ontologies employed in Linked Data repositories. It uses instance-based ontology matching where, in order to produce a mapping between two classes, the algorithm analyses the overlap between their sets of individuals. Below we give a short overview of existing schema matching approaches similar to ours.

Among the existing approaches, instances of classes and relations are often considered as features when computing similarity between schema-level entities: e.g., CIDER [50] and RiMOM [128]. One particularly interesting approach, which uses schema alignment and coreference resolution in combination, was introduced in the ILIADS system [119]. ILIADS focuses on the traditional ontology matching scenario, where two schemas have to be integrated, and performs schema-level and data-level matching in a loop, where newly obtained instance-level mappings are used to improve schema-level alignment and vice versa. This is similar to our approach where schema-level matching is performed to enhance the instance coreferencing process. However, unlike ILIADS, our approach was primarily motivated by the data-level integration scenario and exploits information from many sources rather than only two.

The second relevant category of schema-matching techniques are those which utilise external sources as background knowledge. An approach proposed in [2] performs matching of two ontologies by linking them to an external third one and then using semantic relations defined in the external ontology to infer mappings between entities of two original ontologies. The SCARLET tool [100], mentioned above, employs a set of external ontologies, which it searches and selects using the Watson ontology search server[5]. These approaches, however, only consider schema-level evidence from third-party sources, while our approach relies on instance-level information.

Thus, our algorithm implements the following novel features, which we consider our contribution:

- Use of data-level coreference links to and from individuals defined in third-party repositories as background knowledge for schema-level ontology matching.

- Producing schema-level mappings suited for the needs of the instance coreference resolution process. In particular, our algorithm produces fuzzy mappings representing degree of overlap between classes of different ontologies rather than strict equivalence or subsumption relations.

The mappings between classes are primarily needed to identify which subsets of two data repositories are likely to contain co-referring individuals, so that such subsets can be processed by a coreference resolution algorithm afterwards. Let us consider an example scenario, where deriving such mappings is problematic and hampers coreference resolution (Fig. 7.6). Both DBPedia and DBLP datasets contain individuals representing computer scientists. In many cases the same person is described in

[5]http://watson.kmi.open.ac.uk/WatsonWUI/

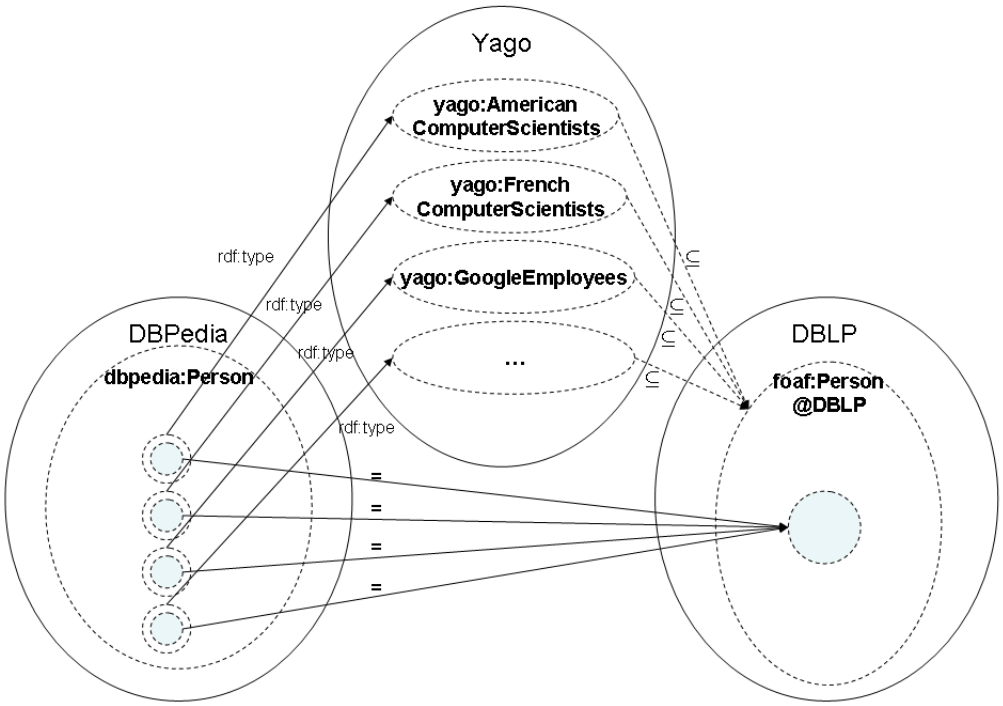

Figure 7.6: DBPedia and DBLP: exploiting schema-level links with third-party datasets. Solid arrows show existing *owl:sameAs* (=) and *rdf:type* links. Dashed arrows represent discovered schema relations. The system identifies the subset of *dbpedia:Person* instances, which overlaps with DBLP *foaf:Person* instances, as a union of classes defined in YAGO.

both repositories, but under different URIs. However, only a small proportion of possible coreference links between them is available[6]. More links can be discovered by performing automatic coreference resolution, but this task is complicated by two issues:

- Datasets do not contain overlapping properties for their individuals apart from personal names.

- Individuals which belong to overlapping subsets are not distinguished from others because of the conceptualisation scope mismatch between corresponding classes: in DBLP, all paper authors belong to the *foaf:Person* class, and in DBPedia the majority of computer scientists is assigned to a generic class *dbpedia:Person*. However, DBLP only contains information about computer scientists, while in DBPedia computer scientists represent only a small proportion of *dbpedia:Person* individuals. As a result, it becomes complicated to extract the subset of people in DBPedia which can potentially be represented in DBLP.

Applying name comparison for all *foaf:Person* and *dbpedia:Person* individuals is likely to produce many false positive results because of the ambiguity of personal names: DBPedia contains many non-computer science people which have the same names as DBLP paper authors. Before performing instance matching we need to narrow the context and exclude from comparison individuals which are unlikely to appear in both datasets. Since the actual schema ontologies used by the repositories, which have to be connected, are not sufficiently detailed, then evidence data defined in other data sources should be utilised.

[6]196 links in total in DBPedia 3.2 on 13/06/2009

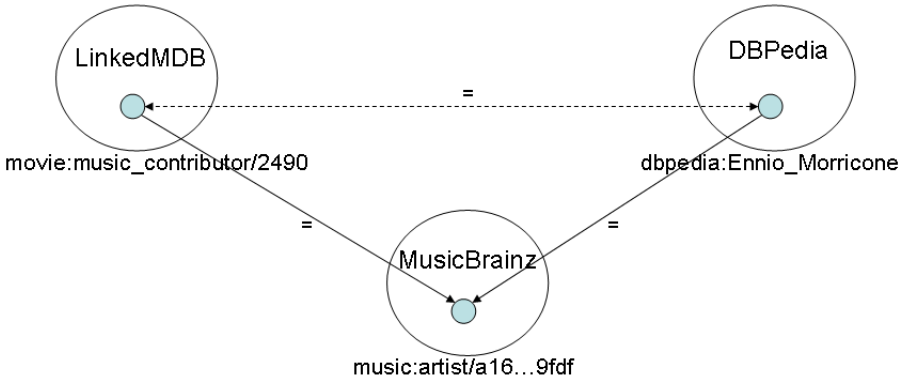

Figure 7.7: Obtaining *owl:sameAs* links by computing transitive closure

Instance-based ontology matching techniques are particularly suitable to infer schema-level mappings in the Linked Data environment because of the availability of data to exploit: Linked Data repositories are already connected by *owl:sameAs* relations between individuals, although these sets of relations are often incomplete. Sometimes this scenario is present in Linked Data repositories directly: for instance, in the example shown in Fig. 7.6 DBPedia individuals are structured by the DBPedia own ontology, but also have *rdf:type* links to the classes defined in the YAGO[7] and Umbel[8] ontologies. However, more often such sets can be constructed by clustering together individuals connected via existing *owl:sameAs* coreference links (see Fig. 7.7). Such transitive closure sets are likely to be incomplete because intermediate datasets may not contain all individuals from their neighbours or because some links on the path are not discovered, but they still can be used to derive relations between classes.

A crucial difference between the Linked Data environment and the traditional

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

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

ontology matching scenario, which focuses on matching two ontologies, is the possibility of using individuals and concepts defined in other repositories and links between them as background knowledge. In our approach we exploit two types of background knowledge:

- Schema-level evidence from third-party repositories.
- Data-level evidence from third-party repositories.

