ABSTRACT

Title of Document: ON THE FOUNDATIONS OF DATA INTEROPERABILITY AND SEMANTIC SEARCH ON THE WEB

Hamid Haidarian Shahri, Doctor of Philosophy, 2011

Directed By: Professor Donald Perlis
Department of Computer Science

This dissertation studies the problem of facilitating semantic search across disparate ontologies that are developed by different organizations. There is tremendous potential in enabling users to search independent ontologies and discover knowledge in a serendipitous fashion, i.e., often completely unintended by the developers of the ontologies. The main difficulty with such search is that users generally do not have any control over the naming conventions and content of the ontologies. Thus terms must be appropriately mapped across ontologies based on their meaning. The meaning-based search of data is referred to as semantic search, and its facilitation (aka semantic interoperability) then requires mapping between ontologies.

In relational databases, searching across organizational boundaries currently involves the difficult task of setting up a rigid information integration system. Linked Data representations more flexibly tackle the problem of searching across organizational boundaries on the Web. However, there exists no consensus on how ontology mapping should be performed for this scenario, and the problem is open. We lay out the foundations of semantic search on the Web of Data by comparing it to keyword search in the relational model and by providing effective mechanisms to facilitate data interoperability across organizational boundaries.

We identify two sharply distinct goals for ontology mapping based on real-world use cases. These goals are: (i) ontology development, and (ii) facilitating interoperability. We systematically analyze these goals, side-by-side, and contrast them. Our analysis demonstrates the implications of the goals on how to perform ontology mapping and how to represent the mappings.

We rigorously compare facilitating interoperability between ontologies to information integration in databases. Based on the comparison, class matching is emphasized as a critical part of facilitating interoperability. For class matching, various class similarity metrics are formalized and an algorithm that utilizes these metrics is designed. We also experimentally evaluate the effectiveness of the class similarity metrics on real-world ontologies. In order to encode the correspondences between ontologies for interoperability, we develop a novel W3C-compliant representation, named skeleton.

ON THE FOUNDATIONS OF DATA INTEROPERABILITY AND SEMANTIC SEARCH ON THE WEB

By

Hamid Haidarian Shahri

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

To my parents and brother

Acknowledgements

First, I express my deep gratitude to Dr. Don Perlis, my advisor. Don gave me the freedom to pursue my research interests, and provided guidance and encouragement throughout my studies at Maryland. His insight, communication skills, and patience have always amazed me. I have learned many life lessons from him that will help me put things in perspective in the future. Also, I am grateful to the members of my committee Drs. Mark Austin, Amol Deshpande, Tim Finin, Jennifer Golbeck, and Adam Porter for their suggestions and kind support.

I appreciate the suggestions that I received about this work from Jim Hendler, Hugh Glaser, and Philip Bernstein. Many friends have helped me in my graduate studies. I would like to thank Shomir Wilson, Wikum Dinalankara, Greg Sanders, Vladimir Kolovski, Christian Halaschek-Wiener, Taowei Wang, Ron Alford, and others I may have forgotten. I also thank the members of the ALMECOM research group: Michael Anderson, Michael Cox, Scott Fults, Darsana Josyula, Tim Oates, and Matt Schmill. Thanks to my past mentors: Ahmad Abdollahzadeh, Hossein Pedram, Mahmoud Naghibzadeh, Mohsen Kahani, and Mohammad Hossien Yaghmaee.

My family in the US have been very caring and supportive, especially Saied, Mehdi, and Mina. Mehdi is the person that I can always turn to, when I am facing unfamiliar and difficult situations. I am truly indebted to my brother, Saied, for his kindness, willingness to help, and technical expertise.

Finally, I am grateful to my parents for their unconditional love and support, and for all the sacrifices that they have made so that I get the best education. I appreciate all the opportunities they made available to me.

Table of Contents

List of Tables

List of Figures

Chapter 1: INTRODUCTION

This dissertation studies the problem of facilitating semantic search across disparate ontologies, i.e., enabling the search of data across organizational boundaries. Today, users need to search, browse, and discover knowledge stored in different ontologies (knowledge bases), where in general, the ontologies are developed independently by different organizations and maintained autonomously. Hence, users do not have any control over the content of the ontologies, and there are no unifying standards that the ontologies must follow, making such search difficult. In particular, the same term may be used for different entities in different ontologies, and different terms for the same or similar entities. Thus terms must be appropriately mapped or associated across ontologies; any search that ignores meaning will in general perform poorly.

The meaning-based search of data is referred to as semantic search (refer to Section 2.6), and its facilitation (aka semantic interoperability) then requires ontology mapping (refer to Chapter 3). There are then two underlying issues to semantic interoperability: (i) mapping (based on meaning), and then (ii) capturing the results of such mapping in a useful representation.

There is tremendous potential in enabling users to browse and discover knowledge from independent knowledge bases in a serendipitous fashion, i.e., often completely unintended by the developers of the knowledge bases, at the time when the knowledge bases were designed (at design-time). One simple example in the health care domain is as follows: If we are looking at Alzheimer's disease, for drug discovery, there is a large amount of Linked Data which is just coming out, because scientists in that field realize that this is a great way of getting out of their data silos, i.e., scientists had their genomics data in one database in one building, and they had their protein data in another. Now, they are exposing their data in Linked Data format and they can ask: "What proteins are involved in signal transduction and also related to pyramidal neurons?" We can type that question into Google. Of course, there is no one page on the Web which has answered that question, because nobody has asked that question before. From a Google search, we get 223000 hits, but not results that answer the question. We query the Linked Data, which they have now put together, and we get 32 hits, each of which is a protein which has those properties. Now, we have found the proteins that we were looking for. This searching of data across organizational boundaries enables us, as scientists, to answer questions that we were not able to answer before -- questions which actually bridge across different disciplines [Ber09]. In order to perform such a search, however, we need semantic mappings between different data sources.

Applications of semantic mappings: Data can be represented in different ways, for example in relational schemas, ontologies, and XML DTDs. In many applications, there is a need for finding semantic mappings between different representations of data. Finding such semantic mappings is necessary for enabling the manipulation, translation, and querying of data, as explained below. These applications have been studied actively in the database and AI communities.

In databases, one of the earlier applications, which has been studied since the 80's, is in schema integration where a set of schemas are merged into a global schema [Bat86, She90, Par98]. Another application is in data translation between databases, where data from different databases needs to be transformed to conform to a single target schema to allow further analysis. Data translation is one of the critical steps in data warehousing and data mining [Mil00, Rah01].

In recent years, data integration systems which provide a unified query interface to a number of data sources are becoming ubiquitous [Gar97, Ive99, Lam99, Hal05, Hal06]. This unified query interface is achieved by posing the query against a mediated schema that has mappings to the local data sources. There has also been considerable attention on model management, which aims to create tools to easily manipulate models of data, e.g. data representations and ER diagrams [Ber00, Rah01, Ber07]. Matching is a key operation in these data model manipulations.

In the field of AI, building new knowledge bases based on existing ones has been of interest since the 80's. In such knowledge base construction applications it is necessary to find matching entities and relationships [Hef01, Bro01, Ome01, Mae01]. In the last decade, with the start of the Semantic Web efforts, there is a move towards publishing data on the Web (Linked Data) and annotating Web pages [Ber01, Biz09, Mul10, Hea11]. Many of the applications that work with Linked Data and annotated

Web pages need to find correspondences between ontologies in order to facilitate access to independent data repositories.

Challenges: When finding semantic mappings between data representations, each element from one representation needs to be compared to all the elements of the other representation. This exhaustive nature of the search can be a limiting factor in some applications. Often, in the matching process, there is no access to the creators of the representations and the documentation that describes the data representation. Creators may have moved to other organizations and documentations may be old and out of date. So the elements need to be matched based on similarity metrics, but any such metric is unreliable. For example, *area* and *location* can mean the same thing in one scenario, but they can mean different things in another scenario. In other words, there is an inherent uncertainty involved in the matching process. Finally, matching itself can be subjective, i.e. different users may have different opinions about whether two elements correspond or not.

In relational databases, searching across organizational boundaries currently involves the very tedious and difficult task of setting up and maintaining a rigid information integration system, which is confined to some pre-determined and specific domain, for example a flight reservation system such as Expedia. As another example, in a project in the GTE communications company, the goal was to integrate 40 databases that have a total of 27000 attributes in relational tables. The estimated time required to find and document the correspondences was more than 12 person-years [Li00].

Linked Data representations for the Web, like RDF, have been developed and standardized recently, to more flexibly tackle the problem of searching across organizational boundaries – searching the data in an ad hoc and serendipitous fashion [bus07]. However, there exists no consensus on how ontology mapping should be performed for this scenario, and the problem is open [van08, Mil10]. That is, there are no good current methods to solve this problem. In this dissertation when we use the term ontology, we are primarily referring to Linked Data in the form of RDF(S).

Traditional Web search engines, like Google, only perform relevance ranking and largely ignore this Web data. They primarily focus on the shallow Web (Web of documents – unstructured data), which is free-formed text, and not the deep Web (Web of Data – semi structured data), which is data stored in databases or ontologies. Also, traditional relational databases can process expressive queries but do not attempt to work on Web data.

1.1. Contributions

The central focus of this dissertation is to lay out the foundations of semantic search on the Web of Data by comparing it to keyword search in the relational model and by providing effective mechanisms to facilitate data interoperability across organizational boundaries.

I demonstrate the advantages of using semantic search on Linked Data by comparing it to keyword search on the relational data model. Then, some of the crucial research challenges of semantic search on Linked Data are presented. I develop a suitable representation that enables users to discover knowledge from different knowledge bases, more generally and effectively than existing alternatives. I refer to this representation as a skeleton and design the necessary algorithms for creating the skeleton. More specifically, the skeleton provides a uniform representation of mappings across ontologies and is stored independently of those ontologies. A user's search may then be largely guided and managed via the uniform skeleton representation, freeing the user of the burdensome and time-consuming task of mapping during the search.

The key contributions of this dissertation are as follows:

- I provide an overarching definition of ontology mapping that allows the systematic analysis and distinction of different goals of ontology mapping. I clarify the relationship between ontology merging and facilitating interoperability through precise use cases. Then, different implications of the goals are collectively analyzed. These implications serve as a guideline for performing ontology mapping, and they influence the design of tools and algorithms for ontology mapping.
- I rigorously compare facilitating interoperability between ontologies with information integration in databases. Based on this comparison, class matching is emphasized as a critical part of facilitating interoperability, and various class similarity metrics are formalized and evaluated on real-world ontologies.
- I design algorithms for class matching and creating a novel W3C-compliant representation, named skeleton, to encode the correspondences between

ontologies and facilitate interoperability between them. These algorithms are compared to existing approaches.

1.2. Organization

The rest of the dissertation is organized as follows. Chapter 2 describes a detailed map of a system architecture for answering questions using semi-structured data on the Web. We call such a system The Automated Information Assimilator (TAIA). TAIA receives a query from a user. The query is narrowed down to determine what the user is really asking. Relevant data sources are accessed, and the results are packaged and presented back to the user. In Section 2.3, we consider how a user query can be mapped to the specific pieces of information that the user is asking for. In Section 2.4, we describe how the result of a query can be presented to the user in natural language. Section 2.6 demonstrates how the sources of information can be narrowed down, so that the results match the user's intention. We compare semantic search in Linked Data with keyword search in the relational model to show the advantages of semantic search. Then, we outline some of the crucial research challenges of semantic search.

In Chapter 3, the ontology mapping problem is defined, elaborated, and key algorithms are provided. In Section 3.3, we put the ontology mapping problem in context based on its use cases. With the use cases, we distinguish the ontology development goal from facilitating interoperability. Section 3.4 provides the formal definitions for the necessary terminology. In Section 3.5, the ontology mapping problem for interoperability is compared to the information integration problem in databases. We describe the processes involved in information integration and interoperability in databases. Then, we describe how facts are expressed using the RDF and relational models and compare the models. Finally we illustrate that class matching is a critical part of ontology mapping for facilitating interoperability. In Section 3.6-3.8 we formalize four different class similarity metrics and describe the role of the ontology reasoner.

In Chapter 4, we present the algorithms and experiments. Section 4.2 assesses the skeleton for interoperability between ontologies and presents the algorithm for creating the skeleton. Then we discuss the experimental methodology and evaluation.

Chapter 5 describes the related work from different areas, including: dialogue agents; semantic search; Linked Data; ontology integration; schema matching and information integration; and data cleaning in databases. Chapter 6 concludes with a summary of the dissertation and directions for future research.

Parts of the work described in this dissertation have been published in conferences. The work on dialogue agent in Chapter 2 is published in [Hai08a, Hai10b]. The work on semantic search in Chapter 2 is published in [Hai10c]. The work on data interoperability on the Web in Chapter 3 and 4 is published in [Hai08d, Hai10e].

Chapter 2: THE AUTOMATED INFORMATION ASSIMILATOR

2.1. Overview

Humans are inundated with vast amounts of information, today. This information can come from many distributed sources and is far beyond what we can deal with on our own. As a result, there is an increasing demand for (semi)automated systems that sort through and assimilate this "information glut" for us. The high-level goal is to create an assimilator that is a go-between humans and the information. The assimilator would get the queries from a human and then gather information from all relevant sources, by culling through it, as accurately as possible. It pulls together all that bears usefully on what the human wants to know and provides the human with a coherent solution that corresponds to the human's intent.

By analogy, the assimilator is somewhat like the President's chief of staff, deciding which of the very many, eagerly presented, inputs get through to the President. This should be done in an order and form most useful to the President's concerns. However, the ordinary query from the ordinary human is actually confronted with a much harder problem than that of the chief of staff. The latter may have to deal with several hundred or even a thousand potential inputs per day, from various sources of information. But, the ordinary person with an ordinary question is faced with many millions of potential inputs, and growing daily. This is far beyond the ability of any human "chief of staff" to manage.

The task of gathering information from different sources is performed by many real-world applications, which are sometimes referred to as information integration applications. For instance, if we want to buy a plane ticket for Chicago on Saturday, we would go to a site, like Expedia, to pose our query and get the price for all available flights from different airlines, e.g. United, Delta, American Airlines, etc. Similar to the Expedia site for the airline domain, there are also online shopping (ecommerce) sites, like pricegrabber.com, which sell products on the web from hundreds of vendors, through a uniform query interface.

Example: Consider the following motivating example, which illustrates the applications of information gathering on the Web. Barbara is a sophomore math major at the University of Maryland. She is considering taking a math course next semester. She also has access to The Automated Information Assimilator (TAIA). She asks, "What Algebra course will be offered next semester?" There could be many different courses, named *Algebra*, which are being offered by different universities. These courses may lead to results that are not what Barbara intended. TAIA would need to only return the results that match Barbara's intention. Also, there is an issue of time, i.e. TAIA would need to know when the courses are being offered, in order to only return the courses that will be offer "next" semester. TAIA would need a correct interpretation of current time, so that it can return accurate results. Barbara, however, may have different questions about various things. For instance, right now she wants to find out about *Algebra* courses offered for the next semester. Later on, she may want to know what flights to Chicago are available for this weekend. Still later, she may want general information about Rhesus monkeys, Michael Jordan, and then about a particular local restaurant.

Currently, for each of the above topics, there are useful websites (e.g. an online campus schedule of courses, Expedia, Wikipedia, campusfood.com, etc.) that may provide some answers. In other words, there are general-purpose online information sources (e.g. Wikipedia), databases specific to a particular narrow topic (e.g. a campus course schedule), and sometimes even a specialized collection of databases for a given topic (Expedia, campusfood.com). Expedia has the additional feature of handling queries across databases -- Barbara need not specify an airline -- whereas for campusfood.com, she is directed to a particular campus and cannot easily get information about, for instance, French-Japanese fusion restaurants near West Coast campuses.

These websites, however, have some limitations. The questions and answers are often not in natural language. Also, these sites usually use relational databases, and not more expressive data representations, i.e. ontologies. More importantly, current sites only work for questions that have been anticipated by the site's developers. In addition, the sites only answer questions that are related to a specific domain. Barbara has to juggle different search engines and/or databases: Expedia, campus schedules, Wikipedia, and campusfood.com. This is just the beginning: other questions she may have will require additional resources. Instead of having the user know about all the (changing) engines and databases, why not automate this? Why not a general-purpose integrated multiple-database search -- combining the features of Wikipedia and Expedia: The Automated Information Assimilator? TAIA would free the user from any need to know about what sources and query interfaces are available, let alone which pertain to which topic.

Building TAIA presents a whole host of issues. The Web overall is far too large for any set of mappings to be even remotely close to being complete. The Web is largely unstructured, so that disparate sources (sites) may be in various forms (e.g. natural language, images, tables, etc.). Meanings of a given expression can change from one source to another, within one source at different times, and even within one source in different contexts. The aims of the user (who queries the system) are not always clear -- intentions sometimes must be inferred based on world-knowledge and context that are not present in the query itself. The enormous abundance of data on the Web can lead to more results than a human user can digest, which raises the need for appropriate ranking and summarization. The results may be in so many disparate forms that they will confuse the user, again requiring suitable "packaging" and/or explanation. There may be significant gaps, redundancies, and inconsistencies in the available data, which might mislead the user, if not pointed out.

These days, a considerable amount of data is available on the Web. Consequently, the problem of answering questions using data sources on the web, is gaining great interest and attention in the database community. This interest may also have been fueled in part by the Semantic Web (Data Web) and Linked Data research efforts. For example in 2009, the 35th International Conference on Very Large Data Bases (VLDB) dedicated two panels to this problem. One panel was titled "Answering Web Questions Using Structured Data - Dream or Reality?" and the other was titled "How Best to Build Web-Scale Data Managers?"

In a recent work, van Harmelen states that with the rapid growth of the Internet and the Web, more principled mechanisms to facilitate semantic interoperability (i.e. facilitate querying of data) across organizational boundaries have become necessary [van08]. He emphasizes that despite many years of work on the semantic interoperability problem, this old problem is still *open*, and has acquired a new urgency, now that physical and syntactic interoperability barriers have largely been removed. Physical interoperability between systems has been solved with the advent of hardware standards, such as Ethernet, and with protocols, such as TCP/IP and HTTP. Also, syntactic interoperability between systems has been largely solved by agreeing on the syntactic form of the data that we exchange, particularly with the advent of eXtendible Markup Language (XML). For semantic interoperability between systems, we not only need to know the syntactic form (structure) of the data, but also the intended meaning of the data. Note that the skeleton (refer to Chapter 3) is a solution to the semantic interoperability problem, i.e. the skeleton enables users to query the data across organizational boundaries.

2.2. Toward a Solution

Ideally, TAIA would perform the following tasks:

(i) Accept and parse a question in ordinary natural language,

(ii) Narrow this down to an interpretation of what the human is really asking (her intention),

(iii) Decide what sources are most relevant,

(iv) Access those sources and cull through their responses for useful results,

(v) Package those results succinctly, and

(vi) Format and present it back to the human in natural language.

Figure 2.1 shows the schematic view of TAIA.

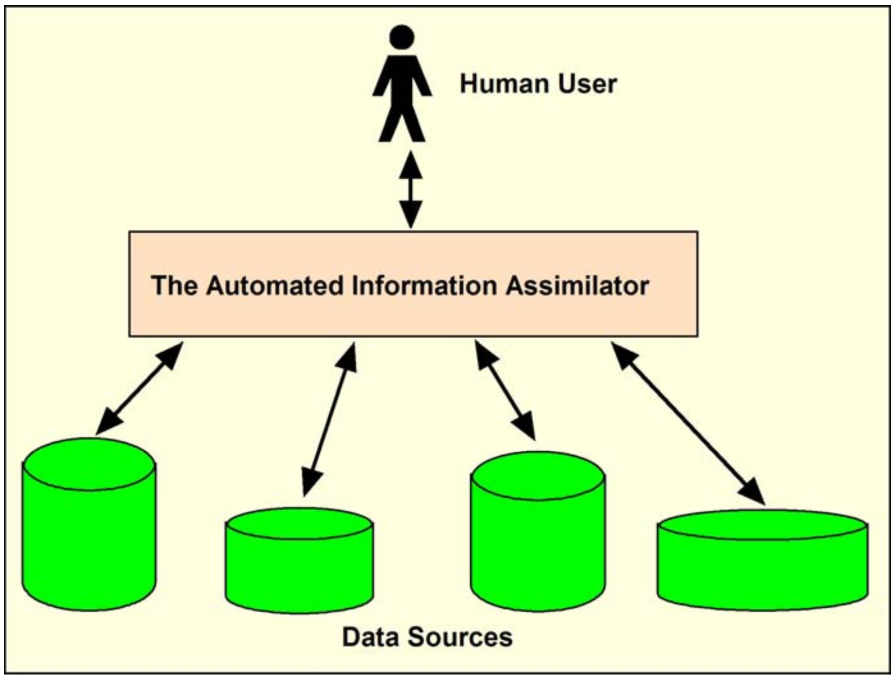

Figure 2.1. A schematic view of The Automated Information Assimilator (TAIA).

Let us see what these six tasks amount to in terms of the state of the art today. Tasks (i), (ii), and (vi) are related to natural language processing problems, on which much progress has been made. Of these, (iii) and (iv) are perhaps the thorniest at present, and requires more work for creating more useful assimilators. Tasks (iii), (iv), and (v) require "world knowledge" about existing sources and their levels of relevance and accuracy, and also a solution to "the mapping problem:" different sources tend to use different formats and terminology, making it hard to know when *Abstract Algebra* from *University of Maryland* means the same as *Modern Algebra* from *Stanford*. This problem also crops up in (ii), since the human may also use different forms and terms from other humans or even from himself at some other moment; as such this is also an NLP problem: that of word sense disambiguation. For tasks (iii) and (iv), TAIA may need to deal with the redundancy of results from different sources, inconsistency of results, and also gaps, i.e. information that could be missing.

The human questioner may at times want not simply some general information, but details about some specific one, which may well be reliably available only from special sources; thus if TAIA simply jams together everything that comes back from a query on *Algebra*, much of it may be irrelevant or misleading. In particular, if one wants to know whether *Algebra* is a required course for Maryland math majors, the information about *Algebra* for math majors at other institutions is not relevant; hence queries sometimes either explicitly or implicitly contain contextual constraints that the assimilator may have to ascertain, as in task (ii). In addition, TAIA must have access not merely to a vast compendium of courses, but of where a given course is taught -- which bears on tasks (iii) and (iv). While this may sound simply like more data, traditional data representations often make this difficult or impossible.

This dissertation, in Chapter 3, primarily focuses on deciding what sources are most relevant to a user's query and accessing those sources and culling through their responses for useful results, i.e. tasks (iii) and (iv). In Section 2.6, we also study the narrowing down of the question according to user's intent and packaging of results, i.e. tasks (ii) and (v).

The DeepQA project is a recent research effort at IBM that is relevant to TAIA [Dee]. It shapes a grand challenge in Computer Science that aims to illustrate how the wide and growing accessibility of natural language content and the integration and advancement of Natural Language Processing, Information Retrieval, Machine Learning, Knowledge Representation and Reasoning, and massively parallel computation can drive open-domain automatic Question Answering technology to a point where it clearly and consistently rivals the best human performance. A first stop along the way is making a formidable Jeopardy contestant named Watson. Jeoperdy is a game in which humans compete against each other to answer a series of questions correctly and quickly.

2.3. Natural Language Interface

In the Barbara example in Section 2.1, task (i) is to accept and parse a question in natural language from user. In this section, we further investigate task (i) and describe how a dialogue agent can parse the user's utterance. In essence, TAIA works in a fashion that is similar to a dialog agent, i.e. TAIA needs to engage the user in a dialogue and find the correct answer to the user's question.

2.3.1. Motivations for a Dialogue Agent

Software agents and computer systems are all around us nowadays and we, as humans, need to interact with such systems on a daily basis. These interactions are rapidly increasing, especially with the spread of pervasive and ubiquitous computing paradigms. In many cases, we do not sit in some specific place to interact with a computer that has the traditional keyboard and monitor. Ideally, we would like the human-computer interaction to be a constituent part of our daily activities. The most intuitive way of communication between human beings is via a conversation and other artificial forms of communication with devices (e.g. configuring and programming them) are usually cumbersome. For example, consider an automobile that turns the stereo on and off or adjusts the temperature, by receiving commands, instead of pushing of buttons. This interaction can be more than issuing of simple commands, and could be a robust dialog with an agent, with the objective of "the agent realizing the needs of the human user."

Applications of a dialog agent that converts user utterances into machine understandable commands are endless. Some examples of these applications are the following: A robot that provides services to patients in hospitals or performs routine tasks for the elderly at home, would be much more usable, if it interacts with ordinary people through a meaningful conversation; An online shopping bot that interacts with a user to determine his preferences and then finds the requested item, by searching various sites; A PDA and schedule planner that communicates with users through a built-in dialog agent; A GPS route planner that negotiates with users, about different routes and priorities, instead of the user trying to find out how to operate the device and configure his preferences. Regardless of how tech-savvy we are, we have all had the personally frustrating experience of figuring out the way to operate some new device, flipping through user manuals, and asking technicians for support.

A basic dialog agent must deal with syntax, which determines the structural relationship between words. A more flexible system also involves semantics, which is knowledge about the meaning of words, usually represented using a lexicon or dictionary, i.e. a simple ontology. Humans have an amazing and innate ability to engage in free-ranging conversation. They have the ability to recognize an unknown concept and to engage in learning by listening, appropriate questioning, and venturing tentative opinions. This ability, also called conversational adequacy, has been studied by [Per98] and seems fundamental to human dialog and more generally to human reasoning. The principles of conversational adequacy are largely cognitive and not specific to conversation.

Considering the numerous applications and advantages of communicating with devices through a flexible agent, instead of learning the exact operation of a device by human, Perlis et al. are building a cognitive dialog agent that processes the human utterance, to disambiguate and make sense of the concepts that a human user relates to [Hai10b]. After processing user's utterance and collecting the required information, the agent sends the commands to the device. In essence, the dialog agent allows a more meaningful human-device interaction. Notice that semantic knowledge about concepts is usually represented using an ontology. In order for the dialog agent to have a meaningful communication with devices in various domains, it needs to have a mapping between the concepts that are understandable for the agent and the concepts that are understandable for the device. Therefore, the dialog agent requires an algorithm to find the matching concepts in two ontologies.

We design this essential matching algorithm for the dialog agent in Chapter 4. The effectiveness of the algorithm is evaluated experimentally through different experiments. Using this algorithm, the dialog agent interweaves the individual threads of meaning between the human and device. In fact, a correct mapping of concepts facilitates a "meeting of minds" and prevents miscommunication between human and device.

2.3.2. Overview of the Needed Architecture

Our design for the dialog agent is in the context of a much broader project to tackle the brittleness problem and build intelligent systems that are more robust [And05]. In this project, in order to make autonomous systems more robust and tolerant to perturbations, a metacognitive loop is built into the system, which monitors performance and alters its own decision-making components, when necessary [And08]. Contiguous to tackling the brittleness problem, one consideration in the design of our dialog agent is to create cognitively plausible natural language processing systems. In this section, we provide a brief overview of the system architecture, and how the dialog agent interacts with other components in the system. The general goal of the dialog agent is to facilitate the communication between human users and devices. We provide a novel algorithm for finding matching concepts in the agent's ontology and the device ontology.