In the following subsections 7.6.1 and 7.6.2 we will describe these types of evidence and briefly outline how they are used to produce schema-level mappings.

7.6.1 Schema-level evidence

In the example shown in Fig. 7.6 the problem with finding overlapping subsets of DBLP and DBPedia is caused by insufficiently detailed classification of individuals provided by the repositories' ontologies. In this situation additional schema-level information has to be introduced from external sources.

Individuals in DBPedia are connected by *rdf:type* links to classes defined in the YAGO repository. The YAGO ontology is based on Wikipedia categories and provides a more detailed hierarchy of classes than the DBPedia ontology. Our algorithm uses this external ontology to identify the subset of DBPedia which overlaps with the DBLP repository. The procedure involves the following steps:

1. Construct clusters of identical individuals from DBPedia and DBLP using existing *owl:sameAs* mappings. In this scenario each cluster corresponds to one *owl:sameAs* link and contains two individuals: one from DBLP and one from DBPedia.

2. Connect these clusters to classes in the YAGO and DBLP ontologies respectively. In the latter case only the class *foaf:Person* is involved. For example, the cluster containing the individual *dbpedia:Andrew_Herbert* is connected to several YAGO classes (e.g., *yago:MicrosoftEmployees*, *yago: BritishComputerScientists* and *yago:LivingPeople*) and to *foaf:Person*.

3. Infer mappings between YAGO classes and the *foaf:Person* class used in DBLP using instance-based matching (see section 7.8). A set of overlapping YAGO classes is produced as a result: e.g., mappings between *foaf:Person* and *yago: MicrosoftEmployees* and between *foaf:Person* and *yago: BritishComputerScientists*.

4. Run instance-level coreference resolution for individuals belonging to the mapped classes to discover more coreference resolution links. For example, at this stage we discover the link between the individual *dbpedia:Charles_P._Thacker* belonging to the class *yago:MicrosoftEmployees* and its DBLP counterpart, which did not exist in the original link set.

7.6.2 Data-level evidence

Data-level evidence includes individuals defined in third-party repositories and coreference links to and from them. The scenario shown in Fig. 7.8 illustrates the use of this type of evidence. The LinkedMDB repository[9] contains data about movies structured using a special Movie ontology. Many of its individuals are also mentioned in DBPedia under different URIs. Some of these coreferent individuals, in particular,

[9]http://data.linkedmdb.org/

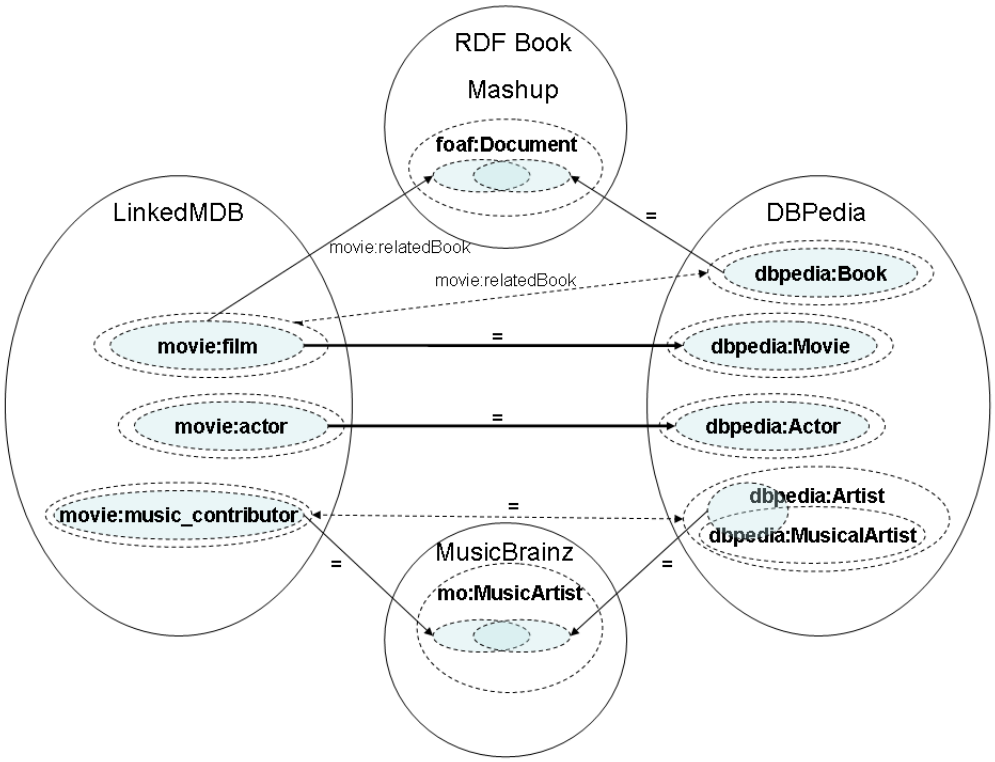

Figure 7.8: LinkedMDB and DBPedia: exploiting instance-level coreference links with third-party datasets. Solid arrows show existing *owl:sameAs* (=) and *movie:relatedBook* links. Dashed arrows connect sets containing potentially omitted links.

those belonging to classes *movie:film* and *movie:actor*, are explicitly linked to their counterparts in DBPedia by automatically produced *owl:sameAs* relations. However, for individuals of some classes, direct links are not available. For instance, there are no direct links between individuals of the class *movie:music_contributor* representing composers, whose music was used in movies, and corresponding DBPedia resources. Then, there are relations of the type *movie:relatedBook* from movies to related books in RDF Book Mashup but not to books mentioned in DBPedia. Partially, such mappings can be obtained by computing a transitive closure for individuals connected by coreference links: obtaining all pairs of individuals which are connected indirectly via a chain of coreference mappings. However, many links are missed in this way because of the omission of an intermediate link in a chain (e.g., 32% of *movie:music_contributor* instances were not connected to corresponding DBPedia instances). Again, such links can be discovered by comparing corresponding subsets of LinkedMDB and DBPedia directly. To discover these subsets our approach computes a transitive closure over existing mappings and combines co-referring individuals into clusters. These clusters are used as evidence for the schema matching procedure to derive schema-level mappings: in our example, we derive the correspondence between *movie:music_contributor* and *dbpedia:Artist* and the *rdfs:range* relation between the property *movie:relatedBook* and the class *dbpedia:Book*. These mappings are used afterwards to perform coreference resolution over related subsets. In our workflow (see Fig. 7.2) the *ontology integration* stage is performed in a "bottom-up" way exploiting both schema-level and data-level evidence while the *knowledge base integration* stage uses them in a "top-down" way. In sections 7.7 and 7.8 we will describe these two main stages of the workflow in more detail.

7.7 Inferring schema mappings: the "bottom-up" stage

The process of inferring schema mappings starts by composing clusters of individuals from different repositories. At this stage pairs of connected individuals belonging to different datasets are retrieved. Then the system forms clusters of coreferent individuals by computing transitive closures over available links.

These clusters represent the basic evidence, which we use to infer schema-level mappings. For each individual in a cluster we extract its class assertions. We consider that a cluster belongs to a certain class if at least one individual from this cluster belongs to a class. At this stage classes which are used in different datasets are always treated as different classes, even if they have the same URI. For instance, in our example, the Movie ontology used in LinkedMDB and the Music ontology used in Musicbrainz both extend the standard FOAF ontology. But we treat the class *foaf:Person* in both these ontologies as two distinct classes: *foaf:Person@Movie* and *foaf:Person@Music*. This is done in order to discover the actual usage pattern for each class, which may implicitly extend its ontological definition.

At the next step we construct mappings between classes. As we said before, instead of equivalence and subsumption the algorithm produces a special type of relation, which we called *#overlapsWith*. Formally this relation is similar to the *umbel:isAligned* property[10] and states that two classes share a subset of their individuals. However, in our case, a quantitative assessment of the relation is necessary to distinguish between strongly correlated classes and merely non-disjoint ones. For instance,

[10] http://www.umbel.org/technical_documentation.html

the concept *dbpedia:Actor* denotes professional actors (both cinema and stage), while the concept *movie:actor* in LinkedMDB refers to any person who played a role in a movie, including participants in documentaries, but excluding stage actors. Because of this, these classes are strongly correlated, but not equivalent. On the other hand, the classes *movie:actor* and *dbpedia:FootballPlayer* only share a few instances (e.g., 'Vinnie Jones") which, however, do not allow these classes to be defined as disjoint.

The *#overlapsWith* relation has a quantitative measure varying between 0 (meaning the same as *owl:disjointWith*) and 1 (meaning that there is a *rdfs:subClassOf* relation in one direction or both). We calculate similarities between classes based on the sets of clusters assigned to them. Two criteria are used to produce the output set of *#overlapsWith* relations between classes:

1. The value of the overlap coefficient compared to a threshold.

$$sim(A,B) = overlap(c(A), c(B)) = \frac{|c(A) \cap c(B)|}{min(c(A), c(B))} \geq t_{overlap},$$

   where c(A) and c(B) are sets of instance clusters assigned to classes A and B respectively. The overlap coefficient was chosen as a similarity metric to reduce the impact of dataset population sizes. If the first dataset is populated to a lesser degree than the second one, then for most classes $|c(A)| << |c(B)|$. In this case relatively small changes in $|c(B)|$ would have a big impact on such distance metrics as Jaccard score or Dice coefficient, while different values of $|c(A)|$ would not change the value significantly.

2. Choosing the "best match" mapping among several options. It is possible that for the same class, A, several relations are produced, which connect it to classes

at different levels of the hierarchy. For instance, we can have both *overlapsWith(A, B)* and *overlapsWith(A, C)*, where $B \sqsubseteq C$: e.g., the class *movie:actor* from LinkedMDB can be matched either to the class *dbpedia:Actor*, or to its superclass *dbpedia:Person*. In our scenario the relation with a more generic class will always override the more specific one: it will mean that individuals of A will have to be compared to individuals of C. Therefore, only one such relation should be chosen, and the original overlap coefficient value cannot be used as a criterion: if $|c(A)| \leq |c(B)|$, then the relation $sim(A, B) \leq sim(A, C)$ always holds. Selecting the relation with the more generic class will mean that possibly more coreference resolution links will be discovered between individuals of A and $C \setminus B$: for example, in our LinkedMDB-DBPedia example, we will find relations between some *movie:actor* instances and DBPedia people who participated in movies without being professional actors (e.g., General Montgomery in the 1943 documentary "Desert Victory"). On the other hand, if the overlap between A and $C \setminus B$ is small and $|C \setminus B|$ is big, then more erroneous mappings can be produced and the damage to results quality due to the loss of precision will be higher than a possible gain from recall increase: e.g., if we align *movie:actor* with *dbpedia:Person*, we will potentially generate many incorrect mappings between namesakes. To make this decision we use the following criterion:

$$\frac{(|A \cap C| - |A \cap B|)/|A \cap C|}{(|C| - |B|)/|C|} \geq \lambda,$$

where λ reflects both the expected ratio of errors for the instance coreference resolution algorithm and relative importance of precision comparing to recall.

If the inequality holds, then *overlapsWith(A, C)* is chosen, otherwise *overlapsWith(A, B)* is preferred.

In our tests we used an additional restriction: pairs of classes (A, B) where either $|A| = 1$, $|B| = 1$ or $|A \cap B| = 1$ were ignored. This was done to filter out weak overlap mappings such as the one between *foaf:Person@DBLP* and *yago: PeopleFromRuralAlberta*, which led to noise at the instance-level matching stage.

The resulting schema-level mappings obtained by the system are reused at the *knowledge base integration* stage of the KnoFuss workflow to produce coreference resolution links between individuals and improve the recall in comparison with existing relations.

7.8 Exploiting inferred schema mappings for coreference resolution: the "top-down" stage

The schema-level mappings obtained at the previous stage are used to identify sets of individuals in different repositories, which are likely to contain equivalent individuals not discovered before. These sets of relevant schema-level mappings are provided as input to the *instance transformation* stage of the KnoFuss tool (Fig. 7.2). As described in section 7.4.2, it uses schema mappings to translate SPARQL queries, which select sets of individuals to be compared, from the vocabulary of one ontology into the terms of another one. It is possible that a class in one ontology is found to be connected to several classes in another ontology (not related via a *rdfs:subClassOf* relation). Such mappings are aggregated into a single *ClassUnion* mapping. For instance, in our DBLP example to select individuals from the DBLP dataset we use

the following query:

SELECT *?uri* **WHERE**

{ *?uri rdf:type foaf:Person*}

To select potentially comparable individuals from the DBPedia repository this query is translated into:

SELECT *?uri* **WHERE**

{ {*?uri rdf:type yago:AmericanComputerScientists*}

UNION { *?uri rdf:type yago:GermanComputerScientists*}

UNION { *?uri rdf:type yago:GoogleEmployees*}

UNION... }

Using these translated queries individuals from both repositories are processed in the same way as if they shared the same schema. The system can employ several basic matching techniques, which can be selected and configured depending on the type of data as described in [87].

To avoid redundancy and potential errors, individuals, which were already connected either directly or indirectly via a third-party dataset, are excluded from analysis. The final set of instance-level mappings produced by the tool can then be added to existing ones.

7.9 Evaluation

For our initial experiments we used three scenarios mentioned before:

1. Finding equivalence links between individuals representing people in DBPedia and DBLP (auxiliary dataset: YAGO, gold standard size 1229).

2. Finding equivalence links between *movie:music_contributor* individuals in LinkedMDB and corresponding individuals in DBPedia (auxiliary dataset: Musicbrainz, gold standard size 942).

3. Finding *movie:relatedBook* links between *movie:film* individuals in LinkedMDB and books mentioned in DBPedia (auxiliary dataset: RDF Book Mashup, gold standard size 419).

Our goal was to check the applicability of our approach in general and, in particular, the possibility to improve coreference resolution recall in comparison with already existing links. Thus, all three scenarios were of relatively small scale so that both precision and recall could be checked manually and the actual coreference resolution was performed using simple label-based similarity (Jaro metric[11]). Test results (precision, recall and F1 measure) are given in the Table 7.5. For each scenario three sets of results are provided:

- Baseline, which involves computing the transitive closure of already existing links published in corresponding repositories.

- Results obtained by KnoFuss when applied to all comparable individuals (i.e., without discarding those already connected via existing links.

- Combined set of existing results and new results obtained by the algorithm.

As was expected, in all cases applying instance-level coreference resolution using automatically produced class-level mappings led to improvement in recall due to the

[11]In the first two scenarios the metric was adapted for personal names, e.g., to match complete name with initials

Table 7.5: Test results for the instance-based schema matching algorithm

Dataset	Test	Precision	Recall	F1
DBPedia vs DBLP	Baseline	0.90	0.14	0.25
	All individuals	0.95	0.88	0.91
	Combined set	0.93	0.89	0.91
LinkedMDB vs DBPedia (music contributors)	Baseline	0.99	0.68	0.81
	All individuals	0.98	0.91	0.94
	Combined set	0.98	0.97	0.98
LinkedMDB vs DBPedia (books)	Baseline	0.97	0.82	0.89
	All individuals	0.98	0.90	0.93
	Combined set	0.96	0.97	0.96

discovery of previously missed mappings and to a better overall performance, as measured by the F1-measure. In all cases, the best performance and F1-measure was achieved by combining newly produced mappings with existing ones. It means that the algorithms, which produced these sets of links, could generate complementary results and no set of links was redundant.

Obviously, the precision of the combined set of links was lower than the precision of the best algorithm in all three tests. In our tests this decrease was relatively small and was compensated by the increase in recall. However, in cases where the same data were already processed by algorithms of higher quality, the situation can be different. It makes the issue of tracing provenance of existing links important, as mentioned in section 7.8.

Considering the schema matching stage we found two factors which were potential causes of errors. The first factor was insufficient evidence. When only a small number of existing coreference links are available as evidence, distinguishing between "weakly overlapped" and "strongly overlapped" classes is problematic. For example, in the DBPedia-DBLP scenario the class *yago: FellowsOfWolfsonCollege,Cambridge*

received a higher overlap score with *foaf: Person@DBLP* than the class *yago: IsraeliComputerScientists*, which in fact was strongly overlapped. This happened because for both of these classes there were only 2 evidence links available and the class *yago: FellowsOfWolfsonCollege,Cambridge* contained fewer instances. At the coreference resolution stage instances of such weakly overlapped classes caused the majority of false positive mappings because of name ambiguity.