Note that our design for the dialog agent is domain-independent, i.e. the dialog agent can be used for interacting with any domain. That is because the ontologies can model various domains and be specified using the RDF or OWL languages, for example. An RDF ontology mainly consists of three entities, namely instances, concepts and relationships. Basically, the things about which we want to represent knowledge are called instances. Instances are grouped into concepts. The relationships are specified among pairs of instances. More formal definitions are provided in Section 3.4 and also available in [RDFp, OWLg].

One of the domains and experimental test beds that we are using for the dialog agent is a Mars rover application. The general architecture of the system is depicted in Figure 2.2. A human user needs to interact with a robot (i.e. Mars rover), which is situated on Mars. The user interacts with the agent via a dialog, and the agent eventually converts user's utterances into commands that are comprehensible by the robot. The dialog agent essentially acts as a mediator, to facilitate the human-robot interaction.

The robot gradually creates and maintains a model of the environment (Mars), as it discovers new facts (shown in Figure 2.2). The commands of the robot are represented in the form of an ontology. The ontology contains a semantic representation of different information, for example, it specifies what are the various ways an action can be performed, what parameters and information are required for carrying out a specific action, what are the preconditions and considerations involved in planning for some course of action, etc.

The dialog agent is a complex component and handles many issues. It has a user model (lexicon) to keep track of user's utterances and requests (refer to Figure 2.2). The dialog agent starts with a simple ontology. It may augment the ontology with various terms, as the conversation proceeds, since the human may use some vocabularies in his utterance, which do not exist in the agent ontology. Over the course of the conversation, the agent receives user's utterances and processes them. The agent also asks further questions by using its ontology, to clarify user intensions and specify missing information (such as missing parameters). Finally, the dialog agent needs to map the concepts in its ontology to the concepts in the robot ontology, before sending commands to the robot for execution. Notice that the dialog agent relieves users from the burden of acquiring that knowledge, which is necessary for operating the robot (device).

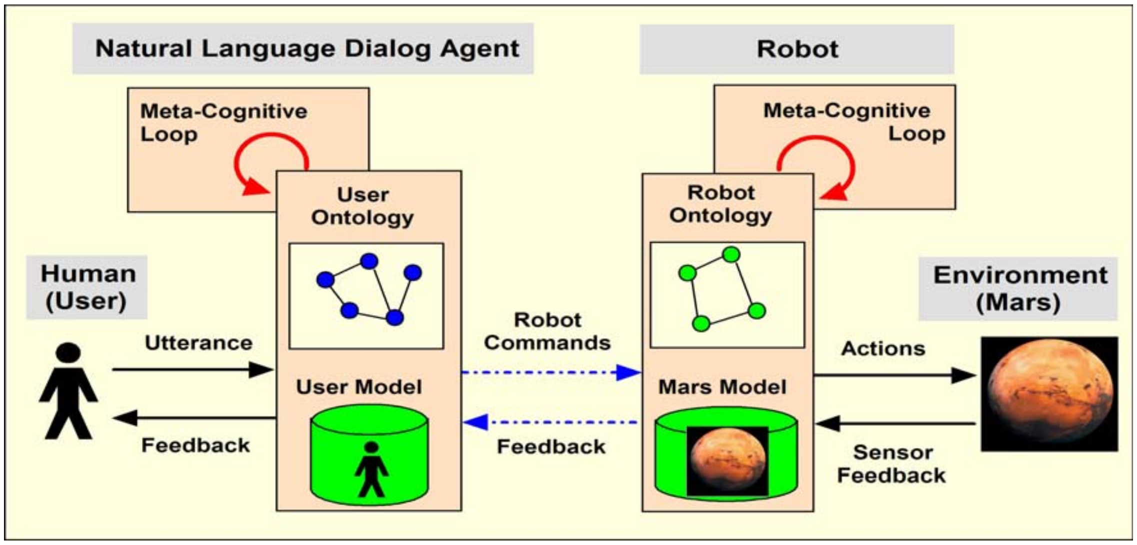

Figure 2.2. Overview of the needed architecture, which shows the communication between human user and robot via a dialog agent.

The dialog agent can similarly be used in the online shopping and e-commerce domain. In the Barbara example in Section 2.1, consider that Barbara specifies through a conversation, in natural language, with the dialog agent that she is looking for some "magazine." The dialog agent understands this concept, i.e. its ontology includes the term "magazine." When the dialog agent searches the ontology of different vendors on the Web, to find the requested item, in some other ontology the term "journal" may be used instead of "magazine". Now the dialog agent needs to match the two concepts using any available similarity metric, and this mapping of concepts is essential for communication between the human user and different vendors on the Web.

An effective algorithm for finding similar concepts in different ontologies should exploit various concept similarity metrics. The concept matching algorithm is provided in Chapter 4. In a nutshell, the dialog agent needs to discover the correct sense of a concept and disambiguate it, and the concept similarity metrics serve as evidence to decide whether two concepts match or not. This mapping of similar concepts is critical to achieving semantic convergence and attaching meaning to terms, which could be used differently in the agent's ontology and the domain (robot's) ontology. Sometimes, the concepts that are being compared may not even have similar lexical representations, while they are semantically equivalent, which makes the matching process more difficult.

In the Barbara example in Section 2.1, an example query in the university domain could be "What are all the courses that are offered in the Math department?"

TAIA would respond “The courses are Modern Algebra, Abstract Algebra, and Geometry.” Another example could be “Who is the instructor for Modern Algebra course?” TAIA would ideally respond “The instructor is Prof. Knuth.” However, in the latter example, the ontology may not contain the term “instructor.” This can be noticed by the metacognitive loop, and a response could be “I do not understand instructor.” The user can reply “I mean teacher.” TAIA would respond “The teacher is Prof. Knuth.”

2.3.3. Examples of Anomalies in Dialogue

Some of the anomalies that may arise between the user and the natural language dialogue agent, depicted in Figure 2.2, are as follows.

LEXICAL LEARNING (learning an unknown word):

USER: Move to the hilltop.

AGENT: What location is hilltop?

USER: Hilltop is at location 3.

AGENT: <moves rover to location 3>

NOVEL GRAMMATICAL CONSTRUCTION/ELLIPSIS:

USER: Explore crater.

AGENT: What location is crater?

USER: 5.

AGENT: <sends rover to explore location 5>

MILD GRAMMATICAL PROBLEMS:

USER: 3 Move to.

AGENT: Do you want me to move the rover to 3?

USER: yes.

AGENT: <move the rover to 3>

MISSING INFORMATION (learn default values):

USER: Explore.

AGENT: What location should I explore at?

USER: 3.

AGENT: <send rover to do science at location 3>

USER: Move to 3.

AGENT: <moves the rover to 3>

USER: recharge.

AGENT: <tells the rover to recharge at current location, 3>

COMPOUND TASKS:

USER: TakePanoAt 6

AGENT: What command is TakePanoAt?

USER: TakePanoAt is MoveTo and then TakePano

AGENT: <moves the rover to location 6 and takes a panoramic picture there>

REPORT CAPABILITIES (basic and situational):

USER: TakePano

AGENT: I cannot TakePano until I Recharge.

USER: What can we do at location 6?

AGENT: The rover can DoScience, Calibrate, or Recharge at location 6.

USER IDIOSYNCRACIES:

USER #1: MoveTo hilltop

AGENT: <moves the rover to location 1>

USER #2: MoveTo hilltop

AGENT: <moves the rover to location 2>

CONTEXTUAL POLYSEMY, SNYNONYMY:

This is similar to lexical learning.

ANAPHORA:

USER: MoveTo 3

AGENT: <sends rover to location 3>

USER: DoScience there

AGENT: <does science at location 3>

2.4. English Generation

In the Barbara example in Section 2.1, in task (vi), TAIA needs to format and present the results of a question back to the human in natural language. In this section, we further describe this task. For example, in response to a question like: "What are the courses offered in the Math Department at the University of Maryland?" the system can provide a list of courses, or it can generate a sentence in English and state that: "The courses offered at the University of Maryland are *x*, *y* and z."

Generating such English sentences may be desirable in various application settings, e.g. story telling, question answering for children, and generating news. Note that language generation can be essentially viewed as the reverse direction of information extraction from text. Each of these issues will be discussed next.

2.4.1. Information Extraction

It is generally accepted that the deep web (which contains structured and semi-structured data) is significantly larger than the shallow web (which contains unstructured or free-formed text). The issues discussed in Section 2.6 and Chapter 3 are addressing the access and retrieval of information on the deep web (also known as the Web of Data). Nonetheless, the unstructured information that available on the Web of documents (as opposed to the Web of Data) is useful as well. Unfortunately, this information is only comprehensible and accessible for humans (not machines), since it is expressed in natural language.

Today, what we can do with the prevailing state of the art technology is to use Web search engines (like Yahoo, and Google) to perform keyword search on the Web of documents. In Section 2.6, we demonstrate how semantic search is more robust than keyword search. However, in order to perform semantic search on top of natural language text (which is the ultimate goal of some search engine developers), we need to extract semi-structured information from the natural language text on the Web of documents.

For example, consider that Barbara wants to know about the “boring” courses offered next semester, in order to avoid those courses. It is quite likely that no officially announced, structured data source would contain such information. But there could be a blog by a student, named Sam, which provides information on this issue. Sam’s blog could read: “I have taken some course in the University of Maryland Math department. I think x, y and z are very boring courses. So much that I had a hard time staying awake, when taking those classes.” Now, we can process this unstructured information, and extract some triples in RDF (semi-structured format). If this semi-structured information is stored in an ontology, we can then use the techniques described in Section 2.6 and Chapter 3 to effectively access this kind of information, which originally used to reside in text. Note that information extraction is an active area of research, which involves language understanding and probabilistic models to deal with the uncertainty of the information being extracted from text [Sar08, Doa06, Etz05].

2.4.2. Language Generation

The task of generating natural language from a machine representation, such as a knowledge base or a logical form, is often referred to as Natural Language Generation (NLG). In some sense, NLG is similar to machine translation, as they may both need to convert a computer-based representation into a natural language representation (e.g. English sentences). Natural language generation may be viewed as the opposite of natural language understanding. In natural language understanding the system needs to disambiguate the input sentence to produce the machine representation. In natural language generation, the system needs to make decisions about how to put a set of facts into sentences.

In the Barbara example, if the underlying information being used, to answer a question, is represented in RDF format, then simple English sentences can be generated relatively easily by concatenating the subject, predicate, and object of the RDF triple. A triple in RDF could state that "Course x, isOfferedBy, Math Department." Concatenating these three parts creates a comprehensible sentence for the user. Of course, if the middle part of the triple (isOfferedBy predicate) has a label that is more user-friendly, the resulting sentence, which is generated by concatenating the three parts of the triple, would be more comprehensible to the human user.

Some other examples of natural language generation systems are the ones that generate letters in standard forms. Such systems do not typically involve grammar rules, but generate a letter to a consumer, e.g. stating that a credit card spending limit is about to be reached. More complex natural language systems dynamically create sentences to meet a communicative goal. As in other areas of natural language processing, this can be done using either explicit models of language (e.g. grammars) and domain knowledge, or using statistical models derived by analyzing texts written by humans.

2.5. Envisioned Architecture at User Level

Here we sketch the ideas that the next chapters develop, namely class matching and the skeleton representation. In Figure 2.3(a), users are situated at their organization and are able to use their ontology to search for items in their organization. However, they are not able to retrieve results from other organizations (ontologies). That is because the matches or correspondences between ontologies are not available. System administrators, who control these ontologies, can find the correspondences between them and create a skeleton to represent these correspondences. After the skeleton is created, users can retrieve more results, and the results come from different ontologies, as shown in Figure 2.3(b).

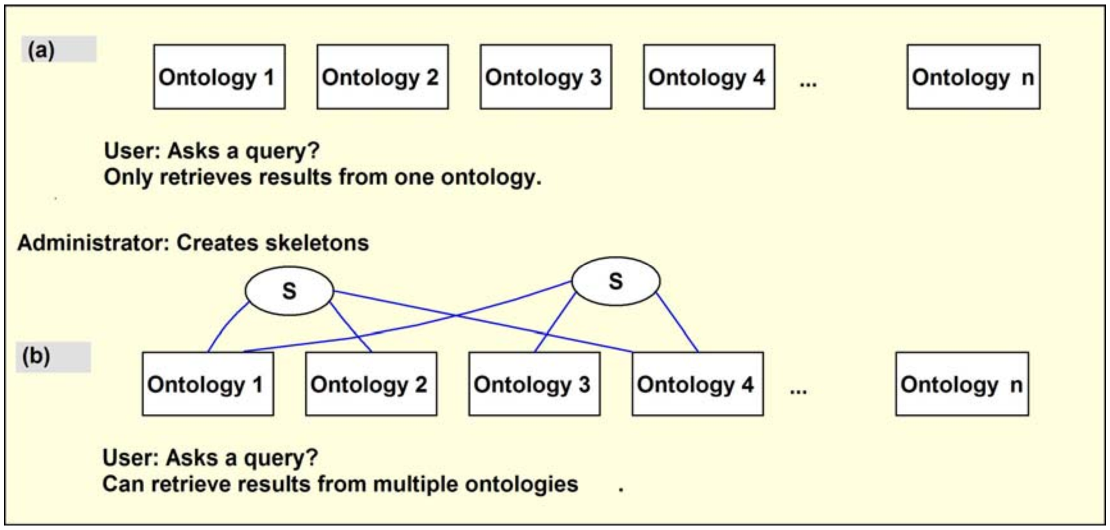

Figure 2.3: (a) Users querying their own ontology. (b) System administrator creates the skeletons, and then users can query multiple ontologies transparently.

In order to create the skeleton, the administrator can use two operators. The first operator is for finding the matches between two ontologies A and B, which is denoted as Match (A, B). The algorithm for this operation is presented in Section 4.1. The second operator is for creating a skeleton using the found matches, which is denoted as Skeleton (A, B). The algorithm for this operation is presented in Section 4.2. Note that the Match operator is commutative, i.e. Match (A, B) = Match (B, A). For facilitating interoperability between ontologies A and B, both Skeleton (A, B) and Skeleton (B, A) can be used, but the shape of the skeleton may be slightly different in these two cases, as discussed in Section 4.2. In other words, the Skeleton operator is not commutative.

2.6. User Intentions and Semantic Search

2.6.1. Overview

In the Barbara example in Section 2.1, in task (ii) for the information assimilator, Barbara needs to narrow down the solutions to a question according to her intention, in order to collect the necessary information. In this section, we further investigate task (ii) and describe how the interpretation of a query can be narrowed down accurately.

When using TAIA, Barbara could ask "What are all the courses related to Algebra, offered next semester?" As described in the work of Grice [Gri], what the question is stating is different from the semantics of this question (i.e. Barbara's real intention). A correct answer to this question may include a course named *Modern Algebra*, offered at *Stanford*. However, Barbara's intention is probably not all the Algebra courses that are offered in all universities in the world, i.e. Barbara is only interested in the courses offered at *University of Maryland*, since she is a student there.

We use the term *keyword search*, when the search is performed on data stored in the relational data model, as in traditional relational databases, and examples of keyword search in databases are [Hri02, Say07]. This should not be confused with the popular keyword search that is used in current Web search engines, like Google. Keyword search on the relational model is in fact inspired by the success and user-friendliness of keyword search in Web search engines, on the Web of documents.

We use the term *semantic search*, when the search is performed on data stored in the RDF data model. Note that when the data is modeled in RDF, it inherently contains explicit typed relations or semantics (refer to Section 3.5), and hence the use of the term “semantic search.”

The central idea of the relational model is to describe a database as a collection of predicates over a finite set of predicate variables, describing constraints on the possible values and combinations of values. The content of the database at any given time is a finite (logical) model of the database, i.e. a set of relations, one per predicate variable, such that all predicates are satisfied. A request for information from the database (a database query) is also a predicate.

The purpose of the relational model is to provide a declarative method for specifying data and queries: we directly state what information the database contains and what information we want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for getting queries answered.

Resource Description Framework (RDF) is a framework and W3C recommendation for representing information on the Web. More details about RDF are provided in Section 3.4. RDF is designed to represent information in a flexible way. The generality of RDF facilitates sharing of information between applications, by making the information accessible to more applications across the entire Internet.

With semantic search, user’s intentions can be accurately narrowed down in a more robust way than keyword search. Keyword search, popularized by the success of Web search engines, has become one of the most widely used techniques for finding information on the Web. However, the customary indexing of keywords, as done by Web search engines, is only effective on text Web pages (i.e. unstructured data), which is also referred to as the shallow Web. It is generally accepted that the deep Web, which contains structured and semi-structured data, is significantly larger than the shallow Web. In other words, considerable amount of data is “locked away” in databases in structured and semi-structured format.

In Section 2.6, we will focus on the deep Web and compare semantic search (in the RDF model) with keyword search (in the relational model), to illustrate how these two search paradigms are different. This comparison addresses the following important questions:

- What can semantic search achieve that keyword search can not (in terms of semantic search behavior)?
- Why is it difficult to simulate semantic search with keyword search on the relational data model (i.e. enabling features in RDF that explain the behavior)?

Let us begin with an example, to illustrate the differences between semantic search and keyword search.

2.6.2. Semantic Search

Recall from Section 2.1 that the *Algebra* course could appear in many forms, with different variations, all of which may not be relevant to Barbara’s question. Let us illustrate semantic search with another similar example. Consider that we want to know more about “Michael Jordan.” This entity of type *Person* could be the *Professor*, who teaches *Computer Science* and is affiliated with *UC Berkeley*. It could also be the *Basketball Player*, who plays for the *Chicago Bulls* and is in the *NBA* *league*. A close analogy from the unstructured Web of documents is that a Google search for "Michael Jordan" returns hundreds of pages, most of which are irrelevant to the Berkeley Professor. Most of the results and the top ranked ones refer to the *Basketball Player*, which may not be our intended entity for the search.

Although the use of additional terms like "Berkeley" will help us in finding our intended entity, we may not know which university he is affiliated with. We might actually be performing the search to find this piece of information. Figure 2.4 demonstrates the two different entities and the information related to these entities, in the RDF model. The nodes represent the entities. The edges represent the properties, which hold between the entities.

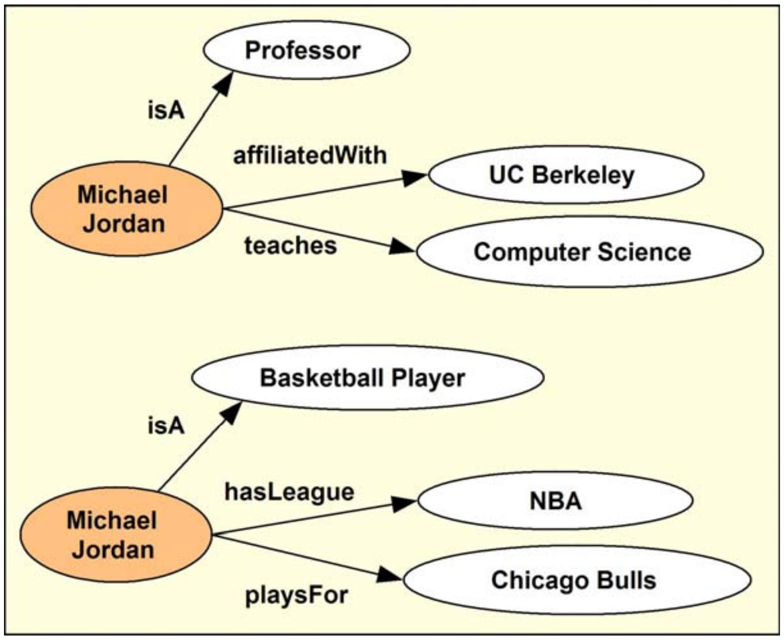

Figure 2.4. Two different entities with the same name "Michael Jordan," represented in the RDF data model. Each entity has a different set of properties and property values. The nodes represent the entities. The edges represent the properties, which hold between the entities.

With semantic search, in the RDF model, users can iteratively refine their search; navigate through the initial results and filter out the results (entities), which do not have the properties that they are looking for. In fact, the explicit representation of properties in RDF (which does not exist in the relation model) facilitates this refinement of search results. In Figure 2.4, the user could search for “Michael Jordan” and the instances, which match the search string, will be shown to the user. Now, since the user knows that he is looking for a *Professor*, he could select the *teaches* property from all the available properties, which refines the search to the entities that have a *teaches* property. This way, the *Professor* entity, which he is looking for, is found. If the user does not know what Michael Jordan teaches, he can find out the answer to this question by seeing the value for the *teaches* property, which is *Computer Science*. On the other hand, if he knows this fact, he can add the *teaches* property and the *Computer Science* property value, to further refine the result of the search if necessary.

Intuitively, humans specify their intended entities in this fashion. In other words, they define an entity by iteratively specifying extra properties about an entity, until the desired entity is uniquely identifiable, for example, “Michael Jordan,” the one who teaches Computer Science and is affiliated with UC Berkeley, etc. Subclasses and other properties help in the search refinement process. Moreover, once the desired entity is uniquely identified, we can browse its various unknown properties, depending on what property we are looking for. A number of interesting open source browsers for RDF data, which follow the semantic search paradigm, have already been implemented, e.g. [Ber06, Huy]. In spirit, these browsers enable users to navigate through "sets of entities of the same type" and gradually refine these sets.

2.6.3. Keyword Search

With keyword search in the relational model, it is very difficult to perform the semantic search behavior, which was described in Section 2.6.2. This difficulty is in part due to the fact that the semantics are not encoded explicitly in the relational model. Consequently, it is difficult to incorporate metadata information (i.e. column names) into the keyword search process, as described below.

While recent research in the database literature has attempted to retrofit keyword search onto relational databases, there are various scalability issues in performing keyword search. For now, ignoring the computational cost (which will be discussed later), in theory, a keyword search like "Michael Jordan" and "Computer Science" is possible. However, this keyword search can only be performed, when we assume that the user knows the property values (i.e. Computer Science) that would sufficiently refine the search. Clearly, this is not always the case. In other words, the user can not browse the available properties (like *teaches*) and navigate through sets of entities, as in semantic search. Notice that in keyword search, unlike semantic search, we are not dealing with the *teaches* property, and instead need to use *Computer Science*, which is a property value for the *teaches* property.

In addition to this limitation in navigation, performing keyword search on top of relational databases is computationally expensive, especially when the keywords (i.e. property values) appear in various tables, or there is a long list of keywords, since this requires many joins. In general, the keyword search process requires various steps, including: finding keys in tables, finding joinable attributes, generating foreign key join candidates, and removing semantically incorrect candidates [Say07]. Moreover, enumerating all possible candidate networks, which may contribute to the results, is computationally expansive [Hri02].

On the other hand, in semantic search, user's knowledge of the domain can be utilized effectively to navigate through sets of entities, and refine the search results. In fact, this "user-driven" navigation in semantic search, replaces the enumeration of candidate networks in keyword search (which is computationally expensive). In other words, while it is *unacceptable* to ask the user to provide a series of joins and SQL operations that are necessary for finding the correct results in the relational model, it is quite *acceptable* to ask the user to provide additional properties and property values to refine the search, as described in Section 2.6.2.

Another issue with performing semantic search on the relational model is related to physical implementation. The physical implementation of most relational databases follows their logical description, i.e. each table (relation) is stored in its own file, or collection of files, on disk. Such an implementation is effective for queries that filter, or aggregate, large portions of a single table. It provides reasonable performance for queries that join many tuples from one table to another table [Mar08]. However, this implementation is much less effective for semantic search, which requires join queries that follow paths from a small number of tuples of one table to another table. Note that semantic search queries try to accumulate facts about a small set of entities (e.g. all the cities in a country). Answering such queries requires one, or more, random I/Os for each table that is used in the path. Therefore, semantic search queries perform poorly on the traditional physical implementation in the relational model.

Aside from the explicit encoding of semantics, another advantage of the RDF model is the global *referencing* of entities on the entire web, which does not exist in the relational model. For example, the “Michael Jordan” entity, who is a basketball player, is specified by a link (or URI) and can be uniquely referenced by different organizations, across the web. Now assume that different kinds of information about the health information and the financial information of “Michael Jordan” are stored in different organizations. With semantic search in RDF, a user can access all this information from various organizations. Note that we are not designing or planning for any specific queries in advance, when the health and financial organizations are being constructed. In other words, the aggregation of information is achieved in an ad hoc manner.

The global referencing of entities in RDF is vital to facilitating interoperability and aggregation/reuse of knowledge across organizational boundaries. Consider that in the relational model, facilitating interoperability across distributed and heterogeneous databases is quite difficult, which is partly due to the lack of such a referencing mechanism.

2.6.4. Research Challenges

Storage and retrieval of data in the relational database model has become highly optimized over the last three decades. Similar performance levels are necessary for

RDF to enable large-scale semantic search. Considering that the representation of data in the RDF and relational models are different (refer to Section 3.5), there are fundamental database research issues that need to be studied for achieving better performance. The semantic search example, described in Section 2.6.2, clarifies some of these issues, which include: native storage mechanisms for RDF or efficient storage of RDF in the relational model, indexing and retrieval of RDF data, optimization of queries specified in SPARQL, and ranking of entities in search results.

Research efforts along this line are underway, e.g., [Aba07] studies the physical design issues and proposes the vertical partitioning of data in the relational model, for RDF storage. [Neu09] focuses on join processing, since the fine-grained and schema-relaxed use of RDF often entails star-shaped and chain-shaped join queries with many input streams from index scans.

From a human-computer interaction standpoint, there are several issues that need to be studied to enable semantic search, including: effective presentation of sets of entities, user interactions to support the selection of some of the entities from the result of a query, presentation of relationships (at both class and instance level), and intuitive interfaces for specifying SPARQL queries (perhaps similar to query-by-example models). Each of these issues becomes clearer, when considered in the "context" of the precise examples, provided in this section, which describe the desired semantic search behavior.

2.6.5. Summary of Semantic Search

With semantic search, user's intentions can be accurately narrowed down in a more robust way than keyword search. The comparison of semantic search in RDF and keyword search in relational model, in this section, clarified the advantages of using RDF instead of the traditional relational data model. We demonstrated that it is difficult to retrofit a robust semantic search *behavior* on the relational data model and find answers to questions about an entity (as available in RDF). In semantic search, typical users (who have not had any special training) can *interactively* explore the data and navigate through sets of entities, by utilizing their knowledge of a domain. Also, they do not need to use unintuitive and complex SQL statements (as in the relational model) – statements which are only understood by people with database training.

We demonstrated why it is difficult to simulate semantic search with keyword search on the relational data model. In other words, the explicit encoding of semantics (via typed relations) and the global referencing of entities in RDF (via links or URIs) are the two critical enabling *features* that make RDF suitable for robust search and information integration across different enterprises. The comparison also revealed some of the crucial research challenges that need to be addressed for scalable semantic search.

2.7. Summary of TAIA

Considering the numerous data sources that are becoming available on the Web, the problem of answering questions using this data is gaining great interest. In this chapter we provided a detailed map of the various components that are necessary for creating The Automated Information Assimilator that is a gofer between humans and the information. TAIA would get the queries from a human and then gather information from all relevant sources, by culling through it, as accurately as possible. It pulls together all that bears usefully on what the human wants to know and provides the human with a coherent solution that corresponds to the human's intent.

We described the design issues of a natural language interface for TAIA, in the context of a broader project for building a dialogue agent. The dialogue agent engages the user in a conversation, parses the user's utterance, and then sends commands to a robot. This robot can execute different tasks in the Mars domain. We also provided some examples for the anomalies that arise in a dialogue with a robot, in the Mars domain.