The second factor concerned the quality of the ontologies themselves and of the class assertion statements. For instance, in the DBPedia dataset many musicians were not assigned to an appropriate class *dbpedia:MusicalArtist* but instead were assigned to more general classes *dbpedia:Artist* or even *dbpedia:Person*. As a result the "best fit" mappings produced by the algorithm did not correspond to the originally intended meaning of classes, because this originally intended meaning was not followed in the dataset (e.g., based on instance data the class *movie:music_contributor* was mapped to the class *dbpedia:Artist* instead of *dbpedia:MusicalArtist*). More serious issues involved instances being assigned to classes, which were actually disjoint (e.g., the individual *dbpedia:Jesse_Ventura* was classified as both a *dbpedia:Person* and a *dbpedia:TelevisionShow*). While, in our scenarios, spurious schema mappings caused by these errors were filtered out by the threshold, in other cases their impact can be significant. Explicit specification of ontological constraints can help to deal with such situations.

7.10 Conclusion

As pointed out at the beginning of this chapter (section 7.1), we pursued four primary goals when implementing the multi-ontology extension of the KnoFuss architecture:

- To make the use of the KnoFuss architecture in a multi-ontology environment possible.
- To discover specific features of multi-ontology data fusion in comparison with the single-ontology case.
- To test the possible impact caused by inaccuracies introduced at the schema-matching stage.
- To discover possible ways of reducing the impact of ontological heterogeneity.

At a more general level, all these goals were related to research question 6: "What is the impact of ontology schema heterogeneity on the data fusion process?"

In comparison with the single-ontology data fusion scenario adding the ontology heterogeneity challenge results both in a decreased reliability of the methods' outputs and difficulties in the precise estimation of this decrease. For data-level coreference resolution methods we assume that the performance of the method depends on some common features of individuals belonging to a class: this assumption was the basis for the use of *application context* structures in the KnoFuss architecture (section 3.6.3). For ontology matching methods, even knowing the estimated quality of a method (e.g., precision/recall in some test scenarios), it is hard to estimate whether it will hold for a different pair of ontologies. Also, it is hard to measure precisely the impact of a single ontology-level error at the data level. This impact can result in:

- Erroneous widening or narrowing of the applicability range of integration methods (misaligned concepts).

- Providing noisy evidence for data-level methods (misaligned properties and ontological restrictions).

Finally, some ontological mismatches, such as those originating in different modelling styles cannot be resolved fully automatically by currently existing tools and can make data-level methods inapplicable.

We can outline several directions for assisting data fusion in the presence of schema heterogeneity. First, given that the output is likely to be noisy it is necessary to keep track of data integration decisions (such as instance coreference mappings or statements considered incorrect) and their provenance.

Second, considering the limited capabilities of automatic ontology matching methods, the availability of trusted and reusable schema-level background knowledge is important. Such reference knowledge bases are useful when they cover the gaps existing in common ontology matching scenarios. Among others, such reference knowledge may include:

- Specifying rich semantic restrictions existing in a certain domain, e.g., disjointness relations, property cardinality and domain/range constraints. Often such semantic restrictions are omitted by the authors of ontologies either in order to reduce the reasoning or just because they are not needed in the intended ontology use scenario.

- Covering common ontological mismatches which cannot be resolved automatically. For instance, these can include transformation rules between different

time modeling approaches and overlaps between subclasses of the same concept divided according to different criteria (e.g., classifying historical artifacts from China by centuries or by dynastic periods). In this way, a complex modeling style mismatch can be reduced to a terminological one, which can be treated automatically.

Third, sometimes existing automatic matching tools impose overly rigid restrictions on their output because they aim to improve their precision. For instance, some tools (including Lily from our tests) produce only one-to-one equivalence mappings assuming that two different classes in one ontology cannot be considered equivalent to the same class in another ontology. Thus, only the best candidate for equivalence is selected and all others are filtered out. While this is a useful assumption for dealing with terminological mismatches, it may miss important mappings in the presence of conceptualisation and modelling style mismatches.

In order to address this issue and improve the performance of the schema matching stage, we proposed an algorithm which uses instance-based schema matching and exploits third-party data repositories as background knowledge. The tests of the algorithm have shown a substantial improvement in the coreference resolution performance in comparison with the existing mappings. In particular, the recall increased because many previously omitted mappings have been discovered by KnoFuss.

Chapter 8

Contributions and future work

This chapter summarises the contributions of the thesis and outlines important directions for future work. We briefly discuss three topics where this dissertation provides contributions and outline two directions, which we consider the most important for future work: adaptation of the developed system for Web scale semantic data integration, and extension of the library of methods which the architecture can use.

8.1 Summary of the research

As stated in Chapter 1, the general research question we have addressed was: "How can we perform data-level fusion of semantically annotated data coming from different sources?" This question includes the following six more specific research questions:

1. How should algorithms performing fusion subtasks be used in combination to implement the semantic data fusion workflow?
2. How can we support the reusability of fusion algorithms across domains?
3. How can we exploit axioms defined in domain ontologies to improve the performance of fusion algorithms?
4. What kind of uncertainty management framework is suitable for fusion?
5. How can we exploit uncertainty and provenance information to improve the fusion performance?
6. What is the impact of ontology schema heterogeneity on the data fusion process?

These questions were formulated based on the review of related work described in Chapter 2 and concern three main aspects:

- The design of an architecture combining different fusion algorithms.
- The development of methods for handling uncertainty in the fusion process.
- The analysis of the impact on the fusion process caused by schema heterogeneity.

In the following sections we summarise our contributions with respect to these three aspects.

8.2 Contributions of the research

Based on the research questions outlined in section 1.3, three main contributions can be outlined. First, we have developed a data fusion architecture KnoFuss, which allows the combination of different fusion algorithms to support flexible, domain-independent integration of semantic data. Second, we have developed a novel algorithm, which uses formal reasoning about uncertainty to refine fusion results. Third, in order to cover the whole data fusion process in a multi-ontology environment, we extended the KnoFuss architecture enabling it to use results from automatic ontology matching tools.

8.2.1 Contribution 1

Chapter 3 concerns research questions 1-3 and describes the first contribution of this dissertation:

- The KnoFuss architecture adopts the principles of problem-solving methods to combine different algorithms performing various fusion subtasks into a workflow in a flexible way. Appropriate methods are selected, based on their capabilities and the data to be integrated, in order to maximise the quality of the output data.

A major factor, which makes data-level fusion different from schema integration, is the problem of granularity: the need to employ different methods and different configuration parameters not only for different input datasets but for different types of entities within the same dataset. This aspect had not been studied in existing ontology matching systems. For this reason, in the KnoFuss architecture, a novel

approach for organizing the library of methods is used. Method settings are specified depending on the data domain to which a method is applied, thus maximising reusability of methods across domains. At the same time, the need to define optimal parameters for methods for each specific class manually would require significant user effort. To reduce this effort, the class hierarchy defined in the ontology is used to assist in determining optimal settings for each class of entities and reuse the settings for entities of related classes (e.g., those sharing the same superclass). A commonly used approach to determine optimal configuration parameters is machine learning, which normally requires training data to be available. Using a class hierarchy in the system configuration allows machine learning algorithms to reuse and combine training data instances belonging to different classes and to produce optimal decision models. These features ensure a very flexible architecture where new methods can be integrated into the system with minimal effort, and allow the data fusion process to be adjusted for specific use case scenarios.

8.2.2 Contribution 2

Chapters 4 and 5 investigate research questions 4-5. Based on the analysis of the data fusion problem requirements we proposed what we consider the second contribution of this dissertation:

- Our algorithm for processing data interdependencies to improve data fusion quality uses the Dempster-Shafer formalism [106] for uncertainty representation, and valuation networks [108] for belief propagation. Interdependencies between data statements and coreference mappings are localised, and translated into belief propagation networks. The confidence degrees of logical statements are

propagated through the network and updated. The updated confidence degrees are used to refine the resulting knowledge base.

Most existing work in the domain of uncertainty reasoning for the Semantic Web employs either fuzzy or probabilistic approaches. However, for the reasons explained in section 4.2, these approaches are not optimal for the data fusion scenario. Fuzzy reasoning deals with the problem of vagueness rather than the data reliability one. The Bayesian interpretation of probability theory is in principle suitable, but its restriction on having a single uncertainty value does not allow distinctions between weak positive evidence and strong negative evidence to be made, which are relevant for the knowledge fusion scenario. Thus, our answer to question 4 was the choice of the Dempster-Shafer interpretation of uncertainty, which makes it possible to represent ignorance and reason about it.