TAIA does not yet exist. Its full design and implementation present a very challenging task involving many advances. This dissertation primarily addresses a few of these advances in Section 2.6 and Chapters 3 and 4. In Section 2.6, we will study the narrowing down of a question according to user's intent and packaging of results. In Chapters 3 and 4, we focus on deciding what sources are most relevant to a user's query and accessing those sources and culling through their responses for useful results.

Chapter 3: ONTOLOGY MAPPING

In the Barbara example in Section 2.1, in task (iii) and (iv), the information assimilator needs to decide what information sources are most relevant to a query, and then access those sources and cull through their responses for useful results, in order to collect the necessary information to answer the query. In this section, we further investigate these two tasks. In order to find the relevant sources of information for answering a query, we need to map between different ontologies, in which the information resides.

3.1. Overview

As more and more information goes online, the capacity to answer queries using these distributed data sources becomes more important. In the Barbara example in Section 2.1, TAIA needs to gather information from these sources in a general way, and not with a particular query in mind. It should provide access to data, and also integrate results, from many relevant sources. However, the data sources are not static; in particular, they can *grow*, by incorporating additional classes, properties, and instances, often borrowed from other data sources. For instance, data source *A* might be enlarged to include aspects of data source *B*. (This is sometimes called merging, although the end result is not that *A* and *B* become a single combined data source; rather it is *A* that grows by copying some aspects of *B*, while *B* remains unchanged.) In order for such merging to occur, it is necessary for entities in *B* to be properly related to ones in *A*, in particular so that "matching" entities are determined (the *movie* class in *A* might be matched with the *video* class in *B*, etc.).

A similar issue arises in integrated multiple-database search. In order for the system to adequately respond to a *query*, it must relate entities across the multiple databases that it consults. The query "Which videos did John Wayne appear in?" will have to treat *video* in *B* and *movie* in *A* as the same concept. This is sometimes referred to as the "integration" part of integrated multiple-database search. Since in general any given search engine has no control over the organization and terminology used within databases (which typically are not owned by the owner of the search engine), the search engine must perform "matching" (i.e. interpret the disparate data), to obtain the most accurate and complete results relevant to a given query -- not to mention that this must be repeated frequently, to keep up with changes within individual data sources.

However, despite the similarity of the growth and querying issues, which both depend on matching, these are highly distinct applications:

(i) Growing one data source (*A*) by incorporating aspects of another (*B*), after determining suitable matches between the two.

(ii) Extracting query-relevant data from two data sources (*A* and *B*), again after determining suitable matches the two (e.g. video and movie).

Therefore, matching is fundamental in different applications of multiple data sources, whether for the purpose of building a larger combined data source, or for responding to a query by consulting multiple data sources. Yet, these two applications are very distinct; “growing” is a data source development task, whereas “extracting” does not change, build, merge, or enlarge any data source, but rather allows sensible simultaneous use of data sources -- what is known as interoperability. (More generally, interoperability of systems refers to their simultaneous use without conflicts of terminology, access policies, resources, etc.)

Let us sum up via an example. On one hand, each airline maintains its own data source of flight information, which not only changes specific flight instances due to changes in scheduling, but also which can dynamically add, remove, or alter classes and properties (via merging from another data source, if for instance one airline buys out another through acquisition), which requires careful attention to interpretation issues. On the other hand, TAIA (similar to Expedia) provides interoperable search across these disparate data source of different airlines, and also depends on the same kinds of interpretation issues, yet while not altering the data sources at all.

Hence, integrated across-database interpretation is fundamental and has two fundamentally different use cases with respect to multiple databases: internal database development (merging, with “write-permission”), and external “read-only” access to databases. Expedia is an example of the latter; it does not change the data, and does not even have permission to do so.

Consider another example, outside the realm of computing, where the interoperability problem arises. In the 19th century many different railroad transportation companies started, but they tended to use different "gauges" (widths) of track for their own trains; this created a mess -- to ship something from New York to California, one might have to use perhaps four companies (e.g. New York to Pittsburg, Pittsburg to Chicago, Chicago to Santa Fe, and Santa Fe to Los Angeles), because no one company had tracks that went the whole distance. This would not be such a problem if the gauges were not different: one could use the same train all the way and just pay the companies for whatever length that their track was used. But because the gauges were not the same, one had to change trains each time, meaning moving the shipment from one train to another four times, which was a huge cost in labor and time. In other words, the rail systems were not *interoperable* except at huge inefficiency. Finally, as a solution, the companies agreed on a gauge and rebuilt their tracks and trains so all were compatible (except for the lucky few that already had that gauge). In essence, by standardizing the gauges in this setting, the trains were able to move and operate across different railroad transportation companies (i.e. organizational boundaries). In the next section, we explicitly define the ontology mapping problem.

3.2. Ontology Mapping: Problem Definition

The "ontology mapping" procedure for two separate and autonomous ontologies, O_1 and O_2, consists of the following steps:

- Step 1: *Finding* corresponding entities in ontologies O_1 and O_2.
- Step 2: *Representing* the found correspondences, and *using* it to achieve some goal.

For Step 1, the main ontology entities that can be considered, when finding correspondences between ontologies O_1 and O_2, are: classes (concepts), individuals (instances), and properties (relations). For Step 2, for using the found correspondences, the correspondences need to be represented in a suitable form.

Note that the goal of ontology mapping determines what candidates to consider, when we are finding the correspondences. The goal also determines how to represent the correspondences. This definition of ontology mapping is overarching, such that it encompasses the different goals of the problem. This definition does not imply any transformations from a source to a target, as described in Section 4.2.

The terms “alignment,” “matching,” “merging” and “integration” have also been used in the literature, with different interpretations. In the rest of this dissertation, (i) we will avoid using “alignment,” (ii) “matching” is the process of finding “correspondences” between two entities, and (iii) “merging” and “integration” of two ontologies have the same meaning and might be somewhat vague now, but will be clear, after we state the goals of ontology mapping in Section 3.3.

3.3. Goals of Ontology Mapping

Currently, there are various ontologies that are being used in different organizations. Often, they have been designed by different communities. Hence, there is a need for a mapping between these ontologies. Based on the definition of the ontology mapping problem (refer to Section 3.2), in order to lay out the foundations of the problem, we start with the goals of ontology mapping. We identify two quite distinct goals for ontology mapping, based on real-world use cases, and illustrate them with motivating examples.

One possible goal of mapping is ontology development; that is when an ontology is being designed or engineered by an organization. The other possible goal of mapping is interoperability; that is when there are various parties, which are using different ontologies, and the users need a mechanism to be able to query the information, which resides in the ontologies. Many ontology mapping applications fit into one of the two goals above, even if the terminology used in some application domain is slightly different.

3.3.1. Ontology Mapping for Ontology Development

Ontology is an abstraction for representing conceptual knowledge. All concepts are covered by the domain of human knowledge and these concepts are connected together in some fashion. Hence, it is hard to limit an ontology in terms of what it should represent. This decision is usually made, based on the business needs of an organization, i.e. the ontology designer decides not to include some concepts, as they seem irrelevant to current organizational demands. Ontology design (also known as ontology development/engineering) is a complex and subjective issue, similar to database design, and requires a human in the loop.

Assume that an organization is currently using an ontology, *C*. Over time, as organizational models change, business processes evolve and are extended. Therefore the ontology *C*, which models one organization's business processes, also needs to be changed and often extended. Sometimes, the new business models (or some fragments of the changes) that are required in the current ontology have already been captured by ontologies that are being used in other organizations. In this case, the required extensions to the current ontology *C* are present in some other existing ontology, *E*. Now, the ontology designer of *C* needs to:

- Step 1: Find the correspondences between ontologies *C* and *E*
- Step 2: Decide on what concepts, instances and relations of the existing ontology *E*, need to be added to the current ontology *C*, based on the changes in the business model and the correspondences found in the previous step.

Note that the existing ontology *E* remains the same, while current ontology *C* will change (and be replaced in fact). This use case closely resembles the problem that has been analyzed in the context of merging/integrating two ontologies in the literature [McG00, Noy00, Stu01].

Motivating Example 1: Consider two organizations (supermarkets), Org_1 and Org_2, offering various products, and using two different ontologies, O_1 and O_2, shown in Figure 3.1. O_1 is shown with white rectangles, while O_2 is shown with grey rectangles. Some classes (i.e. concepts), namely *Sale Items* and *Videos*, in O_1 have corresponding classes (*Products* and *Movies*) in O_2. In Figure 3.1, since class *Videos* in ontology O_1 is defined in a similar context to class *Movies* in ontology O_2, it is conceivable to merge the two ontologies and produce a more comprehensive ontology. In essence, O_1 is being extended with O_2 and the merged ontology is a mix of white and grey rectangles, as shown in Figure 3.1.

It is crucial to note that in our scenario, the business model in Org_1, which was using ontology O_1, has changed. For example, Org_1 gradually needs to develop a more comprehensive ontology for its ecommerce operations. Therefore Org_1 will now be using the merged ontology O_{merged}, which is the result of extending O_1 with some existing ontology O_2. This is while Org_2 will keep using O_2 without any changes to its business model.

This is not relevant to the "ontology development" scenario, however, consider that if Org_2, which is now using O_2, also plans to use the merged ontology (for some reason, perhaps for "interoperability" with Org_1), then *both* organizations (Org_1 and Org_2) will need to make changes in their operations, to use the merged ontology O_{merged}. Furthermore, merging can be problematic, if the ontologies are defining the classes in different contexts, as merging would easily lead to irresolvable inconsistencies. Hence, "interoperability" should be facilitated by means other than merging, as examined in Section 3.3.2. We return to these implications and elaborate in Sections 3.3.3 and 3.3.4, after introducing the second goal, in Section 3.3.2.

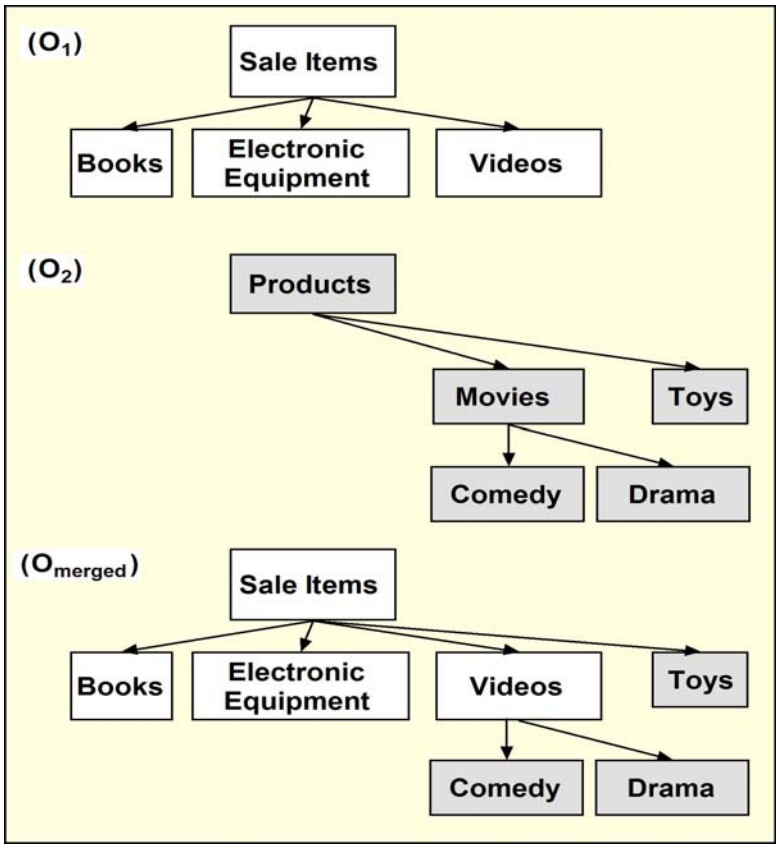

Figure 3.1. Two ontologies O_1 and O_2, and the merged (integrated) ontology O_{merged}.

3.3.2. Ontology Mapping for Interoperability

Different enterprises use their own proprietary systems and are usually not willing to change their business models and operations. However, they also need to exchange information with other enterprises. Hence, interoperability between enterprises needs to be facilitated across organizational boundaries. That is, in many circumstances, users need to query different ontologies (distributed and autonomous sources of information) and retrieve data from all of them, as if all the information is residing in a unified source.

Let us define this scenario more formally. Two different ontologies, O_1 and O_2 are designed separately and are being used by two autonomous organizations, also known as parties. Each ontology is designed based on the business model that governs the operations of the organization that it belongs to. Hence, the ontology being used by each party can not be changed or extended, as we did for the merging use case in Section 3.3.1. To facilitate interoperability between the organizations in this scenario, two steps are required:

- Step 1: Finding the correspondences between ontologies O_1 and O_2
- Step 2: Representing the correspondences in a suitable structure, which we call skeleton *S*.

The skeleton is described using another motivating example, next.

Motivating Example 2: Consider two universities in which faculties, and departments within the faculties, are organized differently. Ontologies O_1 and O_2 are shown in Figure 3.2(a) and 2(c), respectively. Ontologies O_1 and O_2 represent the organizational hierarchy of $University_1$ and $University_2$, and are depicted with rectangles. There are six corresponding concepts in O_1 and O_2, namely: *University*, *Science*, *Maths*, *CS*, *Physics*, and *Chemistry*, shown with a white color. These six concepts appear in different places in O_1 and O_2. The skeleton *S* consists of these six concepts, as shown in Figure 3.2(b), and depicted with ovals.

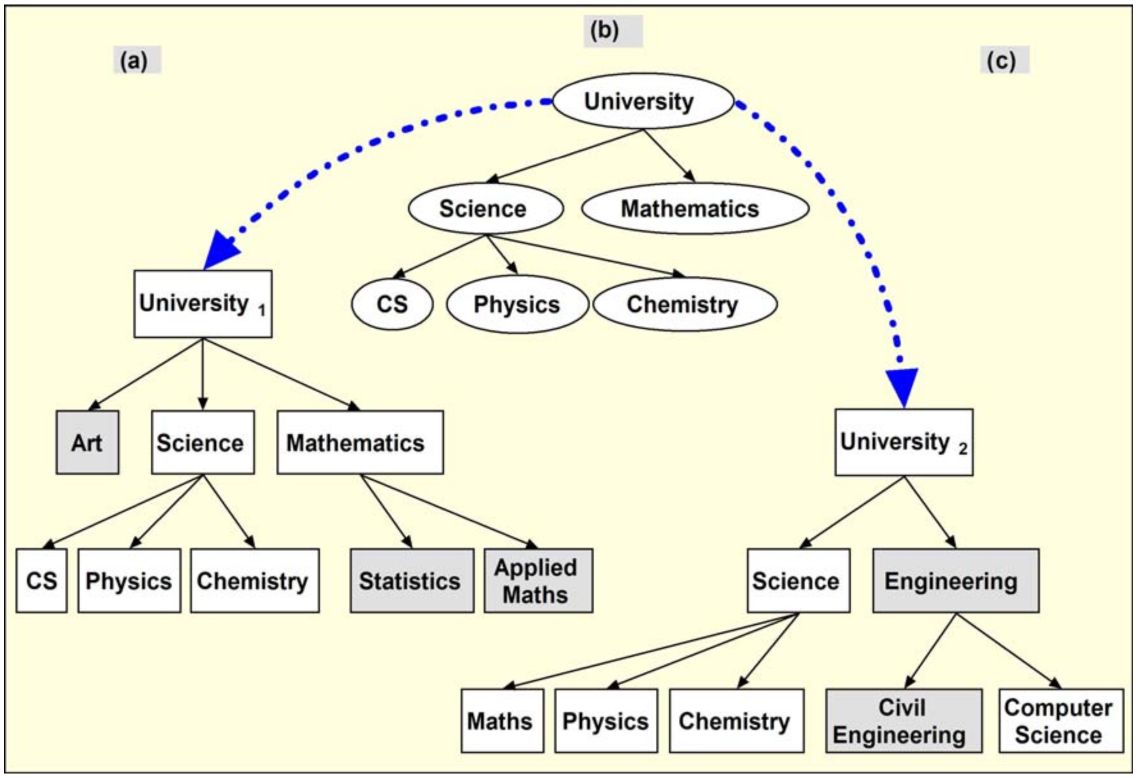

Figure 3.2. Ontologies O_1 and O_2, which belong to two different autonomous organizations, are shown in (a) and (c). Skeleton *S*, connecting the ontologies, is shown in (b), in the middle. The concepts in ontology O_1 (shown in the figure) are the organizational units within *University*$_1$. The instances in ontology O_1 (not shown in the figure) are the courses that are offered by the organizational units within *University*$_1$. Each concept in skeleton *S* is connected to its corresponding concepts in the original ontologies O_1 and O_2, with a subclass relationship.

When creating a skeleton, first, we need to know the shape (i.e. class hierarchy) of the skeleton. The shape of the skeleton governs the relationship between the concepts in the skeleton. The shape of the skeleton is determined by the ontology of one of the parties (i.e. O_1 or O_2). In Figure 3.2, the shape of the skeleton is the same as ontology O_1. Each concept in skeleton *S* is connected to its corresponding concepts in the original ontologies O_1 and O_2, with a subclass relationship.

Note that Figure 3.2 shows such connections for the *University* concept, only. The *University* concept in the skeleton is connected to concepts $University_1$ and $University_2$ in ontologies O_1 and O_2, with blue dotted arrows. Other such connections are not shown in the figure for more readability. In Example 2, the ontology of each organization (i.e. O_1 and O_2) remains intact, unlike Example 1. There is no change in the business models (structures) of the universities, at all. However, both universities (parties) can be queried using the skeleton, which provides interoperability between them, as described next.

3.3.2.A. Multiple Ontology Case:

In Figure 3.2, we create a skeleton for two ontologies, and the skeleton enables the querying of the two systems and facilitates interoperability between them. The algorithms for finding the matching classes and creating the skeleton are presented in Sections 4.1 and 4.2.

If there are more than two ontologies involved, we can similarly create the skeleton for pairs of ontologies. For example for ontologies A, B, and C, we can create Skeleton (A, B), and Skeleton (B, C). Then another skeleton can be created for the resulting skeletons, i.e. Skeleton (Skeleton (A, B), Skeleton (B, C)), as demonstrated below. The resulting structure enables users to start from their own ontology and move up to the skeletons to retrieve results from other ontologies, when there are more than two ontologies involved. This is shown in Figure 3.3(a).

An alternative that we call n-mode is to create one skeleton any matching within the ontologies, i.e. Skeleton (A, B, C). Then the skeleton would be connected to each ontology and enable the users of each ontology to retrieve results from the other ontologies. This is shown in Figure 3.3(b).

The algorithm for finding the common classes between ontologies and creating the skeleton can easily be performed on more than two ontologies, similar to the case of two ontologies. Any matches between classes in the prime ontology and any other ontology are recorded. Then skeleton can be run over and over or just once, as shown in Figure 3.3. More details about this will be discussed later.

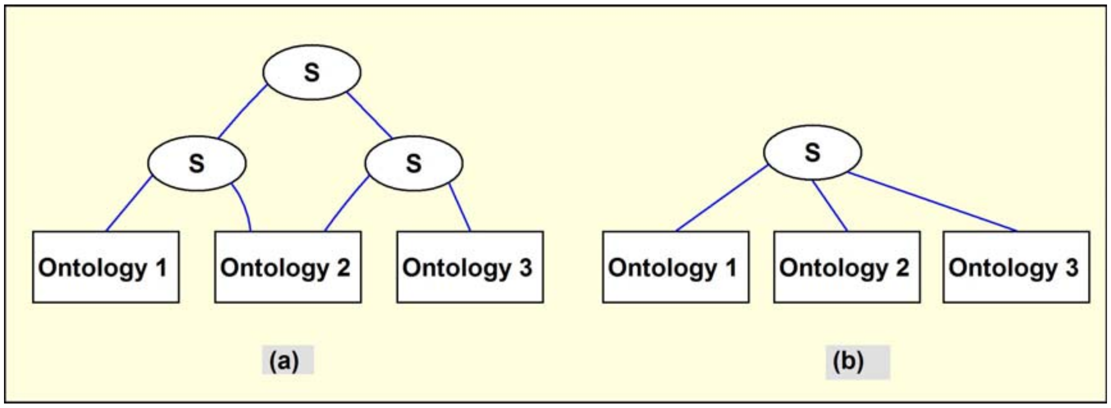

Figure 3.3. Different ways of creating the skeleton for multiple ontologies.

3.3.2.B. All Subclasses Option for User:

In Figure 3.2, in ontology O_1 Science and Mathematics are in the same level, but in ontology O_2 Science is a superclass of Maths. Now consider that a user queries ontology O_2 for Science and expands that query to also get Maths instances in ontology O_2. Then the user moves to ontology O_2 to get corresponding instances of Science, he will not see instances of Mathematics under Science in ontology O_1.

If the user wants to see those instances of Mathematics under Science in ontology O_1, then each of the subclasses of Science in ontology O_2 should be considered separately, and for all those subclasses, we can move across to ontology O_1, using the skeleton, and retrieve the corresponding instances for those subclasses.

3.3.3. Contrasting the Goals at a Glance

In Sections 3.3.1 and 3.3.2, we clarify that interoperability may not always be facilitated by ontology merging. This clarification is a side effect of distinguishing the goals of ontology mapping. In principle, the two use cases in Sections 3.3.1 and 3.3.2 are very different. In Figure 3.2, the concepts in ontologies O_1 and O_2 are the organizational units within $University_1$ and $University_2$. Each concept contains various instances. The instances are the courses that are offered by an organizational unit (concept). For example, the *Computer Science* department (concept) in O_2 contains the courses (instances) that are offered in that department.

Here, we describe interoperability explicitly. In the interoperability use case, we would like to query for all courses related to computer science, and retrieve the results from both universities (i.e. across organizational boundaries). In Figure 3.2, with the skeleton, we can query for *CS* courses in ontology O_1, and using query expansion, we move to the corresponding concept in the skeleton (which is *CS*), and then also retrieve the relevant courses from the *Computer Science* concept in ontology O_2. Therefore, the query would return the results, as if all data resides in a unified source. In essence, the skeleton increases the recall of queries by enabling users to retrieve results from distributed sources, under a unified framework.

In Figure 3.2, let us assume (incorrectly) that we want to merge the ontologies (*O_1* and *O_2*) to facilitate interoperability. Consider that course *abc* is offered in the *CS* department in *O_1*, while a different course, named *xyz*, is offered in the *Computer Science* department in *O_2*. Merging of these two departments by stating that the two concepts (*CS* and *Computer Science*) are equal (similar to but not exactly like Figure 3.1, for the ontology development goal) would imply that instances of one concept are also a member of the other concept. In this example, after merging the *CS* and *Computer Science* concepts, the ontology reasoner would infer that course *abc* is a member of both *CS* department in *$University_1$* and *Computer Science* department in *$University_2$*, and is offered by both departments. Also, course *xyz* is offered in both departments, which is obviously not correct.

Similarly, in the OWL language (W3C recommendation), using the *owl:equivalentClass* construct for the merging of two concepts (*CS* and *Computer Science*), instead of creating a skeleton, for the purpose of interoperability, is not acceptable for the same reason. In other words, stating that *$Class_1$* and *$Class_2$* are equivalent classes using *owl:equivalentClass*, implies that every instance of *$Class_1$* is also a member of *$Class_2$*. This is a very strong statement, and not generally applicable for facilitating interoperability between two systems.

Additionally, in Figure 3.2, when facilitating interoperability between parties, the parties are autonomous, and the data in the ontologies are often separate. While the parties need a mechanism for querying, we can not change the ontologies (business models) of either party, as we did in Figure 3.1 for *$Organization_1$* by merging. In Figure 3.2, ontologies *O_1* and *O_2*, and skeleton *S* are isolated and being administered independently in different *namespaces* (refer to OWL terminology). The ontology of each party does not change at all, and the skeleton is created separately, to connect the existing parties.

Up to now, we have distinguished the two goals (i.e. ontology development vs. interoperability), and also clarified that ontology merging is for ontology development and may not always be used for facilitating interoperability. Notice that merging was originally proposed for ontology development (refer to our explanation of [McG00, Noy00] in Chapter 5).

3.3.4. Implications of Context on Ontology Mapping

Traditionally, ontologies have been used for creating intelligent/expert systems. Those systems were often deployed by a limited number of experts, in constrained domains. The design of suitable ontologies in such systems required tools and expertise. As a result, the ontology development goal (Section 3.3.1) was at the focus of attention, e.g. [McG00, Noy00]. Nowadays, with the advent of the Semantic Web, the need for interoperability (Section 3.3.2) between systems/ontologies is becoming more visible.

Furthermore, we showed that interoperability is different from merging. In order to explore the requirements of the interoperability goal, we carefully probe the above use cases. This will illustrate how ontology mapping should be performed, to achieve interoperability. In this section, we study the ontology mapping goals, across seven dimensions, namely: representation, inconsistency, automation, isolation, class matching, tractability, and the relationship with the Semantic Web. The dimensions serve as a guideline for the ontology mapping task, and they influence the design of tools and algorithms for this task. Table 3.1 provides a brief overview of the ontology mapping goals, across different dimensions.

Table 3.1. A brief overview of the ontology mapping goals across different dimensions.

Dimensions	**Goal 1: Ontology Development**	**Goal 2: Facilitating Interoperability**
Representation	Merged ontology represents the correspondences.	Skeleton represents the correspondences.
Inconsistency	Inconsistencies arise from merging.	There is no inconsistency with the skeleton.
Automation	Merging should be semi-automated.	Skeleton creation is fully automated.
Isolation	With merging, isolation is not preserved.	With skeleton, ontologies reside in separate namespaces and are isolated.
Class Matching	Matching process should consider all entities in the ontology.	Class matching is critical, as it directly facilitates interoperability.
Tractability	Merging requires a human in the loop.	Skeleton creation can be tackled algorithmically.
Semantic Web	Not applicable.	Facilitating interoperability between data sources is a primary design goal of the Semantic Web vision.

A. Representation: Obviously, the goals of ontology mapping should propel the solution forward. Based on the definition of ontology mapping provided in Section 3.2, Step 1 (i.e. finding correspondences) is similar for achieving either goal of ontology mapping. In Step 2, for representing the found correspondences between two ontologies, we need a suitable representation. The question of how to represent the correspondences can be studied more concretely, in light of the distinction that we made about the goals of mapping. For developing and merging ontologies, the merged ontology is in fact the representation for the found correspondences (refer to O_{merged} in Figure 3.1). This is similar to the work of [McG00, Noy00].

As described in Section 3.3.3, merging of ontologies does not create a suitable representation for facilitating interoperability between two systems, in the general case. For interoperability, we present a novel W3C-compliant representation, named skeleton, to encode the correspondences between ontologies and facilitate interoperability between them, as shown in Figure 3.2. In Section 4.2, we provide an algorithm for creating the skeleton and also examine why the skeleton is a suitable representation. In Figure 3.2, there is no merging involved, and O_1, O_2 and S reside in different namespaces.

B. Inconsistency: When merging ontologies for ontology development (Section 3.3.1), various inconsistencies can arise and the task involves complex decision making, since there may be various ways to avoid the inconsistencies. For example, in Figure 3.1, consider the class *Videos* in ontology O_1 and class *Movies* in ontology

O_2, and instances of both these classes, which are movies, categorized using genres from the YahooMovies website. Consider that in ontology O_1, there is a cardinality restriction for instances of *Videos*, such that each instance of *Videos* has exactly one genre from the YahooMovies website. However, in ontology O_2, there is a cardinality restriction for instances of *Movies*, such that each instance of *Movies* has exactly two genres from the YahooMovies website. Now, if we merge the classes for *Videos* and *Movies* (as we did in Figure 3.1 for O_{merged}), it is not obvious how to handle this cardinality inconsistency. There are various options and the ontology designer has to make these decisions at design time, when developing the new ontology. Handling these complex issues is an integral part of the ontology development process, as outlined in the first goal.

For another example of inconsistency in Figure 3.1, assume that in addition to the *Toys* class which is a subclass of *Products* in O_2, the *Electronic Equipment* class in O_1 also has a *Toys* class as subclass. Now, the resulting merged ontology would have two *Toys* concepts, one of which is a subclass of *Sale Items* (shown in O_{merged} in Figure 3.1) and the other is a subclass of *Electronic Equipment* (not shown in O_{merged}). Even combining the two *Toys* concepts may not have the desired effect, since other conflicts could arise, similar to the cardinality problem, as mentioned previously. Additionally, the nature of the merging problem is such that the current ontology is not only being extended, but also needs to evolve, to accommodate the neighboring classes of the corresponding class in the existing ontology. For example, in Figure 3.1, if the class *Movies* did not have a parent class, a simple extension would have sufficed, but now that it has a *Products* class as its parent, we must accommodate the *Products* class as well, when merging *Movies* into O_1.

Considering our small example in Figure 3.1 and the various inconsistencies that could arise from merging, it is obvious that ontology merging is usually not a scalable process and should be performed in the context of developing a new ontology, to meet the new business demands of an organization. It is certainly not suitable for creating a global system to facilitate interoperability between parties. On the other hand, using a skeleton for interoperability does not create such inconsistencies, since the ontologies of the organizations are kept separately in different namespaces, as shown in Figure 3.2.

C. Automation: For both goals of ontology mapping, Step 1 (i.e. finding the correspondences), should have a human user in the loop (unless the results are approximate). Notice that the issue of representing the correspondences is dealt with separately, in Step 2, after the correspondences are given/determined.

As for Step 2 (creating a suitable representation for the correspondences), when merging ontologies for ontology development, there is a potential for inconsistencies to arise. Ontology design (similar to database design) involves subjective decisions. Hence, the merging process can only be “semi-automated” at best. Ontology merging algorithms should have a human user in the loop, as in the PROMPT Suite [Noy00]. Moreover, the process should be interactive, to allow the changes to the ontology to be verified at each step, by the human ontology designer. The designer should be familiar with ontologies and domain knowledge modeling.

On the other hand, when creating the skeleton representation for facilitating interoperability, as long as the correct set of corresponding concepts between the parties is given, as input, the skeleton can be created using a fully automated algorithm (refer to Section 4.2), since the process does not create inconsistencies.

D. Isolation: Let us revisit the two steps in Section 3.3.1 and the two steps in Section 3.3.2. For the ontology development goal, we must consider the changes in the business model of an organization (*Org_1* in Example 1) to guide the ontology development process. However, for the interoperability goal, there is no change in the business models of the organizations (universities in Example 2), at all. Creating a skeleton to represent the mappings for interoperability is more flexible than ontology merging. With the skeleton, the ontologies (parties) are isolated from any changes. Isolation is very desirable, since autonomous organizations, which are using the ontologies, are usually not willing to change their business practices for the sole purpose of communicating with other organizations. Therefore, interoperability must be facilitated by some means other than by changing the ontology (as in merging). The isolation of parties also eliminates inconsistencies.

E. Class Matching: In Step 1 of ontology mapping, when finding the correspondences between two ontologies, various entities in the ontology (e.g. classes, individuals, and properties) could be considered. For ontology merging, all entities are important for correspondences. In Section 3.5, we compare the information integration problem in databases to the interoperability goal in ontology mapping. The comparison shows that finding corresponding classes, is a critical part of ontology mapping for facilitating interoperability. However, this does not imply that matching of individuals is not important. In fact, matching of corresponding individuals provide auxiliary information for the ultimate task of class matching. Note that the skeleton representation (Figure 3.2) is actually geared towards capturing class correspondence, as well.

F. Tractability: The ontology merging process causes inconsistencies and can not be automated. However, the creation of a skeleton for interoperability is more tractable, since it does not create inconsistencies, and it can be automated. An algorithm for creating the skeleton is provided in Section 4.2. Generally, skeleton creation can be streamlined and tackled algorithmically, without a human in the loop. This is an important issue for the use case of creating scalable information integration systems, using ontologies.

G. The Relationship with the Semantic Web: By carefully examining the design goals of the Semantic Web, we illustrate that there is a close relationship between the ontology mapping problem for interoperability and the Semantic Web vision. The design goals of the Semantic Web include [Ber01, OWLu, OWLr]:

Using shared ontologies

Supporting ontology evolution

Ontology interoperability

Inconsistency detection across ontologies

Balance of scalability and expressivity in creating ontologies

Ease of use

Compatibility with other standards

Supporting internationalization.

We will use the item numbers to reference the eight goals, above. By focusing on the ontology mapping problem with an emphasis on interoperability (refer to Section 3.3.2), as opposed to the ontology merging emphasis (refer to Section 3.3.1), we also address the core design goals of the Semantic Web.

Note that in Figure 3.2, allowing various autonomous (isolated) organizations/parties to create and adopt their own ontologies as a community, is in agreement with **goal 2** (supporting ontology evolution). Facilitating interoperability between these isolated parties using the skeleton is in agreement with **goal 3** (ontology interoperability). In Section 3.6, when presenting the class similarity metrics, we demonstrate how **goal 1** (using shared ontologies) supports the class matching process for interoperability. Altogether, this neatly ties the ontology mapping problem for interoperability to the first three design goals of the Semantic Web. The idea of having distributed modular ontologies (adopted by different communities), and providing links between these ontologies to support interoperability is inline and tightly coupled with the spirit of the Semantic Web.

H. Discussion: In this section, we provided a framework to distinguish the goals of ontology mapping. We clarified how ontology mapping should be performed and represented, in different applications. Based on Section 3.3, it is clear that ontology development (as in ontology merging), and interoperability (i.e. information integration) are two distinct goals of ontology mapping. We illustrated that ontologies should not be merged for facilitating interoperability. We also investigated the implications of context (i.e. goals) on the ontology mapping problem.

The above use cases and analysis demonstrate that interoperability is an important goal of ontology mapping. By interoperability, we mean that users should be able to query distributed information sources, as if the data resides in a unified source. In Section 3.5, we focus on this goal; that is, independent of ontology merging. We treat the interoperability goal more thoroughly, by comparing it with the information integration problem in databases. The interoperability goal is quite similar to what the database community is trying to achieve in the context of information integration research and schema matching [Rah01, Len02, Hal05, Mel02, Li00].

3.4. Preliminaries for Formalization

In this section, we provide a formal definition for the terminology used in ontology mapping. Some understanding of these definitions was also necessary for the previous sections; however, the definitions were delayed, not to obfuscate the issue of distinguishing the ontology mapping goals and the implications of this distinction. Nevertheless, the formalization is provided in this section for a thorough and self-contained treatment. Additionally, this formalization will be required more specifically in Section 3.5.

These definitions are primarily based on Resource Description Framework (RDF), which is a framework and W3C recommendation for representing information on the Web. RDF is designed to represent information in a flexible way. The generality of RDF facilitates sharing of information between applications, by making the information accessible to more applications across the entire Internet. The interoperability goal, as identified in the previous section, aligns well with RDF design. Moreover, the Web Ontology Language (OWL) is based on RDF. Hence, this discussion applies to OWL as well.

Definition 1 (Resource): All things described by RDF are called resources.

Definition 2 (Triple): Each triple represents the statement of a relationship between the resources, denoted by the nodes that it links. Each triple has three parts: a subject, an object, and a predicate that denotes a relationship.

The direction of the link is significant; it always points toward the object. An RDF triple is conventionally written in the following order: subject, predicate, object.

Definition 3 (Property): The predicate is usually known as the property of the triple.

Definition 4 (RDF Graph): An RDF graph is a set of RDF triples. The set of nodes of an RDF graph is the set of subjects and objects of triples in the graph.

The graph can be illustrated by a node and directed-arc diagram, in which each triple is represented as a node-arc-node link (hence the term “graph”). The assertion of an RDF triple says that some relationship, indicated by the predicate, holds between the resources denoted by the subject and object of the triple. The assertion of an RDF graph amounts to asserting all the triples in it, so the meaning of an RDF graph is the conjunction of the statements corresponding to all the triples it contains.

Definition 5 (Class and Instance): Resources may be divided into groups called classes. The members of a class are known as instances or individuals of the class. Associated with each class is a set, called the extension of the class, which is the set of the instances of the class.

Classes are themselves resources. They are often identified by URI's and may be described using RDF properties. The rdf:type property may be used to state that a resource is an instance of a class. RDF distinguishes between a class and the set of its instances.

If a class C is a subclass of a class C', then all instances of C will also be instances of C'. The *rdfs:subClassOf* property may be used to state that one class is a subclass of another. The term super-class is used as the inverse of subclass. If a class C' is a super-class of a class C, then all instances of C are also instances of C'.

Definition 6 (Datatype): A datatype consists of a lexical space, a value space and a lexical-to-value mapping. The lexical space of a datatype is a set of Unicode strings. The lexical-to-value mapping of a datatype is a set of pairs whose first element belongs to the lexical space of the datatype, and the second element belongs to the value space of the datatype.

All datatypes are classes. The instances of a class that is a datatype are the members of the value space of the datatype. Each member of the lexical space is paired with (maps to) exactly one member of the value space. Each member of the value space may be paired with any number (including zero) of members of the lexical space (lexical representations for that value).

Definition 7 (Ontology): An ontology is an RDF graph, which is in turn a set of RDF triples.

3.5. Class Matching: A Critical Part of Facilitating Interoperability

Based on the definition of ontology mapping (in Section 3.2), in Step 1, the correspondences between ontologies need to be determined, and the question is what candidates to consider, when finding correspondences between ontologies. Now that we have clarified the goals of ontology mapping (in Section 3.3), we are ready to tackle the question of what candidates to consider. For the ontology development goal and merging of two ontologies, all the entities in an ontology (i.e. classes, individuals, and properties) are usually considered, so that the two ontologies can merged. In this section, we show that for the interoperability goal, finding the corresponding classes (i.e. class matching) is very critical.

Considering the similarity of the ontology mapping problem for facilitating interoperability and the information integration problem in databases, we analyze and compare them, in this section. First, the information integration problem in databases is examined. Then, the RDF model for ontologies and the relational model for databases are described. Finally, the models are compared. The comparison illustrates that class matching is critical, when mapping between ontologies for facilitating interoperability. This governs how ontology mapping should be performed for interoperability. Also, since OWL is based on RDF, our discussion applies to OWL as well.

Note that in this section, we are not mapping between ontologies and databases, i.e. there is no transformation involved between the two models. We are only comparing the models to illustrate what should happen, when facilitating interoperability between ontologies. The problem of facilitating interoperability between databases has been studied for three decades in the context of information integration in the database community [Ber81]. Hence, our comparison provides insight and sheds light on the relatively newer problem of interoperability between ontologies.

3.5.1. Information Integration in Databases

The problem of combining heterogeneous data sources under a single query interface is commonly known as “data integration” or “information integration” in the database community. A thorough and theoretical study of the problem is presented in [Len02]. We base our discussion on [Doa01, Len02], while other interpretations may exist. The main idea is to provide a uniform query interface over a mediated schema. The query is then transformed into specialized queries over the original databases. This process can also be called view based query answering, because we can consider each of the data sources to be a view over the mediated schema. Formally, such an approach is called Local As View (LAV), where “Local” refers to the local sources/databases. An alternate model of integration is one where the mediated schema is designed to be a view over the sources. This approach is called Global As View (GAV), where “Global” refers to the global (mediated) schema.

Figure 3.4 shows an example of the information integration problem in databases. Here, the goal is to generate a mapping between columns (*Town* and *City*) in different local schemas (*S* and *T*), by mapping them to some global schema. The local schemas usually reside in separate autonomous data sources (*DataSource1* and *DataSource2*). Figure 3.4 illustrates one example mapping between schema *S* and schema *T*. More details about the information integration process in databases can be found in [Doa01, Len02]. This is a simple and important example that will be used later in this section to demonstrate how ontology mapping should be performed to facilitate interoperability, and how ontology mapping for interoperability relates to schema mapping (i.e. information integration).

3.5.2. Interoperability

Following the description of information integration in databases, above, it is important to point out that, the term “integration” might be vague to some extent, since it might unintentionally be interpreted as some type of “merging” of schemas. This is not what actually occurs in databases, as the schemas in each local data source are handled autonomously and need to be kept separately. The local data sources are often administered by different organizations, and are not merged (integrated). In fact, organizations are not willing to change their business models and everyday operations.

The ultimate goal of information integration is to provide interoperability between various systems, which is the exact same goal that we identified in Section 3.3.2. The term "interoperability" is clearer for describing the motivations and objectives of the process. By analogy, there is no merging of ontologies involved in ontology mapping, when we are trying to achieve interoperability between organizations, which use different ontologies (refer to Section 3.3.2).

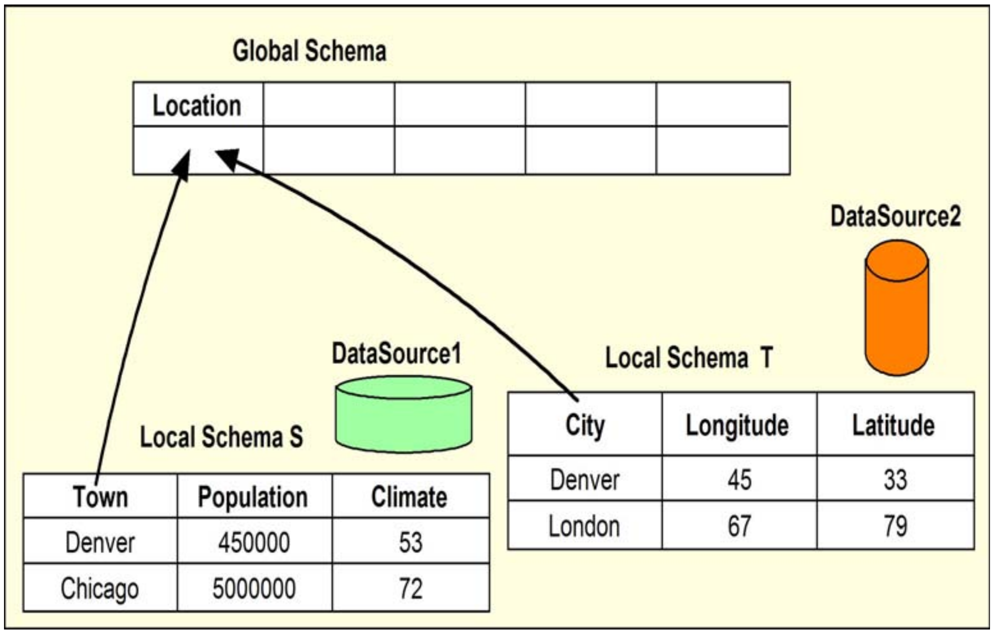

Figure 3.4. Example of the information integration problem in databases. The goal is to generate a mapping between columns (*Town* and *City*) in different local schemas (*S* and *T*), by mapping them to some global schema. The local schemas usually reside in separate autonomous data sources (*DataSource1* and *DataSource2*).

3.5.3. Expression of Simple Facts in the RDF Model

Simple facts in the RDF model indicate a relationship between two resources. Such a fact may be represented as an RDF triple, in which the predicate (i.e. property) names the relationship, and the subject and object denote the two resources. Figure 3.5(a) shows the predicate *hasAuthor*, which is the relationship between the subject and the object. The subject is an instance of class *Book*, while the object is an instance of class *Author*. The classes are depicted as ovals. For example, *The Art of Computer Programming* (which is a book) hasAuthor *Donald Knuth* (who is an author).

In contrast with the relational database model (Section 3.5.4), the use of extensible URI-based vocabularies in RDF facilitates the expression of facts about arbitrary subjects; i.e. assertions of named properties about specific named resources. A URI can be constructed for any resource that can be named, so RDF facts can be about any such resources. The use of Uniform Resource Identifiers (URIs) in the RDF model provides a very powerful mechanism for facilitating interoperability on the Semantic Web. Consider that the success and scalability of the current WWW infrastructure is a vivid illustration of the tremendous potential of the idea of using links or URIs (which seems like a simple idea at first glance).

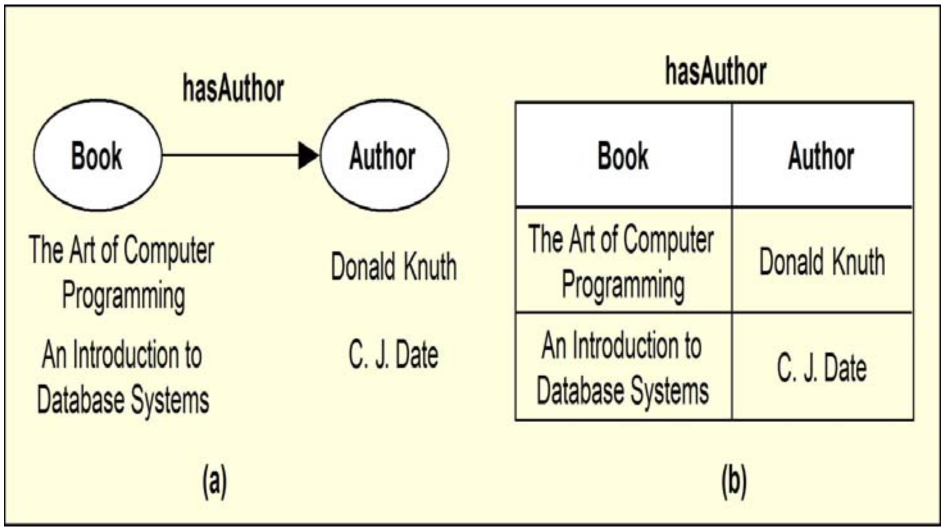
hasAuthor
Book
Author
The Art of Computer Programming
An Introduction to Database Systems
Donald Knuth
C. J. Date
(a)
hasAuthor
Book
Author
The Art of Computer Programming
Donald Knuth
An Introduction to Database Systems
C. J. Date
(b)

Figure 3.5. Correspondence between the RDF and relational models. (a) The predicate *hasAuthor*, which is the relationship between the instances of class *Book* and the instances of class *Author*, in the RDF model. (b) The table *hasAuthor*, which has two columns, namely *Book* and *Author*, in the relational model.

3.5.4. Expression of Simple Facts in the Relational Model

A familiar representation of a fact in the relational model, in databases, is a row in a table. The terms row and table are also known as tuple and relation, respectively. A table has a number of columns (also known as attributes). Figure 3.5(b) shows the *hasAuthor* table. The table has two columns, namely *Book* and *Author*. A row, for example, indicates that, *The Art of Computer Programming* (which is a book) hasAuthor *Donald Knuth* (who is an author).

3.5.5. The Analogy between the RDF and Relational Models

By comparing the explanations of the RDF and relational models above, and by observing Figure 3.4, we can infer that the classes *Book* and *Author* (in RDF) correspond to the columns *Book* and *Author* (in relational). The correspondence between classes and columns is an important one, and it will be used in Section 3.5.6. The correspondence is also depicted in Figure 3.6. The other substantial correspondence is between instances of a class in RDF, with column values in relational.

In the above discussion, the description of the relational model was constrained, such that a table only contained two columns. Now, we consider the general case, where a table contains more than two columns. In Figure 3.7(b), the table in the relational model has three columns, namely *Book*, *Author* and *Publisher*. Then, the RDF model would also have three corresponding classes, as shown in Figure 3.7(a). The correspondence between classes and columns still holds. It is essential to realize that the name of the table in the relational model is arbitrary. We used TableName in Figure 3.7(b). Additionally, two predicates, namely *hasAuthor* and *hasPublisher*, are now used in the RDF model (Figure 3.7(a)). The name of the two predicates in RDF is arbitrary and could be anything.

Notice that in Figure 3.5, we do not infer a correspondence between the name of the table (*hasAuthor* in relational) and the name of the predicate (*hasAuthor* in RDF), as this correspondence has no real substance. The comparison in Figure 3.7 actually eliminates the role of the table name in the relational model. Therefore, the substantial correspondence is between classes and instances in RDF, with columns and column values in relational, respectively. There is no correspondence for predicate (i.e. property) names in the relational model. This is due to the fact that in the RDF model, relations are encoded explicitly, in contrast with the relational model. The explicit encoding of relations in RDF enables users to name the relations, in the data, which gives rise to typed relations.

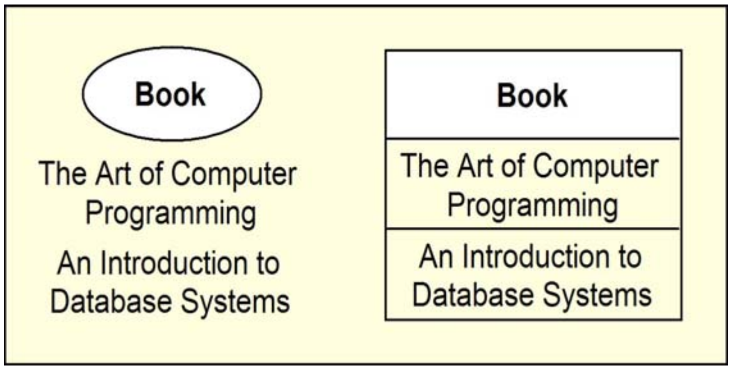
Book
The Art of Computer Programming
An Introduction to Database Systems
Book
The Art of Computer Programming
An Introduction to Database Systems

Figure 3.6. The correspondence between classes (in the RDF model) and columns (in the relational model). There is also a correspondence between instances of a class (in RDF) and column values (in relational).

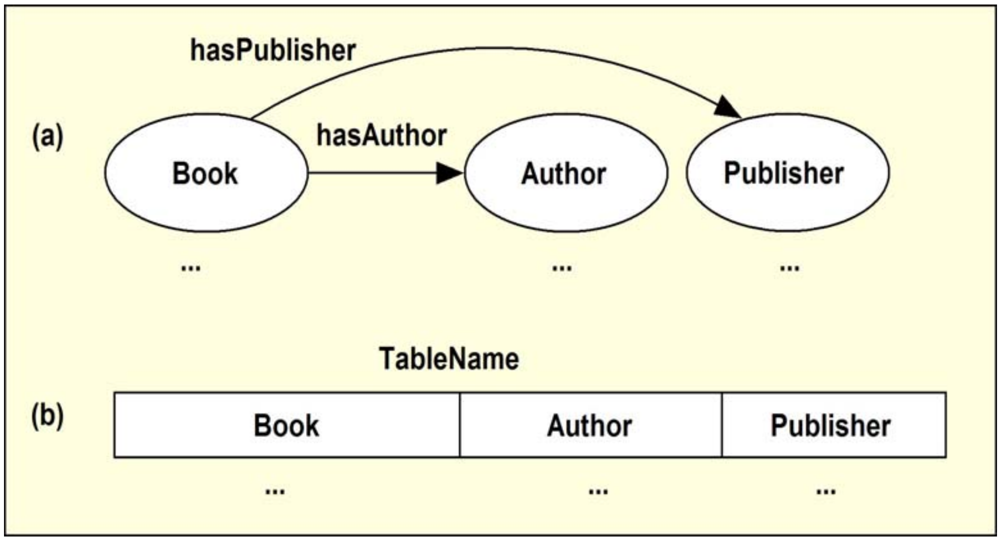

Figure 3.7. A more general correspondence between the RDF and relational models. (a) The three classes in RDF are *Book*, *Author*, and *Publisher*. The name of the two predicates (*hasAuthor*, *hasPublisher*) in RDF is arbitrary and could be anything. (b) A table in the relational model, which has three columns, namely *Book*, *Author*, and *Publisher*. The name of the table (*TableName*) is arbitrary.

3.5.6. Class Matching: Why

Section 3.5.1 showed that in databases, the final output of the information integration process is a mapping between *columns* in different local schemas (refer to Figure 3.4). In Section 3.5.5, we illustrated that *columns* in the relational model correspond to *classes* in the RDF model (refer to Figure 3.5 and 4.6). Therefore, it is clear that to facilitate interoperability between ontologies, the *classes* in the ontologies need to be mapped to each other.

In RDF, the data is the instances of classes. The ultimate objective of interoperability is to *query* and correctly retrieve these data instances, across various ontologies. The data resides in the classes in the ontologies. Notice that as long as the correct mapping between the classes in the ontologies exists, users can query and correctly retrieve the data instances, across various ontologies. For example, in Figure 3.2, users would like to retrieve course instances from both *CS* class in *University$_1$* and *Computer Science* class in *University$_2$*. By analogy, in the relational model, the data is the values stored in the columns. A correct mapping between the columns in the schemas enables users to correctly retrieve the data (column values) across various schemas.

In Figure 3.2, in order to more fully comprehend the interoperability use case, as described in 4.3.2, one should consider a user interaction paradigm similar to the MIT's Tabulator project for the Semantic Web [Ber06]. Looking at Figure 3.2, users can retrieve and view the course instances in the *CS* class in *University$_1$*, using a Linked Data browser like the Tabulator. They can also browse the properties of these course instances in *University$_1$*. In addition, if users would like to retrieve course instances in the *Computer Science* class in *University$_2$*, they can do that via the skeleton. That is, when viewing the course instances in the *CS* class in *University$_1$*, using query expansion, we can move to the corresponding concept in the skeleton (which is *CS*), and then also retrieve the relevant course instances in the *Computer Science* class in *University$_2$*. When viewing the course instances in the *Computer Science* class in *University$_2$*, again users can browse the properties of these course instances.

Let us return to the question (raised in the ontology mapping definition in Section 3.2) that what candidates should be considered, when finding correspondences between ontologies. Our detailed comparison in this section clarifies that a critical part of ontology mapping for facilitating interoperability is the matching of classes. The class matching objective directly facilitates interoperability.

This is one of the critical implications of focusing on the context of interoperability in ontology mapping, as mentioned in Section 3.3.4.E. The class matching objective does not imply that other entities are not used for class matching. Matching of other entities is usually helpful for the matching of classes, as described in Section 3.6. Basically, the question of how to find the matching classes is a separate issue, addressed in the next section.

3.6. Categorization of Class Similarity Metrics

In the previous section, we identified that class matching is a critical part of ontology mapping for facilitating interoperability. In this section, we provide an algorithm for class matching (i.e. finding corresponding classes). The algorithm utilizes various class similarity metrics. Each of the class similarity metrics is formally defined and the time complexity of computing the metric is analyzed, which is important for scalability in real-world settings. Then, the relationship between the matching of classes and the matching of instances is clarified.