In order to reason about uncertain datasets, we developed a novel algorithm which combines logical reasoning over ontological axioms with uncertainty reasoning over uncertain data statements to update initial beliefs of statements, select statements most likely to be incorrect, and reinforce others that are likely to be correct. A distinctive feature of our algorithm is its capability to reason about uncertain data statements and uncertain coreference resolution decisions in combination (Chapter 5). This led to an improvement in the overall fusion performance in our experiments (Chapter 6).

8.2.3 Contribution 3

Chapter 7 considers research question 6. In order to perform data fusion in a multi-ontology environment, the results of automatic ontology matching algorithms must

be exploited. Hence, the third contribution of the thesis is the following:

- The analysis of possible ontology mismatches and their impact for data-level fusion allowed us to discover several important factors (see below), which limit the possibilities to reuse existing ontology alignment systems in combination with data-level techniques. Additionally, we developed an extension of the KnoFuss architecture for the multi-ontology fusion scenario and proposed a novel schema-matching algorithm aimed at assisting the data fusion process in the Linked Data environment.

The analysis of ontology mismatches and the tests we performed provide us with several insights into the problem of overcoming semantic heterogeneity when performing data fusion. First, the correspondence patterns based on description logic relations (equivalence and subsumption), which are produced by most existing ontology alignment tools, were found to be insufficient for the data fusion scenario, where it is important to know the degree of overlap between classes in the ontology. Also, when we applied automatically obtained schema mappings for data fusion, the recall performance of the schema matching stage was found to be more important than precision. To decrease the negative impact of incorrect schema mappings, ontological restrictions, such as class disjointness, were found to be useful.

To address these issues we focussed on instance-based schema-matching techniques and developed a novel schema-matching algorithm, which exploits pre-existing instance coreference mappings published in third-party repositories as background knowledge for schema matching. In contrast with existing schema matching tools, which produce crisp schema mappings in the form of description logic axioms, our algorithm produces fuzzy mappings between classes, which are more suitable for the

data-level fusion task.

8.3 Limitations and future work

In this section we will discuss the issues which we consider the main limitations of our work, and the future work needed to overcome them.

8.3.1 Web-scale data integration

KnoFuss was initially developed for the enterprise knowledge management scenario where semantic data are created by automatically annotating documents produced inside a company. This scenario motivated several development assumptions such as the availability of a single central repository, the possibility of measuring data quality based on the reliability of corresponding information extraction algorithms, etc. However, if we consider the scenario of fusing Semantic Web data (e.g., published according to the Linked Data standard), these assumptions do not hold. Further work is needed to make KnoFuss operable on a Web scale.

The first of such problems concerns schema heterogeneity. As discussed in Chapter 7, different types of schema mismatches can have a negative impact at the data fusion stage and are hard to overcome by automatic schema matching tools. The straightforward approach, where schema matching and data matching are performed sequentially, can lead to significant decrease in data fusion performance, primarily because of omissions of valuable schema-level mappings. Although the schema-matching algorithm we proposed in Chapter 7 addresses this problem, its main limitation is its dependency on pre-existing data-level mappings. A possible way to overcome this

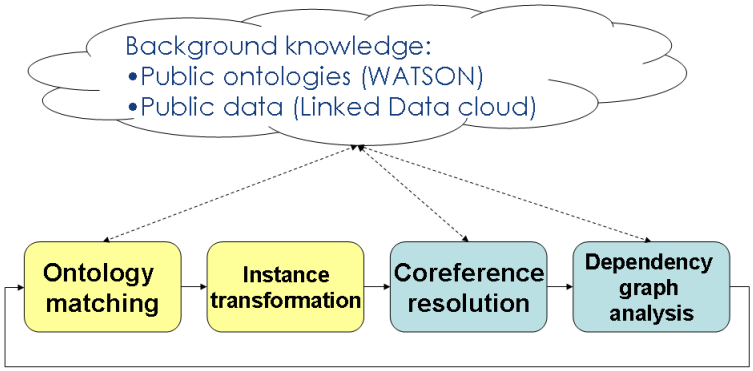

Figure 8.1: Iterative fusion workflow.

issue is to switch from a sequential to an iterative workflow, where schema- and data-level integration steps are performed in a loop (Fig. 8.1), as in the approach proposed in [119]. Implementing such a workflow will require redesigning and extending the KnoFuss architecture in several directions, including:

- Incremental processing of data: at each iteration after each fusion subtask the architecture has to consider what additional information has been produced since the last iteration, what implications it has for generating new coreference links, what methods will be useful at the next stage and whether the system should stop.

- Advanced management of schema mappings: knowledge about mappings established or rejected at one stage has to be reused at the next stage to refine results of instance-level methods.

There are also infrastructural issues, which have to be taken into account in order to make our system usable in a Web-scale data integration context. In particular, these concern storing, publishing and maintaining both schema-level and data-level links. Hence, there are several interesting directions that can followed, such as applying the

coreference bundles approach [48] instead of maintaining sets of pairwise *owl:sameAs* links, and integrating KnoFuss with the *idMesh* approach [22], which reasons about sets of coreference links and their reliability.

Another consequence of dealing with data fusion on a Web scale is related to the physical properties of published semantic data. Public Linked Data repositories are primarily accessible as SPARQL endpoints, not as RDF dumps. While the architecture itself uses SPARQL queries for data retrieval, some methods may need to view the knowledge base as a whole to make decisions (e.g., schema matching algorithms employing structure-level techniques). Additional pre-processing techniques would be needed to collect from a repository all information needed by such methods (e.g., complete schema ontologies which describe data structure). Moreover, given the time and traffic cost of posting a query to a remote repository, the fusion process would have to be optimised to minimise the number of necessary queries. Such optimisation could include local caching of query results or even partial downloading of the repository in advance. Arguably, the most important part of the system in need of optimization is *blocking* for coreference resolution - i.e., selecting pairs of individuals as candidates for matching. Given that datasets may contain thousands of individuals, it is very time consuming to consider all possible pairs. In the original version of KnoFuss blocking was performed using the Lucene index of the target knowledge base: the enterprise knowledge management scenario assumed that fusion will concern adding new data into the main repository, hence, creating and maintaining an index of this repository was justified. This assumption does not hold for a Linked Data fusion scenario, so the indexing approach may not be an optimal one: constructing an index requires downloading of all data from one of the datasets in advance.

Alternative blocking techniques must therefore be investigated.

8.3.2 Populating the library of methods

When designing the architecture we did not focus on implementing complex methods to handle all fusion subtasks. For example, for the coreference resolution stage we primarily used popular string similarity matching techniques. This approach was useful for experiments, as we were interested in the impact of specific factors on the fusion process rather than in finding the best possible algorithm: for instance, when testing the belief propagation algorithm for inconsistency resolution, coreference errors were important as they provided illustrative examples. If basic coreference resolution methods had better performance, it would require a much bigger test dataset to see the impact of belief propagation. For wide practical usage, of course, maximizing the quality of fusion output is essential and including the best available methods is necessary, as well as having a wide range of applicable methods.

It will be interesting to investigate the role of machine learning algorithms for fusion subtasks in the Linked Data environment. The major factor limiting their use is the lack of training data. The KnoFuss architecture tries to minimise the amount needed using the concept hierarchy (Chapter 3) and there are other techniques such as active learning [98] which address the same goal, although they cannot solve the problem completely. However, in the Linked Data scenario, coreference relations between repositories are publicly available and can be used directly as training examples. There are several potentially limiting factors, such as the uncertain quality of these links for different pairs of datasets and omitted coreference mappings, which can be interpreted as negative examples. Hence, further work is necessary to study the implications of training machine learning algorithms on Linked Data repositories.

Another important issue, which was considered external to the work described in

this thesis, concerns the user involvement in the fusion process. The original architecture was designed as an intermediate module of the X-Media kernel system, which had to process documents, extract semantic annotations and populate the corporate knowledge base in an automated way. Thus, in the KnoFuss architecture, only limited user involvement is allowed for defining the method library and configuration parameters. For a wider range of tasks, such as finding coreference links for a newly published dataset, support for an interactive fusion process must be included. The user must be able to check the results of the fusion process, correct them, select and manipulate subsets of mappings, etc. These user's activities can then in turn be logged and used to update the decision models of machine learning methods.