The class similarity metrics, formalized in this section, are inspired by various works in the literature. However, those works are often different from ours. Some works in the literature are in the context of ontology merging efforts, e.g. [Mc00, Noy00, Stu01]. Some other works, which are not in the context of merging, are addressing ontology matching only (i.e. Step 1, in Section 3.2). These works do not necessarily focus on classes only, and they consider other entities as well, e.g. [Ehr04]. Also, for the class matching part, the work often attempts to find subsumption relationships between classes in the ontologies (e.g. [Bou04]), which is different from measuring class similarity. Our algorithm for class matching directly measures class similarity. Note that measuring class similarity is necessary, in our algorithm, since the output of the class matching algorithm (presented in this section) is later used, as the input of the skeleton creation algorithm (presented in Section 4.2).

For the class similarity metrics, formalized in this section, we exploit the information provided inside the ontology. Obviously, outside resources can be utilized in various ways to help the class matching process. For example, resources like dictionaries and thesauri can be used in this process. Also, a human expert (user), who has knowledge about the domain, can make decisions about the matches. Moreover, the expert's knowledge and user interaction can potentially be captured, in the form of labeled data and training examples, to facilitate machine learning.

Definition 8 (Mapping for interoperability): Let C_1 be the set of classes of ontology O_1 and C_2 be the set of classes of ontology O_2. Map m is a total function $m : C_1 \otimes C_2 \rightarrow [0,1]$, where $C_1 \otimes C_2$ is defined as the set of all distinct pairs of elements of sets C_1 and C_2, that is: $C_1 \otimes C_2 = \{(a,b) \mid a \in C_1, b \in C_2\}$.

Definition 9 (Class Matching, Threshold, Similarity Value, Similarity Metric): Class matching is the process of determining corresponding classes between ontologies O_1 and O_2, which is specified using a threshold t. Map m assigns a similarity value to each pair of classes. If the similarity value, defined by map m, is greater than threshold t, then the classes match. We compute the similarity value by summing the result of the following four similarity metrics: lexical, extensional, extensional closure, and global path, which are defined, later. In order to compute the similarity value more effectively, a weighted sum of these metrics could be used, or the result of each metric could be normalized.

In real-world applications, the issue of setting the *threshold* for identifying corresponding classes is an important one, and it may require human judgment. As mentioned in Section 3.3.4.C, in regard to automation, the merging of ontologies for the ontology development goal can not be automated and needs a human user in the loop. On the other hand, ontology mapping for the interoperability goal consists of two steps (Section 3.3.2): finding the class correspondences (Step 1), and representing the class correspondences using a skeleton (Step 2). For Step 1, human judgment may be required for setting the threshold. The process of finding correspondences inherently involves uncertainty; no algorithm can achieve one hundred percent precision. However, if we allow approximate answers (similar to web search engine results which are not always relevant), the threshold can be set automatically (using machine learning techniques), or semi-automatically (with a human user involvement). For Step 2, Section 4.2 provides a fully automated algorithm for creating the skeleton.

Notice that skeleton creation (Step 2) is a separate issue from class matching (Step 1), and happens after class matching. While class matching can only be semi-automated, skeleton creation, which happens after a set of correspondences are given, can be fully automated. Section 4.1 provides the class matching algorithm, and

Section 4.2 provides the skeleton creation algorithm. Section 3.3.4.C and Section 3.10 further clarify the automation issue.

In principle, ontologies can cover any domain of knowledge, and the *nature of data instances* is extremely diverse in different applications. Hence, it is difficult to provide general guidelines on how to set the threshold for all ontologies/applications. Essentially, the threshold needs to be determined experimentally for each application and dataset.

Definition 10 (Lexical Similarity Metric): Let $s \in C_1$ and $t \in C_2$ be two classes in the ontologies. The lexical similarity metric is a function that assigns a real-valued number in the range of [0, 1] to the pair {s, t}, based on the closeness of the strings representing the names of s and t.

Definition 11 (Extensional Similarity Metric): Let $s \in C_1$ and $t \in C_2$ be two classes in the ontologies. The set of individuals, which are direct members of s and t, are represented as e(s) and e(t), respectively. The extensional similarity metric for s and t is computed as $|e(s) \cap e(t)| / |e(s) \cup e(t)|$. This is similar to computing the Jaccard similarity coefficient of two sets.

Definition 12 (Extensional Closure Similarity Metric): Let $s \in C_1$ and $t \in C_2$ be two classes in the ontologies. If class x is a subclass of class y, it is denoted as $x \sqsubseteq y$. The extensional closure of s, denoted as $e_c(s)$, is computed as $e_c(s) = \{\bigcup_{i \in C_1} e(i) \mid i \sqsubseteq s\}$. The extensional closure similarity metric for s and t is equal to $|e_c(s) \cap e_c(t)| / |e_c(s) \cup e_c(t)|$.

This intuitively means that when comparing two classes, the extensional closure considers, not only the individuals that are a member of class *s*, but also all the individuals that are a member of the subclasses of class *s*.

Definition 13 (Global Path Similarity Metric): Let $s \in C_1$ and $t \in C_2$ be two classes in the ontologies. Path of s, denoted as p(s), is the path that starts from the root of an ontology and ends at s. The global path similarity metric for s and t is equal to the score assigned to the similarity of p(s) and p(t). The score is based on the lexical similarity of the classes that appear in the two paths.

Complexity Analysis (Lexical Similarity Complexity): The upper bound complexity of computing the lexical similarity metric for ontologies O_1 and O_2 is $O(|C_1|.|C_2|)$. There is no need for ontology reasoning in the computation of this metric.

Complexity Analysis (Extensional Similarity Complexity): The complexity of computing the extensional similarity metric for classes s and t is $O(|e(s)|.|e(t)|)$. The set of individuals e is computed using the ontology reasoner.

Complexity Analysis (Extensional Closure Similarity Complexity): The complexity of computing the extensional closure similarity metric for classes s and t is $O(|e_c(s)|.|e_c(t)|)$. The subclasses of a class, and their respective individuals, are computed using the ontology reasoner.

Complexity Analysis (Global Path Similarity Complexity): If the length of the path p(s) is denoted as $||p(s)||$ (which is equal to the depth of the class hierarchy in the worst case), then the complexity of computing the global path similarity metric for classes s and t is O(||p(s)||.||p(t)||). The class hierarchy is created from the subclass relationships, using the ontology reasoner.

3.7. Reasoning for Class Matching

The role of a Description Logic reasoner is important in class matching for ontology mapping, and this is one of the points that distinguish ontology mapping from schema mapping in databases, as there are limited reasoning capabilities in databases. Standard ontology reasoners provide the following services:

- *Classification*: Computes the subclass relations between every named class to create the complete class hierarchy. The class hierarchy can be used to answer queries such as getting all the subclasses of a class or only the direct subclasses.
- *Realization*: Finds the most specific classes that an individual belongs to; in other words, computes the direct types for each of the individuals. Realization can only be performed after classification, since direct types are defined with respect to a class hierarchy. Using the classification hierarchy, it is also possible to get all the types for an individual.
- *Consistency checking*: Ensures that an ontology does not contain any contradictory facts.
- *Concept satisfiability*: Checks if it is possible for a class to have any instances. If a class is unsatisfiable, then defining an instance of that class will cause the whole ontology to be inconsistent.

The computation of the extensional, extensional closure, and global path similarity metrics require an ontology reasoner, while the lexical similarity metric does not require any reasoning.

3.8. Instance Matching

Based on Definitions 11 and 12, instances within two classes need to be matched, i.e. duplicate instances should be identified. This task is necessary for both the ontology development goal and the interoperability goal. When merging ontologies for ontology development, duplicate instances in corresponding classes need to be detected and eliminated, so that the classes in the merged ontology would only contain unique instances. When performing class matching for facilitating interoperability, as discussed in Section 3.3.4.D (the isolation dimension), the instances of classes are not merged. Nevertheless, duplicate instances may need to be detected, in order to compute the intersection of instances of two classes correctly, when computing the extensional and extensional closure similarity metrics.

There are various approaches that can be used for instance matching. One simple approach is based on the assumption that if two instances in different ontologies are the same, then they also use the same URI, which would help in identifying the instances uniquely. This assumption is usually not applicable in practical settings, since different organizations use different naming standards and URIs. Another approach, which we also use in our experiments, is to identify duplicate instances using approximate string matching techniques for the name of instances, similar to the lexical similarity metric used for the name of classes (Definition 10).

In a more complicated approach, the domain knowledge about instances and the facts stated in a knowledge base may also be used. As mentioned in Section 3.3.4.G, some of the design goals of the Semantic Web (SW) are quite beneficial for instance matching on the SW. Remember that design goal 1 was using shared ontologies. An example of a shared ontology is FOAF. The Friend of a Friend (FOAF) project is about creating a Web of machine-readable pages describing people, the links between them and the things they create and do [FOAF]. If *T. B. Lee* and *Tim Berners-Lee* are two instances and both have the same *foaf:mbox* property value (i.e. email address), which is an *owl:InverseFunctionalProperty*, the ontology reasoner can infer that the instances are duplicates. In other words, ontology inference helps in identifying duplicate instances that may not be detected using approximate string matching.

Notice that in this example, the use of shared ontologies (e.g. foaf) helps in instance matching, which is a part of the ontology mapping process. This approach, for instance matching on the Web, has important applications for facilitating interoperability in the Semantic Web vision.

Furthermore, there is an interesting analogy between instance matching in ontologies and the problem of approximate duplicate elimination in data cleaning and databases. There is a considerable amount of research on duplicate elimination in the database literature. A recent survey on this problem is [Elm07]. Many of the approaches developed for data cleaning in databases are potentially applicable to finding duplicate instances, when merging ontologies, and when computing extensional similarity metrics for class matching.

3.9. Summary

Facilitating interoperability between different ontologies is one of the classic and long-standing issues in AI. Over the last decade, however, the problem of ontology mapping has attracted significant attention. This is partly due to the deployment of ontologies (in the form of Linked Data) on the Web of Data. We identified two sharply distinct goals for ontology mapping, based on real-world use cases. These goals are: (i) ontology development, and (ii) facilitating interoperability. We systematically analyzed these goals, side-by-side, and contrasted them. Our analysis demonstrated the implications of the goals on ontology mapping and mapping representation.

We showed the consequences of focusing on interoperability with illustrative examples and provided an in-depth comparison to the information integration problem in databases. The consequences include: (i) an emphasis on class matching, as a critical part of facilitating interoperability; and (ii) an emphasis on the representation of correspondences. For class matching, various class similarity metrics were formalized and their time complexities were analyzed.

Chapter 4: ALGORITHMS AND EXPERIMENTS

In this chapter, we provide the algorithms that were discussed in Chapter 3. We present a methodology for performing the experiments. Then we experimentally evaluate the algorithms using ontologies from various domains.

4.1. Class Matching Algorithm

For Step 1 in Section 3.3.2, we now present our class matching algorithm, below. We call this algorithm MATCH. The class matching algorithm exploits the class similarity metrics introduced in Chapter 3. Line 2 relates to Definition 10. Lines 3-5 compute the extensional similarity using the ontology reasoner, as in Definition 11. Lines 6-8 are based on the extensional closure similarity, as in Definition 12. Lines 9-11 compute the global path similarity, as in Definition 13. Line 12 computes the similarity value for classes and compares it to the threshold, as in Definition 9.

```
Class-Matching Algorithm

Input:
  $O_1$, $O_2$: Original ontologies
  $C_1$: Set of classes in $O_1$
  $C_2$: Set of classes in $O_2$

Output:
  M: Set of matching class
       pairs ($c_1$, $c_2$), s.t. $c_1 \in C_1$, $c_2 \in C_2$

1. for all $c_1 \in C_1$, $c_2 \in C_2$
2.   lexSim← lexicalSim($c_1$.name, $c_2$.name)
3.   $c_1$.ex← reasoner.Extensions($c_1$)
4.   $c_2$.ex← reasoner.Extensions($c_2$)
5.   extSim← extensionalSim($c_1$.ex, $c_2$.ex)
6.   $c_1$.all← reasoner.AllExtensions($c_1$)
7.   $c_2$.all← reasoner.AllExtensions($c_2$)
8.   extCSim← extensionalClosureSim($c_1$.all, $c_2$.all)
9.   $c_1$.path← reasoner.GlobalPath($c_1$)
10.  $c_2$.path← reasoner.GlobalPath($c_2$)
11.  gpSim← globalPathSim($c_1$.path, $c_2$.path)
12.  if ( lexSim+extSim+extCSim+gpSim > threshold)
       then M ← M ∪ ($c_1$, $c_2$)
13.end for
14.return M
```

4.2. Skeleton Assessment

In the Barbara example, in Section 2.1, in tasks (iv) and (v), the information assimilator needs to access various information sources and package the results, in order to collect the necessary information to answer the question. In this section, we further investigate these two tasks. In order to return the results of a query from distributed sources of information, we need to somehow consolidate the results of a query. A suitable representation should help us in achieving this goal.

The issue of representation is an important implication of focusing on the context of interoperability in ontology mapping, as mentioned in Section 3.3.4.A. Based on the definition of ontology mapping (in Section 3.2), in Step 2, the found correspondences between the ontologies need to be represented. Now, the question is how to represent them in a suitable format. For the ontology development goal, when integrating ontologies, the outcome of the process is one merged ontology. This merged ontology actually represents the correspondences between the ontologies.

For the interoperability goal, we should not merge the ontologies. Instead, we provide a novel W3C-compliant representation, named *skeleton*, which enables users to search and discover knowledge from different ontologies, more generally and effectively than existing alternatives. More specifically, the skeleton provides a uniform representation of mappings across ontologies and is stored independent of those ontologies. A user's query, in a Linked Data browser, may then be largely guided and managed via the uniform skeleton representation, freeing the user from the burdensome and time-consuming task of mapping during the querying process. In this section, we provide an algorithm for creating the skeleton (refer to Step 2 in Section 3.3.2).

Sections 3.3.2 and 3.3.3 described how the skeleton facilitates interoperability between organizations. We also described the query expansion mechanism in ontologies for query answering. As shown in Figure 3.2, the skeleton represents the class correspondences between the ontologies of organizations. These correspondences are essential for searching and query answering. The skeleton is a suitable representation, as it allows query answering over various ontologies (organizations). The skeleton increases the recall of queries by enabling users to retrieve results from distributed sources.

Note that a suitable representation for the matching classes should comply with the W3C recommendations, like RDF and OWL, so that the representation can be seamlessly deployed in different applications using existing tools, with little implementation effort. Currently, there exists no such standard approach for representing the matching classes between ontologies, as is necessary for facilitating interoperability (refer to Section 3.3.2).

Our design for the skeleton representation is compliant with the W3C recommendations. In other words, when using the skeleton, query answering and query expansion can be performed using standard tools, without any ontology reasoner adjustments and extra implementation effort. That is to say, all these issues are handled by the ontology reasoner in a standard fashion.

Creating a skeleton isolates the original ontologies, and therefore each autonomous organization will use its own business model for everyday operations (Section 3.3.4.D). This isolation, in turn, eliminates the possibility of the inconsistencies that arise, when ontologies are merged (Section 3.3.4.B). Moreover, once the class matching is done and the correspondences have been determined, the process of creating the skeleton is fully automated and performed using our algorithm, without any human user involvement (Section 3.3.4.C).

It should be mentioned that the data exchange (or data transformation) work is different from our skeleton mechanism. That is because the skeleton facilitates query answering over various ontologies, while data exchange does not do that. Hence, transformations (functions) that manipulate the data of a source, to make the data compatible with a target, are not the focus of our approach, when facilitating interoperability between ontologies. However, in databases, such transformations and functions (usually implemented using SQL) are of interest, for example in the Extract, Transform, Load (ETL) process of data warehouses.

To describe the *Skeleton-Creation* algorithm, below, we use the motivating example, depicted in Figure 3.2. The *Class-Matching* algorithm, introduced in Section 4.1, is a prerequisite for the *Skeleton-Creation* algorithm. The output of the *Class-Matching* algorithm is the set of matching class pairs (*M*) of the two ontologies. This set (*M*) is the input of the *Skeleton-Creation* algorithm. In lines 1-5, for each pair in set *M*, a class node is created in the skeleton. The name of the class in ontology O_1 is assigned to the class in the skeleton. The class in the skeleton is connected to the classes in the pair. In Figure 3.2, the classes in the skeleton are *University*, *Science*, *Maths*, *CS*, *Physics*, and *Chemistry*, which are connected to their corresponding classes in O_1 and O_2. Figure 3.2 shows such connections for the *University* concept, only, i.e. the *University* concept in the skeleton is connected to concepts $University_1$ and $University_2$ in ontologies O_1 and O_2, with blue dotted arrows. In line 6, the ontology reasoner infers the class hierarchy of ontology O_1. In line 7, this hierarchy is used for connecting the class nodes in the skeleton, to each other.

```
Skeleton-Creation Algorithm

Input:
 O1, O2: Original ontologies
 C1: Set of classes in O1
 C2: Set of classes in O2
 M: Set of matching class
      pairs (c1, c2), s.t. c1∈C1, c2∈C2
      (same as the Output of the
      Class-Matching Algorithm)

Output:
 S: Skeleton

1.  for each pair (c1, c2) in M
2.    Create a class node s∈S
3.    s.name ← c1.name
4.    Connect s to c1 and c2, using
        subclass relation
5.  end for
6.  H1 ← reasoner.ClassHierarchy(O1)
7.  Create the same class hierarchy
      as H1, between all classes s∈S
8.  Return S
```

Notice that in lines 6 and 7, the hierarchy of ontology *O₁* is used for creating the hierarchy of the skeleton. However, the hierarchy of ontology *O₂* could also be used for the skeleton. Then, the shape of the hierarchy of the skeleton may change, however, interoperability would still be facilitated and the query answering process would not change. To elaborate on this issue, we provide an additional example in this section.

Consider that ontology O_1 contains two classes, namely A_1 and B_1, where $B_1 \sqsubseteq A_1$. Also, ontology O_2 contains two classes, namely A_2 and B_2, where $A_2 \sqsubseteq B_2$. This is shown in Figure 4.1(a) and 4.1(b). Assume that A_1 and A_2 are the corresponding (matched) classes in the two ontologies. Also, B_1 and B_2 are the matched classes in the two ontologies.

Now, using the *Skeleton-Creation* algorithm, in lines 6 and 7, if ontology O_1 is used for the shape of the skeleton, the result is shown in Figure 4.1(a). If ontology O_2 is used for the shape of the skeleton, the result is shown in Figure 4.1(b). While the shape of the skeleton changes (depending on the party that is used for the hierarchy of the skeleton), interoperability is still achieved in both cases. In other words, in both figures, using the skeleton, we can query for instances of class A_1 in ontology O_1 and using query expansion, we move to the corresponding class in the skeleton (which is A_s), and then also retrieve the relevant instances from class A_2 in ontology O_2. Therefore, the query would return the results, as if all data resides in a unified source. Additionally, since ontologies O_1 and O_2, and skeleton S reside in different namespaces, there is no inconsistency.

This example was an extreme case, where the order of the matching classes in the parties were reversed, i.e. $B_1 \sqsubseteq A_1$ and $A_2 \sqsubseteq B_2$. However, in practice this order is usually not reversed, and the shape of the skeleton will not change dramatically, regardless of the party that is used for the shape of the skeleton in lines 6 and 7.

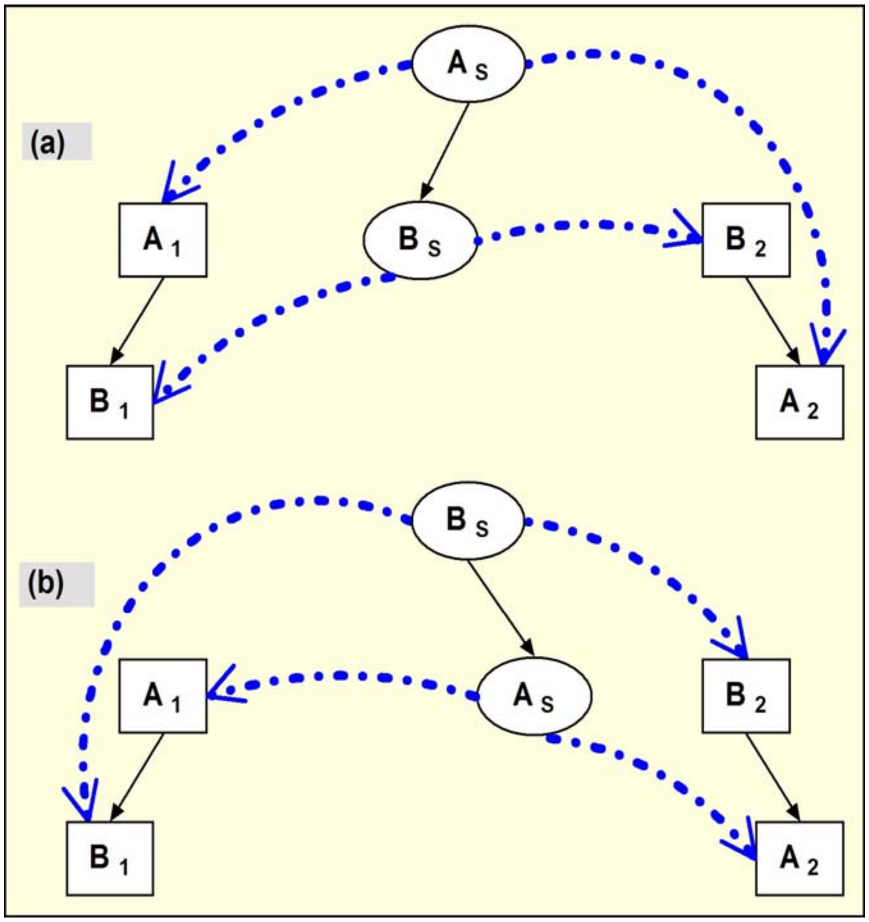

Figure 4.1. The shape of the hierarchy of skeleton *S* is shown in (a) and (b), respectively, when using ontologies O_1 and O_2, for the shape of the skeleton.

In Section 3.3.2.A, we discussed how the skeleton can be created for more than two ontologies. For the case shown in Figure 3.3(a), the input for creating the skeleton is two ontologies, so the algorithm presented in this section can be directly applied. For the case shown in Figure 3.3(b), the input for creating the skeleton is three or more ontologies, so the algorithm in this section can be applied with a minor modification. The modification is that in line 4, we connect the class in the skeleton to the matching classes in all the ontologies (not just two ontologies).

In Section 3.3.2.B, we discussed how users can also get results from all the subclasses of a given class. This can be achieved easily by modifying the subclass retrieval function in an ontology reasoner.

4.3. Experimental Methodology

In this section, we discuss the issues that need to be considered for a good way of testing a class matching algorithm. We address various questions, such as how the ontologies are selected for our experiments, what are the typical features of ontologies, what kinds of ontologies exist, what are ontologies used for, how the gold standard is created for the matching process, and what evaluation metrics should be used.

4.3.A. General Methodology Background

4.3.A.1. Hypothesis:

Variables are things that we measure, control, or manipulate in experiments. Two or more variables are related if, in a sample of observations, the values of those variables are distributed in a consistent manner. After the relationship between two variables is calculated, we would like to know how significant the relationship is. The statistical significance of a result is the probability that the observed relationship (between variables) or a difference (between means) in a sample occurred by pure chance, and that in the population from which the sample was drawn, no such relationship or differences exist.

This significance depends on sample size. In a large sample, small relations between variables are significant, but in a small sample even large relations are not reliable. For computing significance, we need a function that represents the relationship between magnitude and significance of relations between variables, depending on sample size. The function would give us the significance (p) level and tell us the probability of error involved in rejecting the idea that the relation in question does not exist in the population. This alternative hypothesis (that there is no relation in the population) is usually called the null hypothesis. Most of these functions are related to a general type of function, which is called normal.

Comparing to gold standard: For the purpose of comparison, the results of a system can be compared to some gold standard. In other words, we can compute how many of the results of the system are correct (i.e. match the gold standard), and how many of the results are not correct (i.e. do not match the gold standard). The number of correct and incorrect cases in the result will determine the precision and recall measures. Please refer to Section 4.3.A.3 for a description of the precision and recall measures used in our experiments.

Comparing to other tools: The results of a system can also be compared to other existing systems vis-a-vis some gold standard. In this case, the output of each tool is compared to the gold standard, and the F1 measure is computed for each tool. If the experiment is repeated for various data sets, then a mean F1 can be computed.

Please refer to Section 4.3.A.3 for a description of the F1 measure used in our experiments.

After the difference between the mean F1 is observed, we can compute how significant the difference is. The statistical significance of a result is the probability that the observed difference (between mean F1 values) in a sample occurred by pure chance, and that in the population from which the sample was drawn, no such differences exist. Please refer to Section 4.3.A.4 for further description of statistical significance.

How to get the gold standard: When evaluating an algorithm using a gold standard, the gold standard needs to be determined. The gold standard is the result that we would like to get in an ideal situation. The gold standard can be created by independent observers, and if there is disagreement between observers the disagreement needs to be resolved. The gold standard can also be created by the developers of the system, but this should ideally be done before any results from experiments are seen by the developers, in order to prevent any bias.

4.3.A.2. Ideal Criteria for Trial Data:

Ontology pairs: Ontologies have a number of features, for example: the name of the ontology, the organization that created the ontology, the domain that is covered by the ontology, main topic, number of classes, number of instances, level of expressivity, and the size of the ontology. In order to measure how well the class matching algorithm performs, in the general case, we should use a set of ontologies (trial data) that includes the different kinds of ontologies that exist.

As will be explained below, a typical number of ontologies used for a trial set ideally would be around thirty ontologies. For example, ontologies that cover sciences, like biology, or topics like cells and diseases. Ontologies may model the publication process and bibliographic entries. They may represent concepts related to conferences or their registration process. They could be about various foods, like pizza, or beverages, like wine and beer. Ontologies may be created by large groups of developers or they can be created by one person for some specific task.

Typical size of trial set: Test statistics are not always normally distributed, but most of them are either based on the normal distribution or on distributions that are related to and can be derived from normal, such as t, F, or Chi-square tests. These tests usually require that the variables analyzed are themselves normally distributed in the population. A problem may occur when we try to use a normal distribution-based test to analyze data from variables that are themselves not normally distributed. In such cases, we have two general choices.

First, we can use some distribution-free test, but this is often inconvenient because such tests are typically less powerful in terms of types of conclusions that they can provide. Alternatively, in many cases we can still use the normal distribution-based test if we only make sure that the size of our samples is large enough.

The latter option is based on an important principle that is largely responsible for the popularity of tests that are based on the normal function. Namely, as the sample size increases, the shape of the sampling distribution (i.e., distribution of a statistic from the sample) approaches normal shape, even if the distribution of the variable in question is not normal. As the sample size (of samples used to create the sampling distribution of the mean) increases, the shape of the sampling distribution becomes normal. This principle is called the central limit theorem. For n=30, the shape of the distribution is almost perfectly normal [statsoft]. Hence, a data set of around 30 cases would be a good sample size to use, for performing statistical tests.

The distribution of an average will tend to be normal as the sample size increases, regardless of the distribution from which the average is taken, except when the moments of the parent distribution do not exist. All practical distributions in statistical engineering have defined moments, and thus the central limit theorem applies [sta].

4.3.A.3. Measures:

Precision and recall are two widely used metrics for evaluating the correctness of a pattern recognition algorithm. They are an extended version of accuracy, a simple metric that computes the fraction of instances for which the correct result is returned. When using precision and recall, the set of possible labels for a given instance is divided into two subsets, one of which is considered "relevant" for the purposes of the metric. Recall is then computed as the fraction of correct instances among all instances that actually belong to the relevant subset. Precision is the fraction of correct instances among those that the algorithm believes to belong to the relevant subset. In other words, recall (R) is the number of correct results divided by the number of results that should have been returned. Precision (P) is the number of correct results divided by the number of all returned results.