8.4 Outlook

The research described in this thesis considers the problem of data fusion in the specific context of semantic data structured using ontologies. While the research on data integration has a long history, the features of semantic data, such as its distributed and uncertain nature, expressiveness of ontological languages, and varying reliability of applicable algorithms, generate their own challenges, which require special consideration. In our research we proposed several ways to address these challenges.

The design of the KnoFuss architecture takes into account the ontological structure of the data to select and configure fusion algorithms. At the same time the architecture does not make any assumptions about a particular domain or a specific ontology. Uncertainty degrees of data statements and fusion methods' results are formally represented at the architecture level and serve as evidence for taking decisions

about fusion results. Uncertainty reasoning is combined with ontological inferencing to process complex cases of interdependencies between data statements, coreference mappings, and ontological restrictions to maximise the quality of the architecture's output.

This research work started by assuming a specific and restricted scenario of populating a corporate knowledge base. Since then we have witnessed a rapid growth of semantic data published on the Web, which makes the data fusion issue even more important. While the data integration in the Linked Data scenario presents new problems, the basic assumptions we incorporated in the design of KnoFuss still hold and some of them are even emphasised: the range of domains is wider, information is physically distributed, and the problem of uncertainty becomes more acute because of the varying quality and origin of datasets.

The growing Linked Data environment provides perfect opportunities for applying and extending the work on data fusion. At the moment, the integration of newly published datasets is usually performed on an ad-hoc basis: specific techniques are used to find coreference relations with individuals from other data sources describing overlapping topics. With the growing number of published data sources, the automation and the generalization of this procedure will be needed. The capabilities of the KnoFuss architecture (in particular, its extendability) make it a suitable starting point for an automatic Linked Data integration service.

Bibliography

[1] Sibel Adali and Ross Emery. A uniform framework for integrating knowledge in heterogeneous knowledge systems. In *ICDE*, pages 513–520, 1995.

[2] Zharko Aleksovski, Michel C. A. Klein, Warner ten Kate, and Frank van Harmelen. Matching unstructured vocabularies using a background ontology. In *15th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2006)*, pages 182–197, 2006.

[3] Boanerges Aleman-Meza, Chris Halaschek, Amit Sheth, I. Budak Arpinar, and Gowtham Sannapareddy. SWETO: Large-scale Semantic Web test-bed. In *Workshop on Ontology in Action, 16th International Conference on Software Engineering and Knowledge Engineering (SEKE2004)*, pages 21–24, 2004.

[4] Daniel Barbará, Hector Garcia-Molina, and Daryl Porter. The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 4(5):487–502, 1992.

[5] Nuel J. Belnap. A useful four-valued logic. In J.M. Dunn and G. Epstain, editors, *Modern uses of multiple-valued logic*, pages 8–37. D. Reidel Publishing Co., Dordrecht, Netherlands, 1977.

[6] Richard Benjamins and Christine Pierret-Golbreich. Assumptions of problem-solving methods. In *9th European Knowledge Acquisition Workshop (EKAW-96)*, volume 1076 of *Lecture Notes in Artificial Intelligence*, pages 1–16. Springer-Verlag, 1996.

[7] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American Magazine*, 284(5):34–43, 2001.

[8] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 39–48, Washington DC, 2003.

[9] Mikhail Bilenko, Raymond J. Mooney, William W. Cohen, Pradeep Ravikumar, and Stephen Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

[10] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS) (to appear).*

[11] Jens Bleiholder and Felix Naumann. Conflict handling strategies in an integrated information system. In *Workshop on Information Integration on the Web (IIWeb)*, 2006.

[12] Isabelle Bloch, Anthony Hunter, Alain Appriou, Andr Ayoun, Salem Benferhat, Philippe Besnard, Laurence Cholvy, Roger Cooke, Frdric Cuppens, Didier Dubois, Hlne Fargier, Michel Grabisch, Rudolf Kruse, Jrme Lang, Serafn Moral,

Henri Prade, Alessandro Saffiotti, Philippe Smets, and Claudio Sossai. Fusion: General concepts and characteristics. *International Journal of Intelligent Systems*, 16(2):1107–1134, 2001.

[13] Paolo Bouquet, Luciano Serafini, Stefano Zanobini, and Simone Sceffer. Bootstrapping semantics on the web: Meaning elicitation from schemas. In *15th International World Wide Web Conference (WWW 2006)*, pages 505–512, Edinburgh, UK, 2006.

[14] Paolo Bouquet, Heiko Stoermer, and Barbara Bazzanella. An Entity Name System (ENS) for the Semantic Web. In *5th Annual European Semantic Web Conference (ESWC 2008)*, pages 258–272, 2008.

[15] Silvana Castano, Alfio Ferrara, Davide Lorusso, Tobias Henrik Näth, and Ralf Möller. Mapping validation by probabilistic reasoning. In *5th Annual European Semantic Web Conference (ESWC 2008)*, pages 170–184, Tenerife, Spain, 2008.

[16] Balakrishnan Chandrasekaran, Todd R. Johnson, and Jack W. Smith. Task-structure analysis for knowledge modeling. *Communications of the ACM*, 35(9):124–137, 1992.

[17] Zhaoqi Chen, Dmitri V. Kalashnikov, and Sharad Mehrotra. Adaptive graphical approach to entity resolution. In *ACM IEEE Joint Conference on Digital Libraries 2007 (ACM IEEE JCDL 2007)*, pages 204–213, Vancouver, British Columbia, Canada, 2007.

[18] Laurence Cholvy and Anthony Hunter. Information fusion in logic: A brief

overview. In *1st International Joint Conference on Qualitative and Quantitative Practical Reasoning*, pages 86–95, 1997.

[19] Fabio Ciravegna and Yorick Wilks. *Annotation for the Semantic Web*, chapter Designing Adaptive Information Extraction for the Semantic Web in Amilcare. 2003.

[20] Munir Cochinwala, Verghese Kurien, Gail Lalk, and Dennis Shasha. Efficient data reconciliation. *Information Sciences*, 137(1–4):1–15, 2001.

[21] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*, 2003.

[22] Philippe Cudré-Mauroux, Parisa Haghani, Michael Jost, Karl Aberer, and Hermann de Meer. idMesh: Graph-based disambiguation of linked data. In *18th International World Wide Web Conference (WWW 2009)*, pages 591–600, Madrid, Spain, 2009. ACM.

[23] Newton C. A. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15(4):497–510, 1974.

[24] Paulo Cesar G. da Costa, Claudia d'Amato, Nicola Fanizzi, Kathryn B. Laskey, Kenneth J. Laskey, Thomas Lukasiewicz, Matthias Nickles, and Michael Pool, editors. *Uncertainty Reasoning for the Semantic Web I, ISWC International Workshops, URSW 2005-2007, Revised Selected and Invited Papers*, volume 5327. Springer, 2008.

[25] Paulo Cesar G. da Costa, Kathryn B. Laskey, and Kenneth J. Laskey. PR-OWL: A Bayesian ontology language for the semantic web. In *Workshop on Uncertainty Reasoning for the Semantic Web, ISWC 2005*, 2005.

[26] Luis M. de Campos, Juan F. Huete, and Serafn Moral. Uncertainty management using probability intervals. In *Advances in Intelligent Computing IPMU '94*, pages 190–199, 1994.

[27] Zhongli Ding and Yun Peng. A probabilistic extension to ontology language OWL. In *37th Hawaii International Conference On System Sciences (HICSS-37)*, 2004.

[28] Hong-Hai Do and Erhard Rahm. COMA: A system for flexible combination of schema matching approaches. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 610–621. VLDB Endowment, 2002.

[29] AnHai Doan, Ying Lu, Yoonkyong Lee, and Jiawei Han. Object matching for information integration: A profiler-based approach. In Subbarao Kambhampati and Craig A. Knoblock, editors, *IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, pages 53–58, Acapulco, Mexico, 2003.

[30] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the semantic web. *VLDB Journal*, 12(4):303–319, 2003.

[31] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Ontology matching: A machine learning approach. In *Handbook on Ontologies in Information Systems*, pages 397–416. Springer, 2004.

[32] Xin Dong, Alon Halevy, and Jayant Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 85–96, New York, NY, USA, 2005. ACM.

[33] Marc Ehrig. *Ontology Alignment: Bridging the Semantic Gap*. Springer, New York, NY US, 2007.

[34] Marc Ehrig and Steffen Staab. QOM - Quick Ontology Mapping. In *3rd International Semantic Web Conference*, pages 683–697, 2004.

[35] Marc Ehrig, Steffen Staab, and York Sure. Bootstrapping ontology alignment methods with APFEL. In *4th International Semantic Web Conference (ISWC-2005)*, pages 186–200, Galway, Ireland, 2005. Springer.

[36] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[37] Jérôme Euzenat. An API for ontology alignment. In *3rd International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 698–712, Hiroshima, Japan, 2004. Springer.

[38] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg, 2007.

[39] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in OWL-Lite. In *15th European Conference on Artificial Intelligence (ECAI)*, pages 333–337, Valencia, Spain, 2004.

[40] Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of American Statistical Association*, 64(328):1183–1210, 1969.

[41] Dieter Fensel and Enrico Motta. Structured development of problem-solving methods. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):913–932, 2001.

[42] Dieter Fensel, Enrico Motta, Frank van Harmelen, V. Richard Benjamins, Monica Crubézy, Stefan Decker, Mauro Gaspari, Rix Groenbroom, William Grosso, Mark Musen, Enric Plaza, Guus Schreiber, Rudi Studer, and Bob Wielinga. The unified problem-solving method development language UPML. *Knowledge and Information Systems*, 5:83–131, 2003.

[43] Alfio Ferrara, Davide Lorusso, and Stefano Montanelli. Automatic identity recognition in the Semantic Web. In *Workshop on Identity and Reference on the Semantic Web, ESWC 2008*, Tenerife, Spain, 2008.

[44] Eugene Fink. Automatic evaluation and selection of problem-solving methods: Theory and experiments. *Journal of Experimental and Theoretical Artificial Intelligence*, 16(2):73–105, 2004.

[45] Haim Gaifman. A theory of higher order probabilities. In *TARK '86: Proceedings of the 1986 conference on Theoretical aspects of reasoning about knowledge*,

pages 275–292, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

[46] John F. Gantz, Christopher Chute, Alex Manfrediz, Stephen Minton, David Reinsel, Wolfgang Schlichting, and Anna Toncheva. The diverse and exploding digital universe: An updated forecast of worldwide information growth through 2011. White paper, IDC, 2008.

[47] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *1st European Semantic Web Symposium (ESWS 2004)*, pages 61–75, Hersounisous, Greece, 2004.

[48] Hugh Glaser, Afraz Jaffri, and Ian Millard. Managing co-reference on the semantic web. In *Workshop on Linked Data on the Web (LDOW 2009), 18th International World Wide Web Conference (WWW2009)*, Madrid, Spain, 2009.

[49] Hugh Glaser, Ian Millard, Afraz Jaffri, Timothy Lewy, and Ben Dowling. On coreference and the Semantic Web. In *7th International Semantic Web Conference (ISWC 2008) (submitted)*, Karlsruhe, Germany, 2008.

[50] Jorge Gracia and Eduardo Mena. Matching with CIDER: Evaluation report for the OAEI 2008. In *3rd Ontology Matching Workshop (OM'08) at the 7th International Semantic Web Conference (ISWC'08)*, Karlsruhe, Germany, 2008.

[51] Eric Grégoire and Sébastien Konieczny. Logic-based approaches to information fusion. *Information Fusion*, 7(1):4–18, 2006.

[52] Ramanathan V. Guha and Rob McCool. TAP: a Semantic Web platform. *Computer Networks*, 42(5):557–577, 2003.

[53] Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure. A framework for handling inconsistency in changing ontologies. In *Proceedings of the International Semantic Web Conference (ISWC2005)*, pages 353–367, 2005.

[54] Peter Haase and Johanna Volker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In *International Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, pages 45–55, 2005.

[55] Mark A. Hall and Geoffrey Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1437–1447, 2003.

[56] Eric J. Horvitz, David E. Heckerman, and Curtis P. Langlotz. A framework for comparing alternative formalisms for plausible reasoning. In *AAAI-86*, pages 210–214, 1986.

[57] Anthony Hunter. How to act on inconsistent news: Ignore, resolve, or reject. *Data & Knowledge Engineering*, 57(3):221–239, 2006.

[58] Anthony Hunter and Weiru Liu. Fusion rules for merging uncertain information. *Information Fusion*, 7(1):97–134, 3 2006.

[59] Jose Iria. Relation extraction for mining the Semantic Web. In *Dagstuhl Seminar on Machine Learning for the Semantic Web*, Dagstuhl, Germany, 2005.

[60] Afraz Jaffri, Hugh Glaser, and Ian Millard. Managing URI synonymity to enable consistent reference on the Semantic Web. In *Workshop on Identity and Reference on the Semantic Web (IRSW2008)*, Tenerife, Spain, 2008.

[61] Matthew A. Jaro. UNIMATCH: A record linkage system, user's manual. Technical report, US Bureau of the Census, 1978.

[62] Ningsheng Jian, Wei Hu, Gong Cheng, and Yuzhong Qu. Falcon-AO: Aligning ontologies with Falcon. In *K-CAP Workshop on Integrating Ontologies*, pages 87–93, Banff (CA), 2005.

[63] Dmitri V. Kalashnikov and Sharad Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Transactions on Database Systems*, 31(2):716–767, 2006.

[64] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau. Repairing unsatisfiable concepts in OWL ontologies. In *ESWC2006*, pages 170–184, 2006.

[65] Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294, 1991.

[66] Wong Kim and Jungyun Seo. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer*, 24(12):12–18, 1991.

[67] Michel Klein. Combining and relating ontologies: an analysis of problems and solutions. In *Workshop on Ontologies and Information Sharing*, 2001.

[68] Sébastien Konieczny and Éric Grégoire. Logic-based approaches to information fusion. *Information Fusion*, 7(1):2–3, 2006.

[69] Sébastien Konieczny and Ramon Pino Pérez. Merging with integrity constraints. In *5th European Conference on Symbolic and Quantitative Approaches*

to Reasoning with Uncertainty (ECS-QUARU'99), volume 1638 of *Lecture Notes in Artificial Intelligence*, pages 233–244, 1999.

[70] Yooonkyong Lee, Mayssam Sayyadian, AnHai Doan, and Arnon S. Rosenthal. eTuner: Tuning schema matching software using synthetic scenarios. *VLDB Journal*, 16:97–122, 2007.

[71] Maurizio Lenzerini. Data integration: A theoretical perspective. In *21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, Madison, Wisconsin, 2002.

[72] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.

[73] Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. *Deterministic and Statistical Methods in Machine Learning*, chapter SVM based learning system for information extraction, pages 319–339. 2005.

[74] Ee-Peng Lim, Jaideep Srivastava, and Shashi Shekhar. Resolving attribute incompatibility in database integration: An evidential reasoning approach. In *10th International Conference on Data Engineering*, pages 154–163, 1994.

[75] Yanbin Liu, François Scharffe, and Chunguang Zhou. Towards practical rdf datasets fusion. In *Workshop on Data Integration through Semantic Technology (DIST2008), ASWC 2008*, Bangkok, Thailand, 2008.

[76] Sandra Marcus, editor. *Automatic Knowledge Acquisition for Expert Systems*. Kluwer Academic, Boston, MA, 1988.

[77] Andrew McCallum and Ben Wellner. Conditional models of identity uncertainty with application to noun coreference. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 905–912. MIT Press, Cambridge, MA, 2004.

[78] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm. In *18th International Conference on Data Engineering (ICDE)*, pages 117–128, San Jose (CA US), 2002.

[79] J. J. Ch. Meyer and W. van der Hoek. Modal logics for representing incoherent knowledge. In *In Handbook of Defeasible Reasoning and Uncertainty Management*, pages 37–75. Kluwer, 1997.

[80] Alvaro E. Monge and Charles P. Elkan. The field matching problem: Algorithms and applications. In *2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 267–270. AAAI Press, 1996.

[81] Amihai Motro. Multiplex: a formal model for multidatabases and its implementation. In *4th International Workshop on Next Generation Information Technologies and Systems (NGITS 99)*, pages 138–158, Zichron Yaacov, Israel, 1999. Springer-Verlag.

[82] Amihai Motro and Philipp Anokhin. Fusionplex: resolution of data inconsistencies in the integration of heterogeneous information sources. *Information Fusion*, 7(2):176–196, 2006.

[83] Enrico Motta. *Reusable Components for Knowledge Modelling*, volume 53 of

Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, 1999.