Assuming that S is the set containing all the correct corresponding classes, and A is the set containing the corresponding classes returned by an algorithm, then recall and precision are computed as: $R = |S \cap A| / |S|$ and $P = |S \cap A| / |A|$. In statistics, the F1 score is a measure of a test's accuracy. It is defined as $2PR / (P+R)$, where P is the precision and R is the recall of the test. The F1 score can be interpreted as a weighted average of precision and recall, where the score reaches its best value at 1, and worst value at 0.

4.3.A.4. Statistical Significance (p-value):

The statistical significance of a result is the probability that the observed relationship (e.g., between variables) or a difference (e.g., between means) in a sample occurred by pure chance, and that in the population from which the sample was drawn, no such relationship or differences exist. In other words, we could say that the statistical significance of a result tells us something about the degree to which the result is "true" (in the sense of being "representative of the population").

When statistical significance is computed, the value of the p-value represents a decreasing index of the reliability of a result. The higher the p-value, the less we can believe that the observed relation between variables in the sample is a reliable indicator of the relation between the respective variables in the population. Specifically, the p-value represents the probability of error that is involved in accepting our observed result as valid, i.e. as representative of the population.

For example, a p-value of 0.05 indicates that there is a 5% probability that the relation between the variables found in our sample is by chance. In other words, assuming that in the population there was no relation between those variables whatsoever, and we were repeating experiments such as ours one after another, we could expect that approximately in every 20 replications of the experiment there would be one in which the relation between the variables in question would be equal or stronger than in ours.

4.3.A.4.1. t-Test for Dependent Samples:

Within-group variation: The size of a relation between two variables, such as the one measured by a difference in means between two groups, depends to a large extent on the differentiation of values within the group. Depending on how differentiated the values are in each group, a given raw difference in group means will indicate either a stronger or weaker relationship between the independent (grouping) and dependent variable.

For example, if the mean WCC (White Cell Count) was 102 in males and 104 in females, then this difference of only 2 points would be extremely important if all values for males fell within a range of 101 to 103, and all scores for females fell within a range of 103 to 105; for example, we would be able to predict WCC pretty well based on gender. However, if the same difference of 2 was obtained from very differentiated scores (e.g., if their range was 0-200), then we would consider the difference entirely negligible. That is to say, reduction of the within-group variation increases the sensitivity of our test.

Purpose: The t-test for dependent samples helps us take advantage of one specific type of design in which an important source of within-group variation (error) can be identified and excluded from the analysis. Specifically, if two groups of observations (that are to be compared) are based on the same sample of subjects who were tested twice (e.g., before and after a treatment, or with matching-tool-1 and matching-tool-2), then a considerable part of the within-group variation in both groups of scores can be attributed to the initial individual differences between subjects.

More complex group comparisons (repeated measures ANOVA): If there are more than two correlated samples (e.g., matching-tool-1, matching-tool-2, and matching-tool-3), then analysis of variance (ANOVA) with repeated measures should be used. The repeated measures ANOVA can be considered a generalization of the t-test for dependent samples and it offers various features that increase the overall sensitivity of the analysis.

4.3.B. Specifics in Our Case

4.3.B.1. Our Class Matching Algorithm:

In the experiments that are reported with more details in Section 4.4, we have used ontologies that are developed independently by different organizations. Hence the terms used for various concepts are different, and we need to find the terms that correspond between different ontologies. This task is done using the class matching algorithm, as presented in Section 4.1.

Parameter setting: The main parameter in our algorithm is the threshold that is used to determine whether two classes are a match or not. The threshold was further explained in Section 3.6. We also use different languages in the ontologies to evaluate their effect on the class matching process. Another issue that is considered in our experiments is the number of instances.

4.3.B.2. Four Near-Ideal Pairings:

Ontologies: In our experiments, we have included the ontologies of the 3XX Benchmark from the Ontology Alignment Evaluation Initiative (OAEI), in order to compare our results with other systems. The 3XX Benchmark contains five ontologies, which model various domains related to a university organization and bibliographic items.

Pairings: To create the pairs for matching, a reference ontology (101) is matched against four ontologies, named 301 to 304.

Other tools: This benchmark is often used by researchers for reporting the performance of their systems and comparing it to other systems. We compare our results vis-a-vis a gold standard with six other tools, namely S-Match, OMViaUO, A-API, BLOOMS, AROMA, and RiMoM. The results are reported in Section 4.4. One drawback of the comparison is that we have the raw data for some of these systems and not all of them. Hence, when raw data is not available, we assume the standard deviation of other systems to be the same as the standard deviation of our system.

Gold standard: When measuring the effectiveness of our algorithm for class matching, we need to have a gold standard, to assess our results and determine the correct and incorrect matching classes. The output of the class matching algorithm is compared against the gold standard. Ideally, the gold standard should be developed by a group of independent human reviewers and there should be consensus about their correctness. For the 3XX Benchmark, the OAEI provides the gold standard along with the ontologies.

4.3.B.3. Other Pairings:

Ontologies: The other ontologies, used in our experiments, model various domains of knowledge, for example bibliographic information, publications, universities, conferences and their registration process, publishing processes, biology, travel and leisure, and food and drinks. These diverse ontologies are used, so that we can ensure the applicability of our class matching algorithm and measure its performance in different domains. Since ontologies may be used to model different domains in the real world, algorithms that are designed to process the ontologies for ontology development or interoperability should also perform reasonably in different domains.

Pairings: Three of the ontologies below are in the biology domain. They form two pairs. Three other ontologies are in the food and drinks domain. They also form two pairs. Another pair of ontologies is related to travel and leisure activities. Eight pairs of ontologies are related to publications, conferences, and publishing and registration processes. These ontologies have different levels of expressivity. They are suitable for the ontology matching task because of their heterogeneous origin and varying level of expressivity. For these ontologies, we evaluated our results by comparing it with a gold standard.

Gold standard: The results of our experiments for these other pairings are compared with a gold standard. For these pairs of ontologies, used in our experiments from various domains, we had to create the gold standard manually. In order to create a gold standard, we started with the output of our system, when the system was setup such that it would produce many suggested matches, some of which were not accurate. Then, the output was verified by hand to remove the suggestions that were incorrect. As discussed in Section 4.3.A.1, the gold standard should ideally be created by independent observers.

4.3.C. Other Issues

Size of ontologies: The size of ontologies usually varies from about 5 KB to 100 KB. There are also a few larger ontologies that model some scientific domain, like biology, with the size of about 500 KB. The Biology-1 ontology, used in our experiments, is 557 KB. In rare cases, the size of an ontology may be even larger than 500 KB, for example the UN ontology is over 4 MB.

In general, most computer programs assume that the input can be loaded into memory. The size of the ontology causes some limitations for the processing of extremely large ontologies, due to the amount of memory that is required. For example, we have tried opening the UN ontology on machines with 512 MB, 2 GB, and 16 GB of RAM, and the ontology would not open using the Swoop ontology editor.

Technical ontologies: We have used some scientific ontologies in our experiments, for example in the biology domain. Other similar ontologies could also be used.

Some cases to keep in mind: Consider that in some cases, we may want to match classes that have similar names, like pear and Asian pear, or apple and crab apple. For such cases, the lexical similarity metric can detect the similarity of the names of classes, for example "pear" is in common between pear and Asian pear, and "apple" is in common between apple and crab apple.

We may also want to match classes that have different names, like lorry and truck, or egg plant and aubergine. If there are common instances for these classes, then the extensional similarity metric can detect the similarity of the classes. If there are no common instances to indicate the similarity of these classes, then an alternative could be to use outside resources, for example a dictionary or a repository of previous matches, to help in the matching of such cases.

In general, if we search for "lorry" in Google, we do not get results for "truck." In other words, this is what users expect, when they are searching on the Web.

4.4. Experimental Evaluations

In order to evaluate the effectiveness of our class matching algorithm, we implemented our algorithm in Swoop. Swoop is an open source research tool for browsing and creating ontologies and contains over 20,000 lines of Java code. It is a hypermedia-based ontology editor that employs a web-browser metaphor for its design and usage [Kal05]. In our implementation, Pellet was used for ontology reasoning [Sir07]. Pellet is an open source reasoner written in Java. The experiments were run on a 1.86 GHz Pentium machine with 512 MB of RAM.

The class similarity metrics, formalized in Section 3.6, are inspired by the work in the literature. However, those works are often different from ours. Some works in the literature are in the context of ontology merging efforts, e.g. [Mc00, Noy00, Stu01]. Some other works, which are not in the context of merging, are addressing ontology matching only (i.e. Step 1, in Section 3.2). These works do not necessarily focus on classes only, and they consider other entities as well, e.g. [Ehr04]. Also, for the class matching part, the work often attempts to find subsumption relationships between classes in the ontologies (e.g. [Bou04]), which is different from measuring class similarity. Our algorithm for class matching directly measures class similarity. Note that measuring class similarity is necessary, in our algorithm, since the output of the class matching algorithm (presented in Section 4.1) is later used, as the input of the skeleton creation algorithm (presented in Section 4.2).

4.4.A. Four Near-Ideal Pairings

In order to evaluate the performance of our class matching algorithm and compare its results with other systems, we used the 3XX Benchmark of the Ontology Alignment Evaluation Initiative (OAEI) data set. It contains four ontologies, which model various domains related to a university organization and bibliographic items. A reference ontology (101) is matched against four other ontologies, named 301 to 304. The number of matching classes, true positive, false positive, precision, recall, and F1 are reported for various thresholds, which provide different results. Table 4.1 shows the results of these experiments.

Table 4.1. The results of our MATCH algorithm on the OAEI Benchmark.

101-301	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	16	16	16	16	16	16
	TP	13	13	13	13	13	13
	FP	19	10	9	8	6	1
	Precision	40.625	56.52174	59.09091	61.90476	68.42105	92.85714
	Recall	81.25	81.25	81.25	81.25	81.25	81.25
	F1	54.16667	66.66667	68.42105	70.27027	74.28571	86.66666
101-302	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	11	11	11	11	11	11
	TP	10	9	9	9	9	9
	FP	23	11	9	8	6	1
	Precision	30.30303	45	50	52.94118	60	90
	Recall	90.9091	81.81818	81.81818	81.81818	81.81818	81.81818
	F1	45.45454	58.06452	62.06897	64.28571	69.23077	85.71429
101-303	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	16	16	16	16	16	16
	TP	14	14	14	14	14	12
	FP	53	27	19	13	10	1
	Precision	20.89552	34.14634	42.42424	51.85185	58.33333	92.30769
	Recall	87.5	87.5	87.5	87.5	87.5	75
	F1	33.73494	49.12281	57.14286	65.11628	70	82.75862
101-304	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	28	28	28	28	28	28
	TP	27	27	27	27	27	27
	FP	47	24	20	17	12	4
	Precision	36.48649	52.94118	57.44681	61.36364	69.23077	87.09677
	Recall	96.42857	96.42857	96.42857	96.42857	96.42857	96.42857

	F1	52.94118	68.35443	72	75	80.59702	91.52542

Then the mean precision, recall, and F1 are computed for the four ontologies from the 3XX Benchmark. These results are reported in Table 4.2. The mean F1 for our system on the 3XX Benchmark is 0.73. As shown in Table 4.3, our mean F1 value is higher than the F1 of four other systems on the 3XX Benchmark, which are 0.13, 0.28, 0.56, and 0.71. Our mean F1 value is smaller than two other systems, which are 0.77 and 0.81.

Table 4.2. The mean of results of the 3XX OAEI Benchmark ontologies, for our MATCH algorithm.

	101-301	101-302	101-303	101-304	Mean for 3XX	s*s
Threshold	0.9	0.9	0.9	0.9	0.9	
#M	16	11	16	28		
TP	13	9	14	27		
FP	6	6	10	12		
Precision	0.684211	0.6	0.583333	0.692308	0.639963	
Recall	0.8125	0.818182	0.875	0.964286	0.867492	
F1	0.742857	0.692308	0.7	0.80597	0.735284	0.052108

Table 4.3. Comparison of results of six matching tools on the 3XX OAEI Benchmark ontologies.

	S-Match	OMViaUO	A-API	BLOOMS	MATCH	AROMA	RiMoM
Precision	0.1	0.28	0.45	0.62	0.639963	0.8	0.81
Recall	0.2	0.28	0.77	0.84	0.867492	0.76	0.82
F1	0.133333	0.28	0.568033	0.713425	0.735284	0.779487	0.814969

Now, we perform a repeated measures ANOVA to determine the statistical significance of the difference between the F1 values of various tools. The F1 values are shown in Table 4.3. We want to know if the data provides sufficient evidence that the difference in F1 values for at least two of these tools is not due to chance. The analysis below shows that there is sufficient evidence to indicate that the observed difference in F1 values for at least two of the matching tools is not due to chance. The computation is shown in Table 4.4.

Then, we perform a t-test for dependent samples on various pairs of tools, to compare our tool with other tools (a.k.a. Bonferroni test). The analysis below shows that there is sufficient evidence to indicate that the F1 of our tool is larger than the F1 of S-Match and OMViaUO ($p<=0.05$), and this is not due to chance. No conclusion can be made with regard to the other four tools. The computation is shown in Table 4.4.

Table 4.4. Statistical tests for comparison with other tools.

	n	F1	s*s	Mean F1	n-1
S-Match	4	0.133333	0.052108	0.574933	3
OMViaUO	4	0.28	0.052108	0.574933	3
A-API	4	0.568033	0.052108	0.574933	3
BLOOMS	4	0.713425	0.052108	0.574933	3
MATCH	4	0.735284	0.052108	0.574933	3
AROMA	4	0.779487	0.211949	0.574933	3
RiMoM	4	0.814969	0.102315	0.574933	3
For the tools that we do not have their raw data, we assume that their standard deviation is the same as ours.					
Mean F1	0.574933				
Does the data provide sufficient evidence to indicate that the F1 differs for at least two of the matching tools and this is not due to chance?					
Ha	F1 of the matching tools differ for at least two of the tools				

H0	F1 of the matching tools do not differ				
Test Statistic:					
F					
n	4				
p	7				
df(numerator)	6				
df(denominator)	21				
Alpha	0.05				
Critical F	2.572712				
SST	1.705582				
MST	0.284264				
SSE	1.724416				
MSE	0.082115				
Source	df	SS	MS	F	
Treatments	6	1.705582	0.284264	3.461774	
Error	21	1.724416	0.082115		
Total	27	3.429998			
Rejection Rule:					
F	3.461774	>=	Critical F		
Decision: Reject H0					
Conclusion: There is sufficient evidence to indicate that the F1 differs for at least two of the matching tools, and this is not due to chance.					
Which pairs of means differ?					
Bonferroni Test is done for pairs of means					
Decision rule: Reject H_o, if the interval does not contain 0.					
c	6			S-Match	0.133333
Alpha/2c	0.15			OMViaUO	0.28
df	27			A-API	0.568033
t(0.15)	1.057			BLOOMS	0.713425
				MATCH	0.735284
Delta	0.214176			AROMA	0.779487
				RiMoM	0.814969
Ha	S-Match!=MATCH				
interval	-0.81613	-0.38777	Reject H0, MATCH is larger than S-Match		
Ha	OMViaUO!=MATCH				
interval	-0.66946	-0.24111	Reject H0, MATCH is larger than OMViaUO		

Ha	A-API!=MATCH				
interval	-0.38143	0.046925	Do not reject H0		
Ha	BLOOMS!=MATCH				
interval	-0.23604	0.192317	Do not reject H0		
Ha	AROMA!=MATCH				
interval	-0.16997	0.25838	Do not reject H0		
Ha	RiMoM!=MATCH				
interval	-0.13449	0.293862	Do not reject H0		
Conclusion: There is sufficient evidence to indicate that the F1 of MATCH is larger than the F1 of S-Match and OMViaUO, and this is not due to chance.					

4.4.B. University Ontologies

The results of our experimental trials in Figure 4.2 to 4.6 are for two real-world ontologies. The ontologies were developed separately by different organizations. This dataset is selected from the datasets that are provided for the Ontology Alignment Evaluation Initiative [OAEI]. One ontology is from Karlsruhe [Kar] and is used in the Ontoweb portal. It is a refinement of other ontologies such as $(KA)^2$. It defines the terms used in bibliographic items and a university organization. The other ontology is from INRIA [Inr] and is designed based on the BibTeX in OWL ontology and the Bibliographic XML DTD. Its goal is to easily gather various RDF items. These items are usually BibTeX entries found on the web, which are transformed into RDF according to this ontology. The ontologies have 24 corresponding classes. Table 4.5 shows the characteristics of the ontologies.

Table 4.5. The characteristics of the university ontologies.

	University	Publication

	Ontology	Ontology
# Classes	64	48
# Properties	72	58
# Individuals	68	59
Min. Depth of Class Tree	1	1
Max. Depth of Class Tree	5	4
Average Depth of Class Tree	2.4	2.3
Min. Branching of Class Tree	1	1
Max. Branching of Class Tree	13	17
Average Branching Factor of Class Tree	3.15	3.25

When computing the lexical similarity metric for comparing the name of classes in two ontologies, various string similarity measures can be used. The results show that the performance of these measures varies considerably, as illustrated in Figure 4.2. The Jaro-Winkler measure shows a more robust behavior for finding corresponding classes in ontologies, based on class name. By decreasing the threshold in the class matching algorithm for each string similarity measure, the recall increases, the precision decreases, and various precision-recall performance levels are achieved, as shown by the plots on the curves, in Figure 4.2. Usually, there is a precision-recall tradeoff, and precisions below the 60 percent level are not very useful, since many of the detected matches would then be incorrect.

In Figure 4.2, by decreasing the threshold for identifying a match, we can increase the recall rate to some extend. However, as the diagram demonstrates, it is not possible to increase the recall to above 80 percent, by only decreasing the threshold, since this would cause a sharp drop in precision, i.e. introduce many incorrect results.

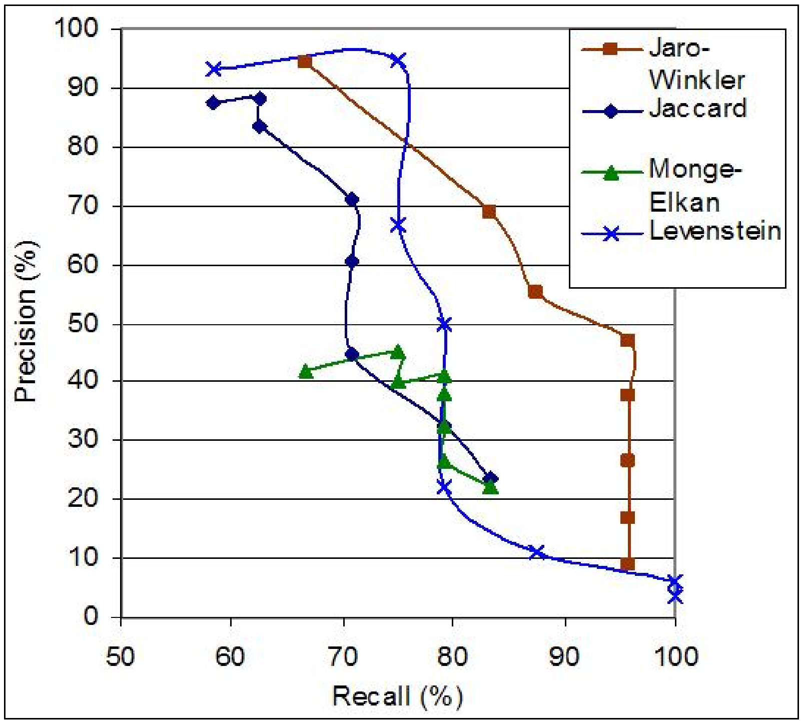

Figure 4.2. Performance of various string similarity measures for finding corresponding classes in ontologies, based on name.

In real-world applications, the issue of setting the threshold for identifying corresponding classes is an important one, and it may require human judgment. In principle, ontologies can cover any domain of knowledge, and the nature of data instances is diverse in different applications. Hence, it is difficult to provide general guidelines on how to set the threshold for all ontologies. Essentially, the threshold needs to be determined experimentally for each application and dataset.

Figure 4.3 shows the running time required for computing the lexical similarity of classes in both ontologies using various string similarity measures. Jaro-Winkler (which shows better precision-recall performance in Figure 4.2) takes 172 ms to compute and lies approximately in between the other string similarity measure, in terms of running time.

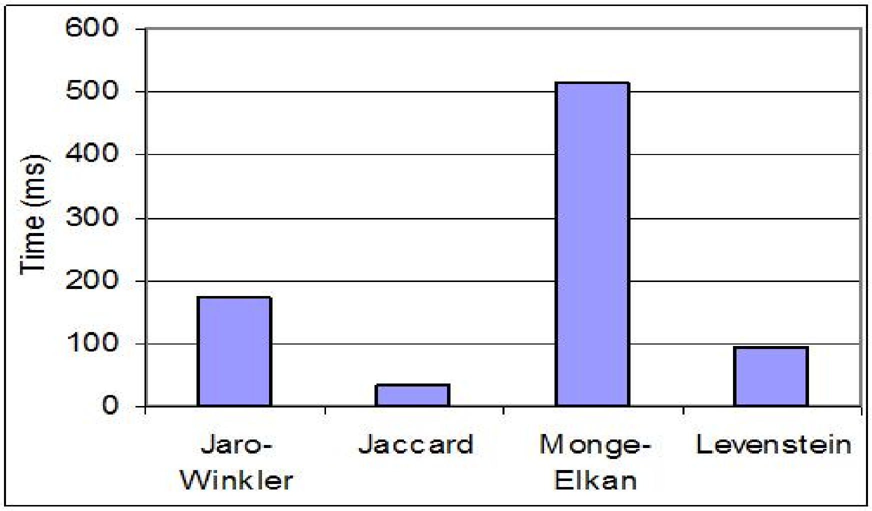

Figure 4.3. Running time of computing the lexical similarity of classes using various string similarity measures.

By using the lexical similarity of classes, measured by the Jaro-Winkler similarity measure, 16 out of the 24 corresponding classes could be identified (i.e. true positives - TP), as shown in Figure 4.4. Decreasing the detection threshold of the lexical similarity metric would decrease the precision and increase the number of false positives (FP). To tackle this problem and find more matching classes (without decreasing the threshold), we also employed the other similarity metrics namely, extensional, extensional closure and global path similarity metrics. The results in Figure 4.4 are cumulative from left to right, and each similarity metric is added to the previous ones. Our experiments show that utilizing these additional metrics helps in finding more correct matching classes (true positives). At the same time, they do not reduce the precision, by introducing many false positives, similar to when the detection threshold is decreased (refer to Figure 4.2).

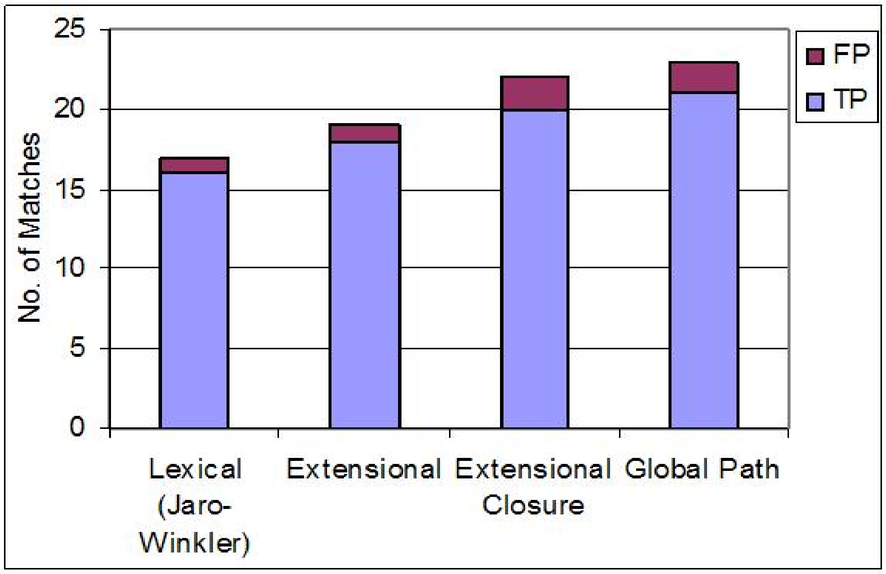

Figure 4.4. Detection of more matching concepts using additional concept similarity metrics, such as extensional, extensional closure, and global path.

Computing the lexical similarity of classes does not require reasoning. However, the reasoner needs to be used to compute the extensional, extensional closure, and global path similarity metrics. For computing the extensional closure, we need to classify the ontology to find the subclasses, and also perform realization to retrieve the instances of all the subclasses. To perform reasoning, the ontology must be consistent, and all classes must be satisfiable. Hence, activating the reasoner in fact triggers all the above steps and accounts for most of the running time. The time required for the rest of the computation, which involves the comparison of retrieved instances or the comparison of the classes in a path, is relatively small. As shown in

Figure 4.5, the running time for computing the lexical similarity is small, compared to the other three similarity metrics, which require reasoning.

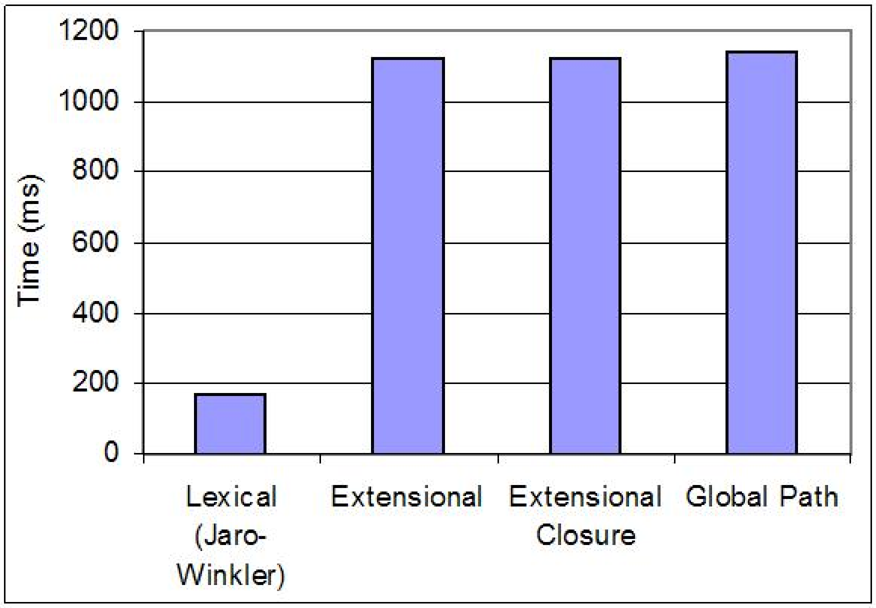

Figure 4.5. Running time of computing lexical, extensional, extensional closure and global path similarity metrics.

The results in Figure 4.6 are cumulative from left to right, and each similarity metric is added to the previous ones. The precision and recall bars for Lexical (Jaro-Winkler) in Figure 4.6, are showing the same precision and recall values, as the first point on the Jaro-Winkler curve in Figure 4.2. Also, for all the experiments in Figure 4.6, we use the same threshold, as the first point on the Jaro-Winkler curve in Figure 4.2. Hence, in Figure 4.6, the threshold does not change, and there is no precision-recall curve (unlike Figure 4.2).

Figure 4.6 shows that using the extensional, extensional closure, and global path similarity metrics, in addition to lexical, improves the recall. At the same time, in Figure 4.6, the precision remains almost the same.

Note that the recall can also be improved by decreasing the threshold in Figure 4.2 - however that considerably reduces the precision of results (as shown in Figure 4.2). In Figure 4.6, by using additional similarity metrics, we can improve the recall, without decreasing the threshold and losing precision. Therefore, we effectively overcome the precision-recall tradeoff, as evident by the increase in the F1 quality measure. This demonstrates that using the additional class similarity metrics helps in finding more corresponding classes and achieving better results.

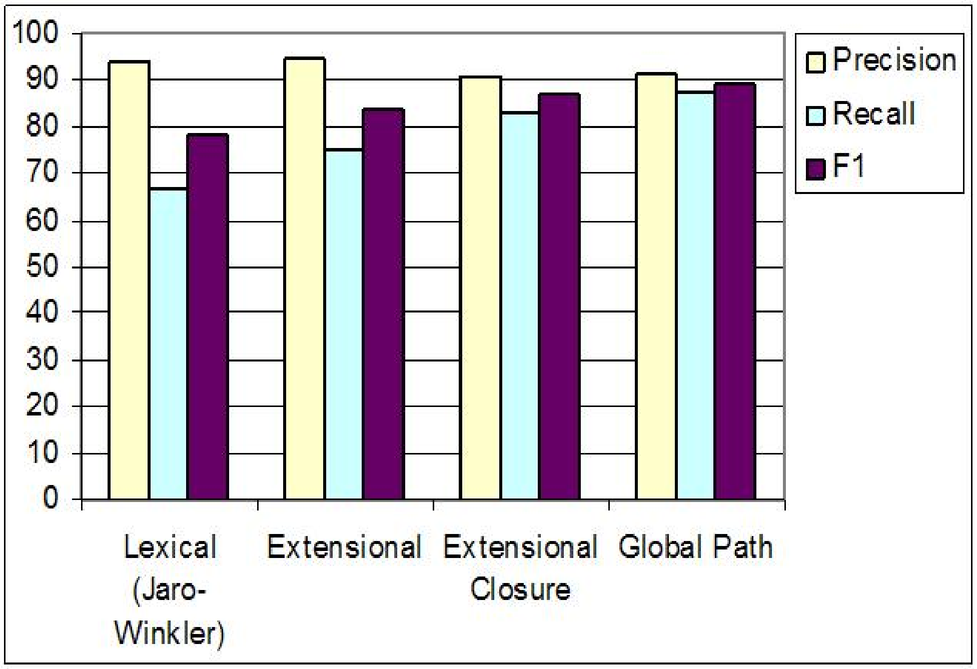

Figure 4.6. Using extensional, extensional closure and global path similarity metrics, in addition to lexical, increases the recall and F1 quality measure.

4.4.C. Ontologies from Different Languages

The results of experiments reported in Figure 4.7 to 4.10 are for two ontologies that are in two different languages. One ontology is in English and the other one is in French. The characteristics of the ontologies are shown in Table 4.6.

Table 4.6. The characteristics of the languages ontologies.

	English	French
# Classes	50	50
# Matching Classes	20	20
# Individuals	200	200

The use of the lexical similarity metric for the English and French ontology does not yield any matches, as the name of concepts are not the same. This is shown in Figure 4.7. Such a scenario happens sometimes in practice, as it is necessary to find correspondences between ontologies that are in different languages. In these scenarios, an effective clue that provides additional information is the common instances. By using the extensional similarity metric for these ontologies, we were able to detect eight corresponding classes (true positives), as shown in Figure 4.7. We also detected three incorrect correspondences (false positives). The results in Figure 4.7 are cumulative from left to right, and each similarity metric is added to the previous ones. In this experiment, there is no difference when adding the extensional closure and global path.

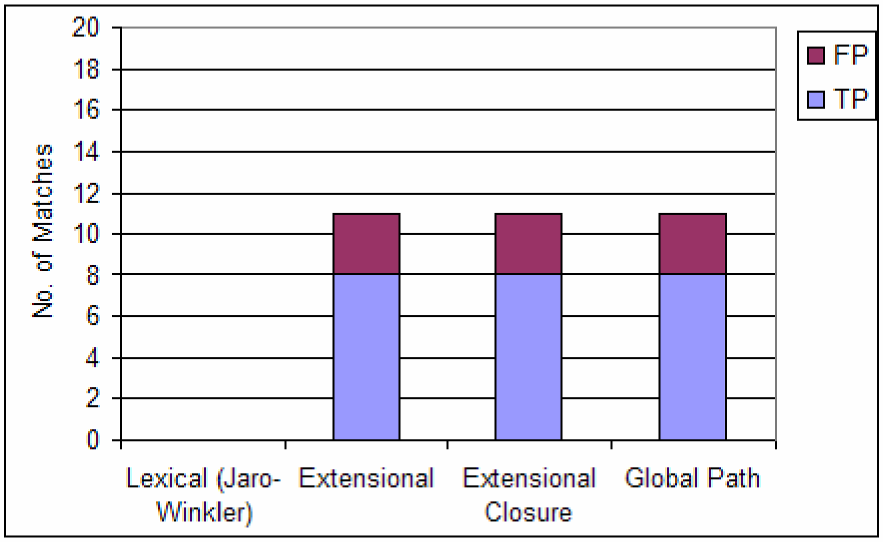

Figure 4.7. The number of matches found using different class similarity metrics for two ontologies in different languages.

In applications where the ontologies are in two different languages, we can effectively enhance the recall and F1 measure using the extensional similarity metric, as shown in Figure 4.8. These results are cumulative from left to right, and each similarity metric is added to the previous ones. In this experiment, there is no difference when adding the extensional closure and global path.

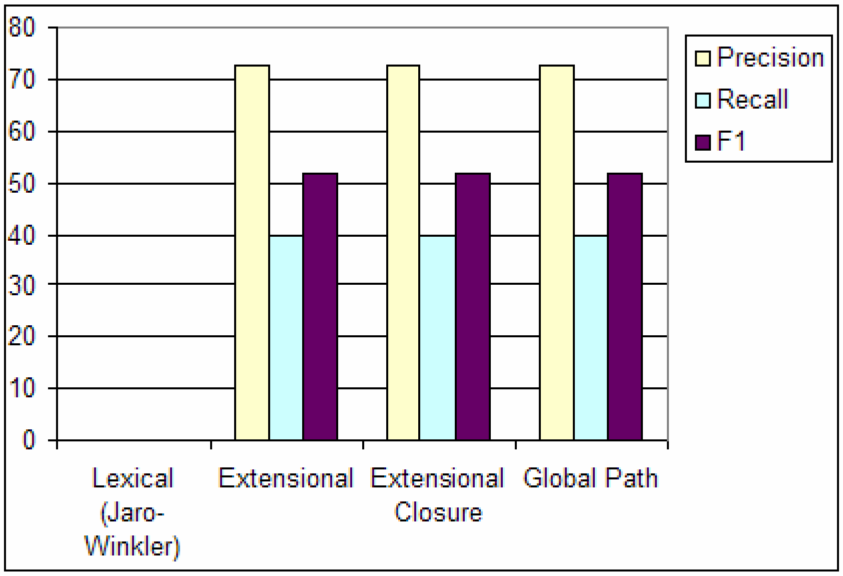

Figure 4.8. While the lexical similarity metric can not find any matches, the use of the extensional class similarity metric enhances the recall and F1 quality measure.

In order to measure the effect of the number of instances on the extensional similarity metric, we gradually removed from 10 to 50 percent of the total number of instances from both ontologies. The recall of the extensional similarity metric dropped from 40 to 30 percent. The results are shown in Figure 4.9. We can see that the more instances we have in the ontologies, the more likely it is for the extensional similarity metric to yield good results.

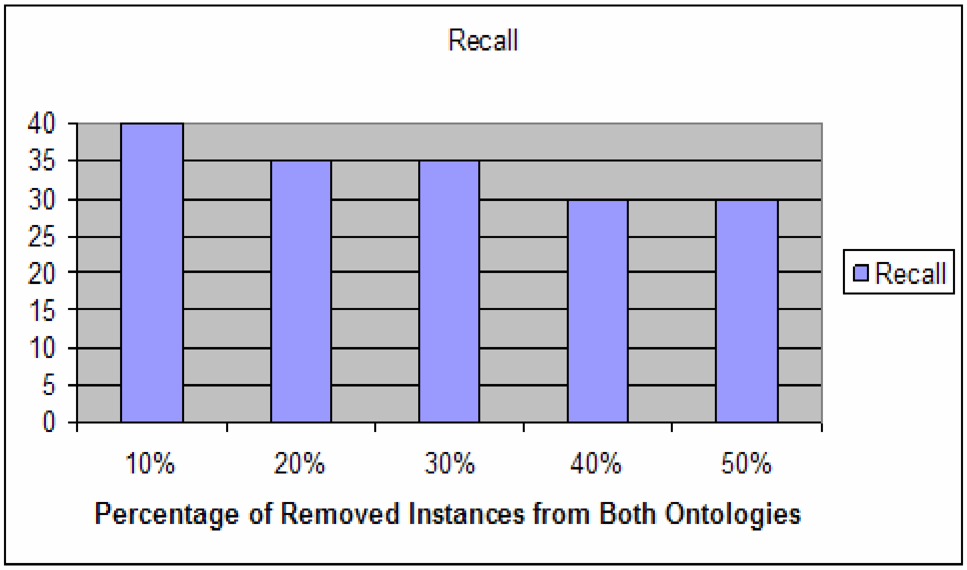

Figure 4.9. The effect of the number of instances on the recall of extensional similarity metric, when removing instances from both ontologies.

Similar to the previous experiment, we again gradually removed from 10 to 50 percent of the total number of instances, but this time from one ontology (and not from both ontologies). The recall of the extensional similarity metric dropped from 40 to 20 percent. The results are shown in Figure 4.10. We can see that the drop in the recall of the extensional similarity metric, in Figure 4.10, is at a higher rate than Figure 4.9. Hence, the extensional similarity metric works more effectively, when the instances are distributed uniformly between the two ontologies.

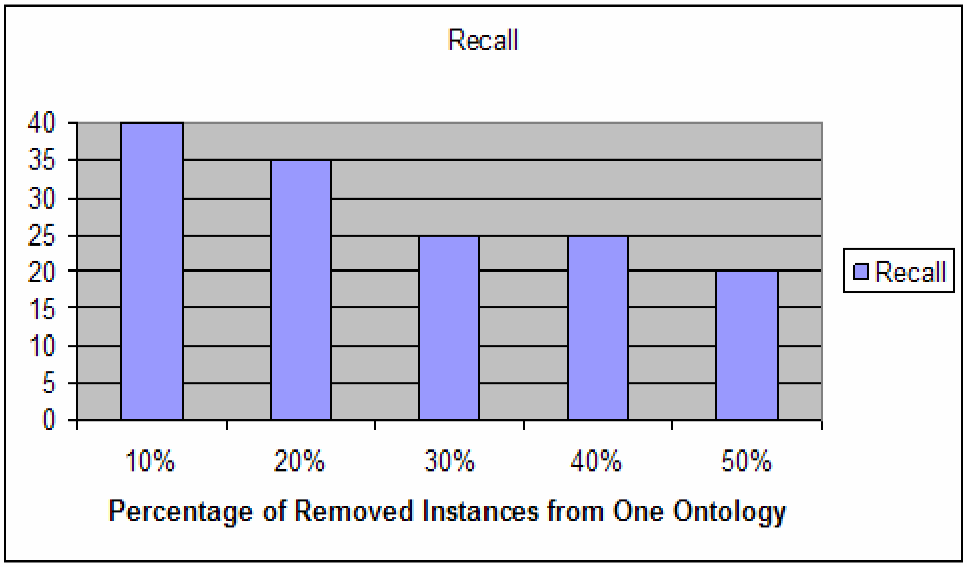

Figure 4.10. The effect of the number of instances on the recall of extensional similarity metric, when removing instances from one ontology.

4.4.D. Business Ontologies

The results of experiments reported in Figure 4.11 and 4.12 are for two ontologies that are modeling two business companies. The characteristics of the ontologies are shown in Table 4.7.

Table 4.7. The characteristics of the company ontologies.

	Paragon	Apertum
# Classes	47	47
# Matching Classes	9	9
# Individuals	69	69

We can find four matching classes using the lexical similarity metric for the company ontologies (true positives). This is shown in Figure 4.11. We also detected one incorrect correspondence (false positive). By using the extensional similarity metric for these ontologies, we were able to detect two additional corresponding classes, as shown in Figure 4.11. The results in Figure 4.11 are cumulative from left to right, and each similarity metric is added to the previous ones. In this experiment, there is no difference when adding the extensional closure and global path.

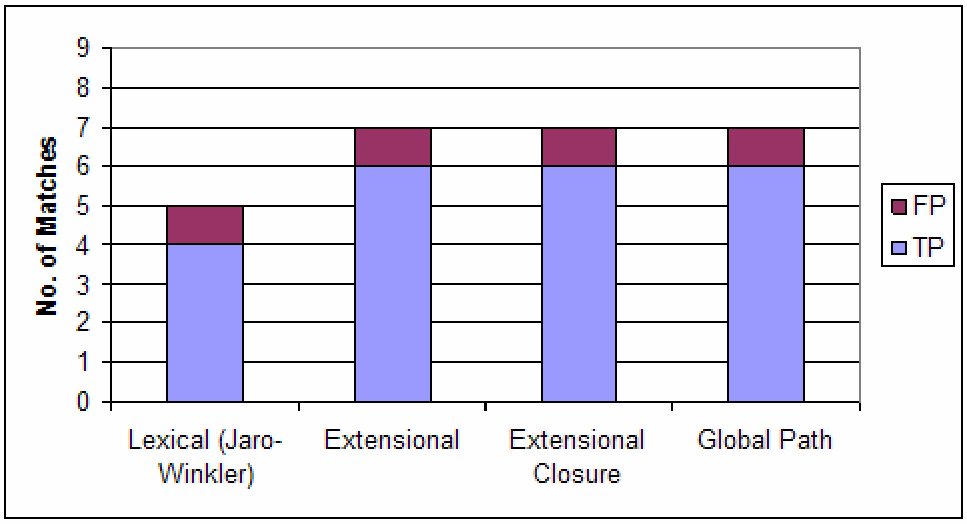

Figure 4.11. The number of matches found using different class similarity metrics for the two company ontologies.

For the company ontologies, the extensional similarity metric enhances the recall and F1 measure, as shown in Figure 4.12. These results are cumulative from left to right, and each similarity metric is added to the previous ones. In this experiment, there is no difference when adding the extensional closure and global path.

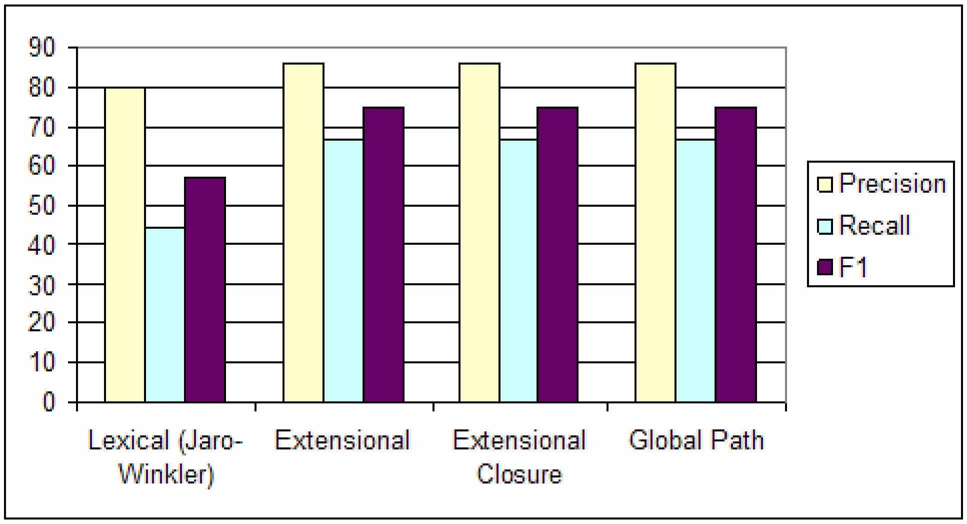

Figure 4.12. The use of the extensional class similarity metric enhances the recall and F1 quality measure.

4.4.E. Ontologies from Other Domains

We also used the class matching algorithm on a variety of other ontologies, from various domains and designed by independent organizations. The first eight pairs of ontologies are related to publications, conferences, and publishing procedures. These ontologies have different levels of expressivity. More details about these pairs of ontologies are provided in Table 4.8. They are suitable for the ontology matching task because of their heterogeneous origin and varying level of expressivity. Also, three of the ontologies below are in the biology domain. They form two pairs. Three other ontologies are in the food domain. They also form two pairs. Another pair of ontologies is related to travel activities. The results of the MATCH algorithm are reported in Table 4.9.

Table 4.8. The names of various ontologies and their features.

Name	# Classes	# Datatype Properties	# Object Properties	DL expressivity	Related link
Crs	14	2	15	ALCIF(D)	http://www.conferencereview.com
Edas	104	20	30	ALCOIN(D)	http://edas.info/
Paperdyne	47	21	61	ALCHIN(D)	http://www.paperdyne.com/
Sigkdd	49	11	17	ALEI(D)	http://www.acm.org/sigs/sigkdd/kdd2006
Iasted	140	3	38	ALCIN(D)	http://iasted.com/conferences/2005/cancun/ms.htm
Micro	32	9	17	ALCOIN(D)	www.microarch.org
MyReview	39	17	49	ALCOIN(D)	http://myreview.intellagence.eu/
Pcs	23	14	24	ALCIF(D)	http://precisionconference.com
Cocus	55	0	35	ALCIF	http://cocus.create-net.it/
Confious	57	5	52	SHIN(D)	http://www.confious.com
Cmt	36	10	49	ALCIN(D)	http://msrcmt.research.microsoft.com/cmt

OpenConf	62	21	24	ALCOI(D)	http://www.zakongroup.com/technology/openconf.shtml
Conference	47	21	61	ALCHIN(D)	NotAvailable
Ekaw	77	0	33	SHIN	http://ekaw.vse.cz
ConfOf	39	17	49	ALCOIN(D)	NotAvailable
Linklings	37	16	31	SROIQ(D)	http://www.linklings.com/
Biology 1	236	52	87	ALOF(D)	http://mged.sourceforge.net/ontologies/MGEDOntology.owl
Biology 2	142	1	69	SHIF(D)	http://secse.atosorigin.es/10000/ontologies/umlssn.owl
Biology 3	331	32	63	RDFS(DL)	http://ontotext.com/kim/kimo.rdfs.xml
Travel 1	165	12	115	RDFS(DL)	http://www.aifb.uni-karlsruhe.de/WBSmeh/mappingdatarussia1b.rdf
Travel 2	154	7	74	RDFS(DL)	http://www.aifb.uni-karlsruhe.de/WBSmeh/mappingdatarussia1a.rdf
Food 1	65	0	9	ALCOF	http://www.w3.org/2001/sw/WebOnt/guide-src/food
Food 2	51	3	9	ALHI(D)	http://www.purl.org/net/ontology/beer
Food 3	138	1	17	SHION(D)	http://www.w3.org/2001/sw/WebOnt/guide-src/wine

Table 4.9. The results of the MATCH algorithm on different pairs of ontologies from diverse domains, with different levels of expressivity, and created by independent organizations.

1 - crs edas	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	7	7	7	7	7	7
	TP	7	7	7	7	7	7
	FP	33	23	18	16	9	0
	Precision	17.5	23.33333	28	30.43478	43.75	100
	Recall	100	100	100	100	100	100
	F1	29.78723	37.83784	43.75	46.66667	60.86956	100
2 - paperdyne sigkdd	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	9	9	9	9	9	9
	TP	9	9	9	9	9	9
	FP	27	18	12	12	7	0
	Precision	25	33.33333	42.85714	42.85714	56.25	100
	Recall	100	100	100	100	100	100
	F1	40	50	60	60	72	100
3 - iasted MICRO	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	8	8	8	8	8	8
	TP	8	8	8	8	8	8
	FP	58	44	33	23	13	0
	Precision	12.12121	15.38462	19.5122	25.80645	38.09524	100
	Recall	100	100	100	100	100	100
	F1	21.62162	26.66667	32.65307	41.02564	55.17241	100
4 - MyReview PCS	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	12	12	12	12	12	12
	TP	12	12	12	12	12	12
	FP	26	17	15	13	5	0
	Precision	31.57895	41.37931	44.44444	48	70.58823	100
	Recall	100	100	100	100	100	100
	F1	48	58.53659	61.53846	64.86487	82.75862	100

5 - Cocus confious	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	9	9	9	9	9	9
	TP	9	9	9	9	9	9
	FP	44	25	24	20	7	1
	Precision	16.98113	26.47059	27.27273	31.03448	56.25	90
	Recall	100	100	100	100	100	100
	F1	29.03226	41.86047	42.85714	47.36842	72	94.73684
6 - cmt OpenConf	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	4	4	4	4	4	4
	TP	4	4	4	4	4	4
	FP	45	35	28	19	4	0
	Precision	8.163265	10.25641	12.5	17.3913	50	100
	Recall	100	100	100	100	100	100
	F1	15.09434	18.60465	22.22222	29.62963	66.66666	100
7 - Conference ekaw	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	13	13	13	13	13	13
	TP	13	13	13	13	13	13
	FP	176	129	118	98	30	2
	Precision	6.878307	9.154929	9.923664	11.71171	30.23256	86.66666
	Recall	100	100	100	100	100	100
	F1	12.87129	16.77419	18.05556	20.96774	46.42857	92.85714
8 - confOf linklings	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	6	6	6	6	6	6
	TP	6	6	6	6	6	6
	FP	43	22	11	7	3	0
	Precision	12.2449	21.42857	35.29412	46.15385	66.66666	100
	Recall	100	100	100	100	100	100
	F1	21.81818	35.29412	52.17391	63.1579	80	100
Biology1 Biology2	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	6	6	6	6	6	6
	TP	6	6	6	6	6	1
	FP	176	94	59	20	0	0
	Precision	3.296703	6	9.230769	23.07692	100	100
	Recall	100	100	100	100	100	16.66667
	F1	6.382979	11.32076	16.90141	37.5	100	28.57143

Biology1 Biology3	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	9	9	9	9	9	9
	TP	9	9	9	9	9	3
	FP	737	253	91	37	3	1
	Precision	1.206434	3.435115	9	19.56522	75	75
	Recall	100	100	100	100	100	33.33334
	F1	2.384106	6.642067	16.51376	32.72728	85.71429	46.15385
Biology2 Biology3	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	3	3	3	3	3	3
	TP	3	3	3	3	3	2
	FP	158	54	15	6	0	0
	Precision	1.863354	5.263158	16.66667	33.33333	100	100
	Recall	100	100	100	100	100	66.66667
	F1	3.658536	10	28.57143	50	100	80
Travel1 Travel2	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	57	57	57	57	57	57
	TP	57	57	57	57	57	52
	FP	320	139	63	35	6	1
	Precision	15.11936	29.08163	47.5	61.95652	90.47619	98.11321
	Recall	100	100	100	100	100	91.22807
	F1	26.26728	45.05929	64.40678	76.51007	95	94.54545
Food2 Food3	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	2	2	2	2	2	2
	TP	2	2	2	2	2	1
	FP	74	31	17	8	1	0
	Precision	2.631579	6.060606	10.52632	20	66.66666	100
	Recall	100	100	100	100	100	50
	F1	5.128205	11.42857	19.04762	33.33333	80	66.66666
Food1 Food3	Threshold	0.7	0.75	0.8	0.85	0.9	0.95
	#M	112	112	112	112	112	112
	TP	112	112	112	112	112	77
	FP	374	282	204	132	36	6
	Precision	23.04527	28.4264	35.44304	45.90164	75.67567	92.77109
	Recall	100	100	100	100	100	68.75
	F1	37.45819	44.26878	52.33645	62.92134	86.15385	78.97436

Our experimental results on ontologies from various domains presented in Table 4.9 demonstrate that setting the threshold depends on the specific ontology. A reasonable threshold like 0.9, which performs well on some ontologies (i.e. yields a F1 value of 100), may not work well on another ontology (i.e. yield a F1 value of 46). This is demonstrated in Figure 4.13. Hence for each matching scenario, the user needs to determine the appropriate threshold depending on the ontology that is being matched.

The data also shows that for a threshold of 0.9, the last six ontology pairs tend to have a greater F1 value than the first eight ontology pairs. That is probably because the gold standards for the last six pairs were created using an initial threshold of 0.9.

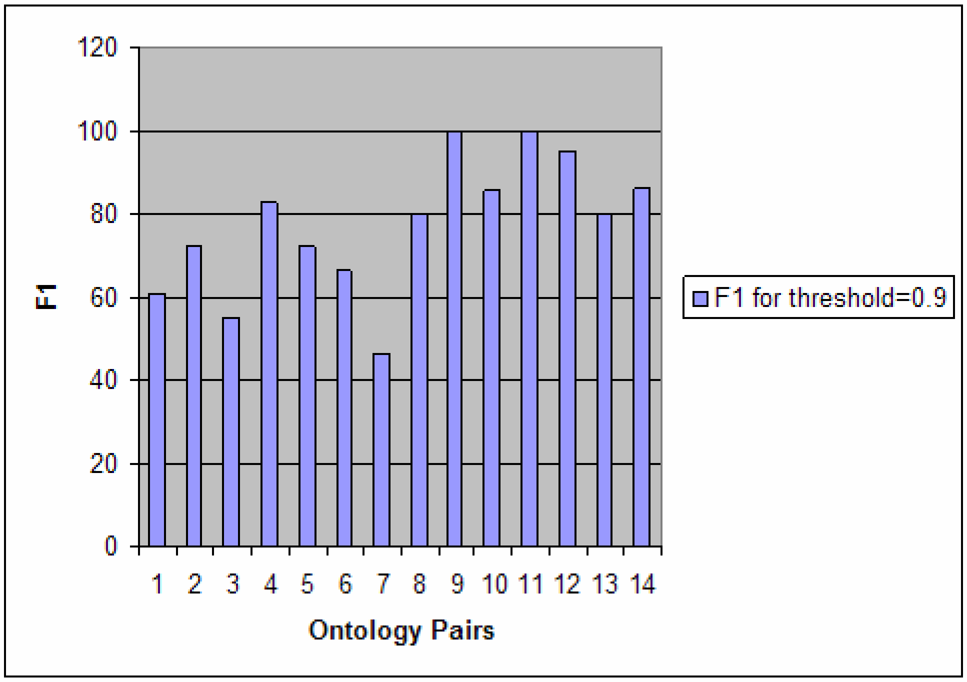

Figure 4.13. A reasonable threshold like 0.9 performs very differently on different ontologies.