[84] Howard B. Newcombe, James M. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[85] Mathias Nickles. Social acquisition of ontologies from communication processes. *Journal of Applied Ontology*, 2(3-4):373–397, 2007.

[86] Andriy Nikolov, Victoria Uren, Enrico Motta, and Anne de Roeck. Using the Dempster-Shafer theory of evidence to resolve ABox inconsistencies. In *Workshop on Uncertainty Reasoning for the Semantic Web, ISWC 2007*, Busan, Korea, 2007.

[87] Andriy Nikolov, Victoria Uren, Enrico Motta, and Anne de Roeck. Integration of semantically annotated data by the KnoFuss architecture. In *16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008)*, Acitrezza, Italy, 2008.

[88] Klas Orsvärn. Some principles for libraries of task decomposition methods. *Internation Journal of Human-Computer Studies*, 49:417–435, 1998.

[89] Yannis Papakonstantinou, Serge Abiteboul, and Hector Garcia-Molina. Object fusion in mediator systems. In *Proceedings of the Twenty-second International Conference on Very Large Databases*, pages 413–424, 1996.

[90] Parag and Pedro Domingos. Multi-relational record linkage. In *KDD Workshop on Multi-Relational Data Mining*, pages 31–48, Seattle, CA, USA, 2004. ACM Press.

[91] Simon Parsons and Anthony Hunter. A review of uncertainty handling formalisms. In *Applications of Uncertainty Formalisms*, volume 1455 of *Lecture Notes in Computer Science*, pages 8–37.

[92] Alun Preece, Kit Hui, Alex Gray, Philippe Marti, Trevor Bench-Capon, Zhan Cui, and Dean Jones. KRAFT: An agent architecture for knowledge fusion. *International Journal for Cooperative Information Science*, 10(1–2):171–195, 2001.

[93] Guilin Qi, Weiru Liu, and David Bell. A revision-based approach to handling inconsistency in description logic. *Artificial Intelligence Review*, 26(1–2):116–128, 2006.

[94] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*, 23(4), 2000.

[95] Dnyanesh Rajpatak, Enrico Motta, Zdenek Zdrahal, and Rajkumar Roy. A generic library of problem solving methods for scheduling applications. In *2nd International Conference on Knowledge Capture (KCAP'2003)*, pages 113–120, Sanibel Island, Florida, USA, 2003. ACM.

[96] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[97] Steffen Rendle and Lars Schmidt-Thieme. Object identification with constraints. In *6th IEEE International Conference on Data Mining (ICDM '06)*, 2006.

[98] Steffen Rendle and Lars Schmidt-Thieme. Active learning of equivalence relations by minimizing the expected loss using constraint inference. In *8th IEEE International Conference on Data Mining (ICDM '08)*, pages 1001–1006, 2008.

[99] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the Semantic Web. In *2nd International Semantic Web Conference, ISWC 2003*, pages 351–368, Sanibel Island, Florida, 2003.

[100] Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Exploring the Semantic Web as background knowledge for ontology matching. *Journal of Data Semantics*, XI:156–190, 2008.

[101] Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. L2R: a logical method for reference reconciliation. In *22nd AAAI Conference on Artificial Intelligence (AAAI-07)*, pages 329–334, Vancouver, BC, Canada, 2007. AAAI Press.

[102] Francois Scharffe and Dieter Fensel. Correspondence patterns for ontology alignment. In *16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008)*, pages 83–92, Acitrezza, Italy, 2008.

[103] Stefan Schlobach. Diagnosing terminologies. In *20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference*, pages 670–675, Pittsburgh, PA, USA, 2005.

[104] Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, Nigel Schadbolt, Walter van de Velde, and Bob Wielinga. *Knowledge Engineering*

and Management: The CommonKADS Methodology. MIT Press, Cambridge, Massachusetts, USA, 2000.

[105] Nigel Shadbolt. Garlik: Semantic technology for the consumer. In *5th Annual European Semantic Web Conference (ESWC 2008)*, page 4, Tenerife, Spain, 2008.

[106] Glenn Shafer. *A mathematical theory of evidence*. Princeton University Press, 1976.

[107] Warren Shen, Pedro DeRose, Long Vu, AnHai Doan, and Raghu Ramakrishnan. Source-aware entity matching: A compositional approach. In *International Conference on Data Engineering (ICDE 2007)*, Istanbul, Turkey, 2007.

[108] Prakash P. Shenoy. *Fuzzy logic for the management of uncertainty*, chapter Valuation-based systems: a framework for managing uncertainty in expert systems, pages 83–104. John Wiley & Sons, Inc., New York, NY, USA, 1992.

[109] Prakash P. Shenoy and Glenn Shafer. *Readings in uncertain reasoning*, chapter Axioms for probability and belief-function propagation, pages 575–610. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

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

[111] Parag Singla and Pedro Domingos. Object identification with attribute-mediated dependences. In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PAKDD-2005)*, pages 297–308, Porto, Portugal, 2005.

[112] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[113] Giorgos Stoilos, Giorgos Stamou, Vassilis Tzouvaras, Jeff Z. Pan, and Ian Horrocks. Fuzzy OWL: Uncertainty and the semantic web. In *International Workshop of OWL: Experiences and Directions, ISWC 2005*, 2005.

[114] Umberto Straccia. Towards a fuzzy description logic for the Semantic Web (preliminary report). In *2nd European Semantic Web Conference (ESWC-05)*, number 3532 in Lecture Notes in Computer Science, pages 167–181, Crete, 2005. Springer Verlag.

[115] Umberto Straccia and Raphaël Troncy. oMAP: Combining classifiers for aligning automatically OWL ontologies. In *6th International Conference on Web Information Systems Engineering (WISE)*, 2005.

[116] Bill Swartout, Yolanda Gil, and Andre Valente. Representing capabilities of problem-solving methods. In *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*, Stockholm, Sweden, 1999.

[117] Andreas Thor and Erhard Rahm. MOMA - a mapping-based object matching system. In *3rd Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, USA, 2007.

[118] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: Weaving the open linked data. In *6th International Semantic Web Conference (ISWC/ASWC 2007)*, pages 552–565, Busan, Korea, 2007.

[119] Octavian Udrea, Lise Getoor, and Renée J. Miller. Leveraging data and structure in ontology integration. In *SIGMOD'07*, pages 449–460, Beijing, China, 2007.

[120] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk - a link discovery framework for the web of data. In *Workshop on Linked Data on the Web (LDOW 2009), 18th International World Wide Web Conference (WWW2009)*, Madrid, Spain, 2009.

[121] Peng Wang and Baowen Xu. Lily: Ontology alignment results for OAEI 2008. In *3rd Ontology Matching Workshop (OM'08) at the 7th International Semantic Web Conference (ISWC'08)*, Karlsruhe, Germany, 2008.

[122] M.S. Waterman, T.F. Smith, and W.A. Beyer. Some biological sequence metrics. *Advances in Mathematics*, 20(3):367–387, 1976.

[123] Melanie Weis and Ioana Manolescu. Declarative XML data cleaning with XClean. In *The 19th International Conference on Advanced Information Systems Engineering (CAiSE'07)*, volume 4495 of *Lecture Notes in Computer Science*, pages 96–110, 2007.

[124] William E. Winkler. Improved decision rules in the Fellegi-Sunter model of record linkage. Technical Report RR93/12, US Bureau of the Census, Washington, DC 20233, 1993.

[125] William E. Winkler. The state of record linkage and current research problems. Technical Report RR99/04, US Bureau of the Census, Washington, DC 20233, 1999.

[126] William E. Winkler. Overview of record linkage and current research directions. Technical Report RRS2006/02, US Bureau of the Census, Washington, DC 20233, 2006.

[127] Mikalai Yatskevich and Fausto Giunchiglia. Element level semantic matching using WordNet. In *Meaning Coordination and Negotiation Workshop, ISWC 2004*, Hiroshima, Japan, 2004.

[128] Xiao Zhang, Qian Zhong, Juanzi Li, Jie Tang, Guotong Xie, and Hanyu Li. RiMOM results for OAEI 2008. In *3rd Ontology Matching Workshop (OM'08) at the 7th International Semantic Web Conference (ISWC'08)*, Karlsruhe, Germany, 2008.

[129] Jianhan Zhu, Victoria Uren, and Enrico Motta. ESpotter: Adaptive named entity recognition for web browsing. In *Professional Knowledge Management Conference (WM2005)*, 2005.