The data in Table 4.9 can also be used to test whether our MATCH algorithm is likely to provide consistent results on ontologies from various domains. We performed a Lilliefors test, on the various F1 values that are achieved using our system and presented in Table 4.9. The Lilliefors test checks the default null hypothesis that the sample comes from a normal distribution, against the alternative hypothesis that it does not come from a normal distribution. The test returns the logical value h = 1 if it rejects the null hypothesis at the 5% significance level, and h = 0 if it cannot. The Lilliefors test is a 2-sided goodness-of-fit test, suitable when a fully-specified null distribution is unknown and its parameters must be estimated.

For our data in Table 4.9, the Lilliefors test returns a value of h = 0, hence we can not reject the null hypothesis that the sample comes from a normal distribution. In other words it is not unreasonable to suppose that the distribution of F1 values is very close to normal. Then we compute the cumulative distribution function of this normal distribution, and the result is shown in Figure 4.14.

From Figure 4.14, we can see the probability that our class matching algorithm would achieve an F1 value that is above a given level. For example, our algorithm will achieve an F1 value of above 50 with the probability of 70 percent, in the general population of ontologies. It will achieve an F1 value of above 80 with the probability of 10 percent.

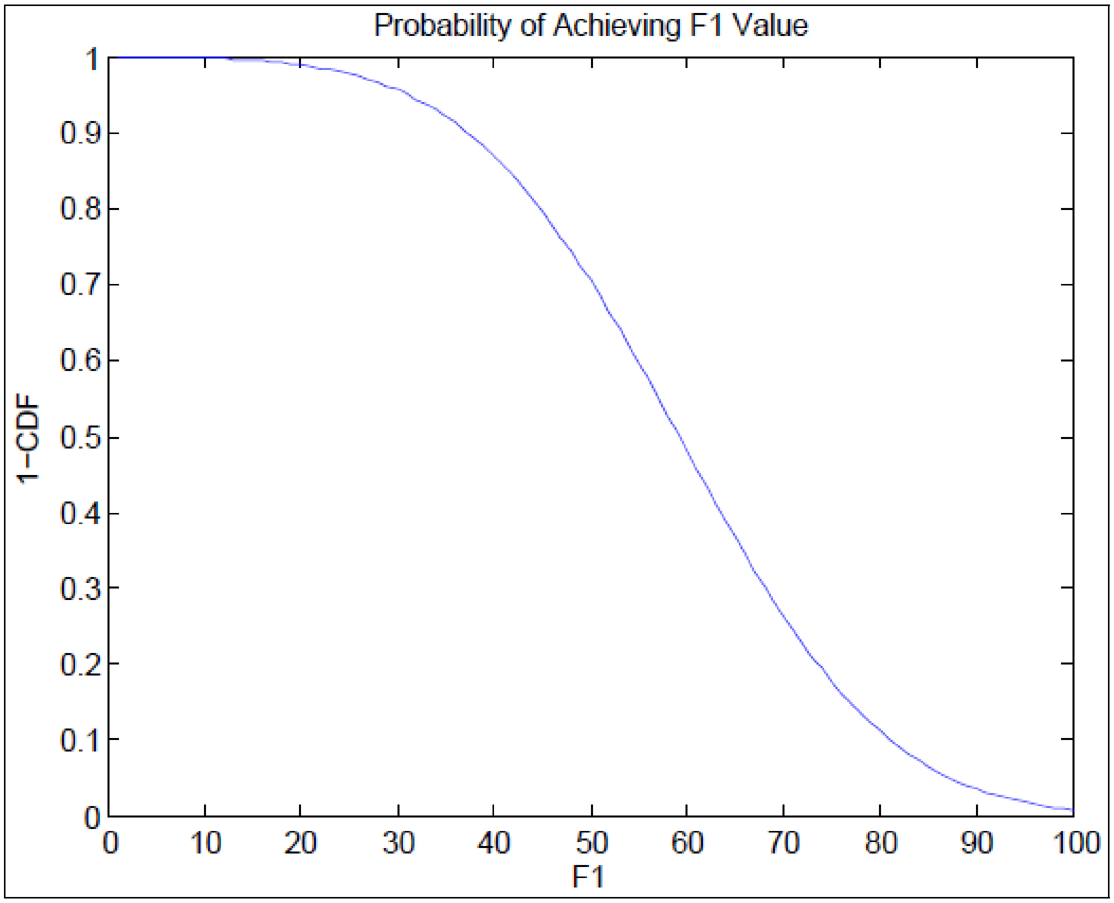

Figure 4.14. The probability of achieving an F1 value that is above a given level, using our class matching algorithm.

4.4.F. Queries

After performing class matching on the university ontologies, we created a skeleton to connect the two ontologies. Ten queries that consist of looking for different items in the two ontologies were considered. When using the skeleton, for all the queries, the results are available in about four seconds. Four seconds is approximately the time that it takes to run the class matching and create a skeleton. Then, two human users were asked to answer these ten queries to look for different items in the two ontologies. Each query was performed by the human. The time that was required for the human to answer each query was measured (i.e. without the skeleton and without the matching). The results are shown in Figure 4.15. The human required more time for answering the queries.

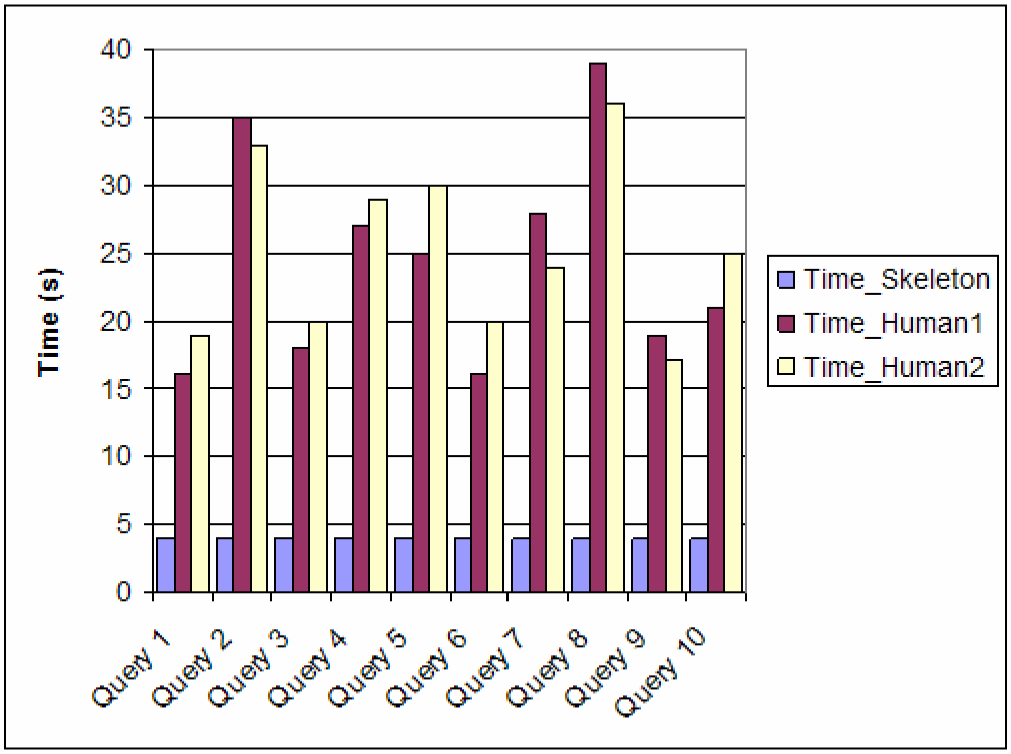

Figure 4.15. The time that was required for a human to answer ten different queries, without using a skeleton and class matching.

The time that is required to execute the class matching algorithm and create a skeleton is about four seconds. The algorithms can find twenty one matching classes out of twenty four and create a skeleton quickly. When two human users were asked to find the matching classes in these two ontologies, this task required more time. The results of this experiment are shown in Figure 4.16. After about five minutes, the humans can find twelve matches. The recall for the class matching algorithm is 0.875, while the recall for the human users is 0.50. The precision for the class matching algorithm is 0.91, while the precision for the human users is 0.77. The number of false positives for the class matching algorithm is 2, while the number of false positives for the human users is 3.5 on average. The F1 for the class matching algorithm is 0.89, while the F1 for the human users is 0.61.

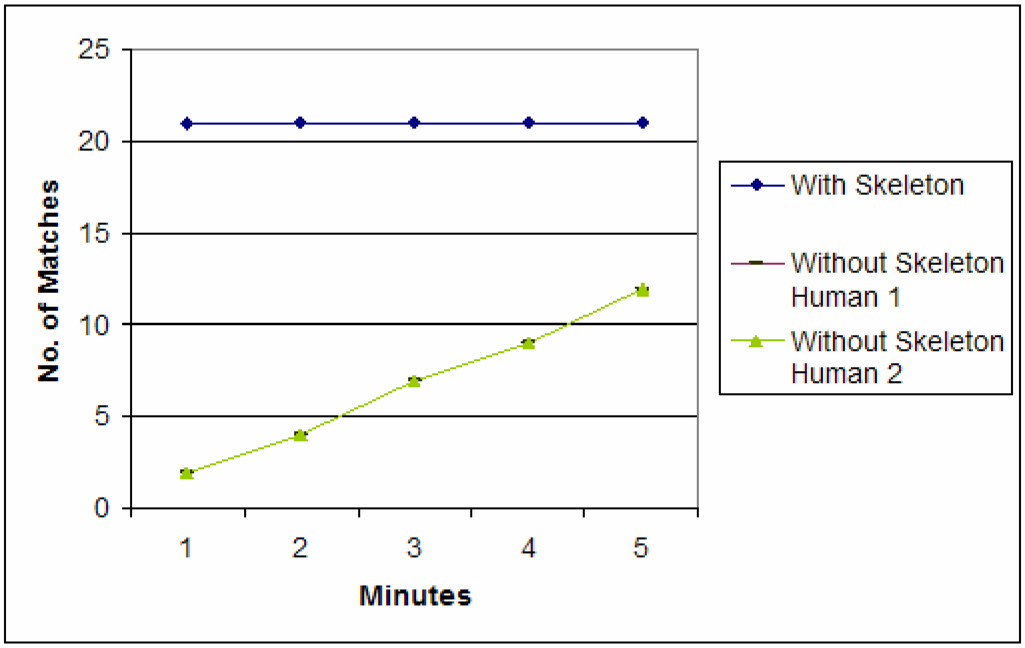

Figure 4.16. The time that is required for a human to find the matching classes vs. the time that is required to execute the class matching algorithm and create a skeleton.

4.4.G. Broader Interpretation

What is our class matching algorithm good at: Our experimental results show that the lexical, extensional, extensional closure, and global path similarity metrics help in finding the matching classes in different ontologies. The lexical similarity metric is quite robust and works reasonably well in a variety of domains.

The extensional similarity metric is also effective, as it can provide some clues about the similarity of classes, even when the names of the classes are different.

What are the limitations: One limitation of the lexical similarity metric is when the ontologies are in different languages, e.g. French and English. In such scenarios, the matches can not be found using the lexical similarity metric.

One limitation of the extensional similarity metric is that the instances need to be distributed uniformly between the two ontologies, in order for this metric to work well. Also, the instances themselves need to come from the same source or be matched with each other. In real world ontologies, it is often the case that the instances are not from the same source, and this causes difficulties in class matching.

The threshold of our class matching algorithm depends on the specific ontology. A reasonable threshold like 0.9, which performs well on some ontologies, may not work well on another ontology.

What is our skeleton good at: Our experiments for the skeleton show that a user is able to answer queries faster with the matcher and skeleton than without the matcher and skeleton. Furthermore, unlike any other approach, the skeleton is compliant with the RDF W3C recommendation and allows query answering and query expansion using standard tools.

One limitation of the skeleton is that it needs to be created by an administrator, before it can be used by end users for searching. This is not always desirable as the user might want to search across ontologies that have not been matched and no skeleton has been created by an administrator.

4.5. Summary

In this chapter, we designed a class matching algorithm, which utilizes various similarity metrics. We evaluated the algorithm on a variety of ontologies. For representation, we developed a novel W3C-compliant structure, named skeleton. An algorithm for creating the skeleton, for interoperability between ontologies, was presented. Our analysis shows that there is sufficient evidence to indicate that the F1 of our MATCH algorithm is larger than the F1 of S-Match and OMViaUO ($p<=0.05$), and this is not due to chance. No conclusion can be made with regard to the other four tools. The MATCH algorithm saves not only time but has a much greater F1 than two human trials, on the university ontology pair. There seems to be a possible correlation between ontology topic and optimal threshold. These items would be good for future study.

Chapter 5: RELATED WORK

In this chapter we will review the works that are relevant to the issues studied in this dissertation. We also discuss in detail how our work advances the state of the art. In Section 5.1, we review the work on dialogue agents as it is necessary for creating an information assimilator (refer to Chapter 2). Section 5.2 discusses the work on semantic search and user intention (refer to Section 2.6). The next four sections are related to our work in Chapters 3 and 4. Section 5.3 discusses the work on Linked Data. Section 5.4 covers the work on ontology integration and development. Section 5.5 reviews the work schema matching and information integration. Finally, Section 5.6 discusses the work on duplicate elimination and data cleaning. For each section, we cite the most relevant work to our research and also provide pointers to recent surveys.

5.1. Dialogue Agent

Some of the research on agent dialog and argumentation frameworks for dialog are focused on the communication between software agents and are restricted to predefined logic-based protocols used by software agents (e.g. [Par03, Rah03, Amg06, Bla07]). That line of work does not assume/require any semantic representation for the relationship between concepts, in the form of an ontology. In our work in Chapter 2, since the communication is between a human user and some device, it is necessary to utilize ontologies in the dialog agent, to capture the semantics of the concepts that humans employ in their utterance. Another related line of work is on programs that are capable of carrying on a limited form of conversation with a human, also referred to as "Turing test" programs. Eliza is an early example of such work [Wei66]. These systems do not perform any tasks and are only good at "running-on," i.e. they keep a superficial resemblance of conversation going, without achieving anything, unlike our work. There are a few application-oriented dialog systems that interact with humans, e.g. [Bob77, All95, Wal01], but they are often tied to a specific domain. [Bob77] is for booking and planning airline flights. Usually, these systems are frame based, i.e. the grammatical rules of the system are hard coded, and they do not use a domain-independent ontology for representing the semantics of concepts, as in our work.

Some of the work related to our study of ontology mapping for agent communication is as follows. Knowledge Query and Manipulation Language (KQML) is a language and protocol for exchanging information and knowledge [Fin97]. KQML is both a message format and a message-handling protocol, to support run-time knowledge sharing among agents. KQML can be used as a language for an application program to interact with an intelligent system or for two intelligent systems to share knowledge in support of cooperative problem solving. [Fan04] directly studies the problem of human-agent communication, where the agent is a knowledge-based question answering system (which takes a very similar role to our dialog agent). Their system uses the content of the knowledge base to automatically align a user's encoding of a query to the structure of the knowledge base.

The work on agent dialog usually considers communication between software agents only, and there is no human involved in the communication, unlike our work. Hence, the agent does not deal with ontologies and uses a logic-based protocol, instead. That line of work on agent dialog does not assume a semantic representation of concepts, in the form of an ontology. For example, [Par03] studies argumentation-based dialogs between software agents and examines how the outcome of the dialog is determined. [Rah03] looks at the factors that affect the negotiation strategy and move selection, in dialog among software agents, independent of the dialog protocol being used by the agents. [Amg06] provides a model for selecting the best move, at a given step in the dialog. [Bla07] provides a protocol and strategy specifically for inquiry dialogs. [Lae07] proposes the use of a logic based argumentation framework for software agents to agree or disagree about the correspondences between ontologies, based on agent's preferences. The focus of [Lae07] is on the argumentation protocol (i.e. the available moves), and the details of computing the concept similarity is not studied at all. It also does not involve humans.

There are some related efforts in agent-based systems literature that bear on similar problems, as the information assimilator. [Jos05] is an automated agent mediator (task-manager) that can pass along human requests for actions to multiple disparate task-performing engines. This work is closely related, but is somewhat in the reverse direction of the assimilator. The main flow of information in [Jos05] is from human to external entities. In the assimilator problem, however, it is primarily from external entities (data sources) to the human. Nonetheless, in each case there is a mediator in the middle, making sure the right things happen.

The work in the area of machine translation (MT) also provides a mediator of sorts, but between distinct languages. In general, rule-based methods for machine translation attempt to parse a text of the source language, to create a symbolic representation (Interlingua). Then, the text of the target language is generated from the symbolic representation [Nir02, Dor94]. The Interlingua is similar to a mediator and is a language-independent representation of the meaning of the text.

The DeepQA project is a recent research effort at IBM that is relevant to TAIA [Dee]. It shapes a grand challenge in Computer Science that aims to illustrate how the wide and growing accessibility of natural language content and the integration and advancement of Natural Language Processing, Information Retrieval, Machine Learning, Knowledge Representation and Reasoning, and massively parallel computation can drive open-domain automatic Question Answering technology to a point where it clearly and consistently rivals the best human performance. A first stop along the way is making a formidable Jeopardy contestant named Watson. Jeoperdy is a game in which humans compete against each other to answer a series of questions correctly and quickly.

5.2. Semantic Search

Swoogle is one of the pioneering search engines for finding the relevant ontologies and documents that are published on the Web [Din04]. It is hosted by the University of Maryland, Baltimore County and provides search capabilities for human users through a Web browser interface. The Swoogle system finds the relevant *ontologies* that contain the search phrases that are provided by users. In Section 2.6, we studied how with semantic search users can interactively find the relevant *entities* that are in different ontologies on the Web. Then we compared semantic search in RDF to keyword search in the relational model.

Falcons object search is a system that provides a great and intuitive search interface for finding linked objects on the Web of Data [Che09]. Watson is another search engine for the Semantic Web, which provides an efficient access point to online ontologies and semantic data [dAq08]. It performs the following tasks: (1) it collects the available semantic content on the Web, (2) analyzes it to extract useful metadata and indexes, and (3) implements efficient query facilities to access the data. Semantic Web Search Engine (SWSE) is another search engine, which provides a Web interface for searching the Semantic Web [Har06]. Although these search engines provide great functionalities, they are difficult to use for ordinary users. It is also generally difficult to find intended entities and relevant information about those entities using existing search engines for the Web of Data at this time.

Tabulator is a nice generic data browser and editor for the Semantic Web. It provides a way to browse RDF data on the Web, using outline and table modes [Ber06]. It can also accept input in RDF/XML, Turtle, or N-Triple format. [Har10] describes another system for searching and navigating large amounts of Web data. This system provides detail, list, and table views for arbitrary types of objects. Sindice Inspector is a tool for extracting structured data (such as microformats or RDFa) contained in HTML documents [sin].

[Jag07] envisions an exciting avenue of research in relational databases. They state that while relational databases have good performance and functionalities, databases are very difficult to use for typical users. They identify five pain points, including: too many joins, too many options, lack of explanation, no direct manipulation, and difficulty of defining structure for data. To tackle these usability issues, they propose a presentation data model and recommend direct data manipulation with a schema later approach. It is interesting that semantic search, as described in Section 2.6.2, in fact strives to also achieve direct manipulation of data in a way that the user does not need to know about the schema of the data. Hence, in some sense semantic search is similar to the proposed presentation data model, and this provides a variety of research opportunities.

Paul Grice in his seminal paper, “Meaning,” first published in 1957, analyzed the issue of speaker’s meaning [Gri57]. This work is relevant to our discussions regarding user intentions in Section 2.6. Grice drew a distinction between what he called natural meaning and what he called non-natural meaning. This is an account of his distinction, provided in [Gri]. Natural meaning is the kind of meaning that we are speaking of when we say something like, “Those spots mean measles” or “A shiny coat in a dog means health”. Non-natural meaning is the kind of meaning we speak of when we say “Those three rings on the bell (of the bus) mean that the bus is full” or “By saying that the child looked guilty, he meant that the child was in fact guilty”.

Further, Grice offered a three-part analysis of non-natural meaning: *A* (an agent) meant something (non-naturally) by *x* (an utterance or gesture), if and only if *A* intended the utterance or gesture *x* to produce some effect in an audience by means of the recognition of this intention. In other work, Grice contemplated a variety of refinements. The preliminary analysis that he offers in “Utterer’s Meaning, Sentence Meaning, and Word-Meaning” (1968) for what he calls the occasion-meaning of indicative-type utterances may be represented as follows [Gri68]:

By uttering *x*, *U* meant that *p* if and only if for some audience *A*, *U* uttered *x* intending (i) that *A* should believe that *U* believes that *p*, (ii) that *A* should believe that *U* intended (i), and (iii) that (i) should be achieved by means of achieving (ii).

Speaker’s (or utterer’s) meaning, so defined, has to be strictly distinguished from what might be called the conventional meaning of a speaker’s words. The place of conventional meaning in Grice’s conception of language appears to be that it constitutes a feature of words that speaker’s might exploit in realizing the intentions referred to in the analysis of speaker’s meaning. This emerges in “Utterer’s Meaning and Intentions” (1969), where Grice considers a variety of purported counter examples to his analysis of speaker’s meaning and as a result produces a much more complex analysis [Gri69]. Particularly important is the conclusion that when an utterer means that *p* by utterance *x* the utterer must suppose that *x* has some feature *f* that the audience is to think of as correlated in a certain way with the response that the utterer intends to produce in the audience. In view of the account of so-called timeless meaning in “Utterer’s Meaning, Sentence Meaning, and Word-Meaning”, Grice’s thought would appear to be that this feature is often the timeless meaning of the utterance.

5.3. Linked Data

[Biz09] and [Hea11] provide nice overviews of the core principles necessary for publishing Linked Data on the Web. As outlined in [Biz09], one of the major research challenges of Linked Data is to address the issue of schema mapping and data fusion, i.e. to retrieve data from different distributed sources of information and present it to the user. This requires a mapping of terms from different ontologies (vocabularies). In Chapter 3, we illustrated this issue with a precise use case, in which there were two independent universities. We demonstrated how users can query different ontologies (distributed and autonomous sources of information) and retrieve data from all of them, across organizational boundaries.

[Mul10] describes an approach for finding associations between data elements that are in a relational model and nodes in a reference Linked Data collection (e.g. DBpedia). In other words, it assigns table columns to classes, table cells to entities, and inferred relations between columns to properties. The resulting interpretation can then be used to annotate relational tables. In general, the need for sharing data with collaborators motivates custodians and users of relational databases (RDB) to expose relational data on the Web of Data. [Pru10] examines a set of use cases from science and industry for taking relational data and exposing it in RDF. [Das10] describes R2RML, a language for expressing customized mappings from relational databases to

RDF datasets. Such mappings provide the ability to view existing relational data in the RDF data model.

[Mil10] introduces the core challenges faced when consuming multiple sources of Linked Data and focuses on the problem of querying. They compare both URI resolution and federated query approaches and outline the experiences gained in the development of the RKB Platform [Gla09]. Virtuoso Universal Server is an open source middleware and database engine hybrid that combines the functionality of a traditional RDBMS, ORDBMS, virtual database, RDF, XML, free-text, web application server and file server in a single system [Ope]. Virtuoso enables a single multithreaded server process to implement multiple protocols and can be useful, when processing Linked Data and dealing with other kinds of data as well. Also, Oracle is a commercial database product that supports RDF functionalities.

5.4. Ontology Integration (Development)

Some works on ontology mapping (e.g. [Kal03a, Doa03, Ehr04]) focus on finding correspondences between ontologies (i.e. ontology matching), and they address Step 1 of ontology mapping (refer to Section 3.2). Some other works on ontology mapping (e.g. [McG00, Noy00, Stu01]) produce a merged ontology as the final output (representation), which is generally in the context of ontology development (refer to Sections 3.3.1). Our approach to facilitating interoperability between ontologies (refer to Sections 3.3.2) is different from ontology merging solutions, since merging inherently follows a different goal (for the distinction, refer to Sections 3.3.3 and 3.3.4). In our approach, for facilitating interoperability, we find the class correspondences between the ontologies and then create a skeleton to represent these correspondences. In Chapter 4 we provided the required *algorithms* for this approach, with attention to the context of the Semantic Web, in a fashion that is compliant with the W3C recommendations.

The ontology matching problem has been studied extensively, e.g. [McG00, Doa03, Euz07]. [McG00] tackles ontology merging for government intelligence in DARPA's High Performance Knowledge Base program. [McG00] deals with knowledge intensive applications, where ontologies are developed by various teams of people with broad ranges of training. The people are responsible for the development, design and maintenance of ontologies. These ontologies need to be integrated into other large application ontologies. Sometimes, the integration is done by people who do not have much training in knowledge representation. Hence, the integration process requires tools that support users in: (1) merging of ontological terms from varied sources, (2) diagnosis of coverage and correctness of ontologies, and (3) maintenance of ontologies over time. [McG00] created a tool called Chimaera for the above tasks.

[Noy00] developed the PROMPT tool for merging ontologies, and the work was also motivated by DARPA's High Performance Knowledge Base program. The goal was to develop ontologies, through reusing existing ontologies. It exploits the graph structure of ontologies to provide suggestions for merging. Merging of ontologies causes inconsistencies and the user is interactively prompted with suggestions to remedy these inconsistencies. The issue of inconsistencies that arise from merging is also highlighted by [Pot03]. The ontology merging process is inherently semi-automated, and the user is actively kept in the loop, as merging is being performed. The user makes the final subjective decisions, by considering the logical entailments, while the system provides helpful suggestions for merging.

Both [McG00, Noy00] provide concrete ontology merging scenarios. It is crucial to note that these works are in the context of ontology development (refer to Section 3.3.1). In both papers, there is no mention of interoperability (i.e. information integration) - refer to Section 3.3.3 and 3.3.4 for the distinction.

[Stu01] uses a set of shared instances or a set of shared documents that are annotated with the concepts of two ontologies. Then, a lattice is generated to merge and relate the classes of the ontologies, using formal concept analysis. Various systems have studied finding lexical matches between ontology entities, or use dictionaries, WordNet and other resources for the matching process. [Kal03a] is a system that also uses formal concept analysis to find the matching concepts of two ontologies by matching them to a third reference ontology. [Aum05] presents the COMA++ tool, which provides a comprehensive and extensible library of individual matchers. The matchers can be selected to perform the matching operation. The graphical interface of this tool offers a variety of interactions and allows the user to influence the matching process. [Kal03b] provides a survey of various ontology mapping systems. [Ont] is a site that provides a comprehensive list of publications related to the ontology mapping problem.

GLUE is one of the earlier systems that utilizes machine learning for ontology matching [Doa03]. It exploits a Naïve Bayes (NB) classifier to detect the similarity of classes in two ontologies. To find the similarity between class *S* in one ontology and class *T* in another ontology, a NB classifier is trained offline on the instances of *S*, and then tested on the instances of *T*. Then, the Jaccard coefficient of classes *S* and *T* is computed, as a measure of the similarity of the classes. Also, the system utilizes the full names of classes for measuring their similarity. The names are created by starting from the root of the taxonomy and moving down the class hierarchy. This system requires training data, unlike our class matching algorithm. Also, in GLUE the instances are assumed to be long textual strings, which is not generally the case in ontologies. We do not make such an assumption about instances.

[Noy04] is a brief survey that highlights some important issues in ontology mapping. [Ehr04] proposes various features for different types of entities, when matching between ontologies. They also provide a process to compute the similarity measures of those features and to aggregate them. The issue of ontology design (also known as ontology development/engineering) has been studied by [Noy97, Noy01, McG03, Gol03, Fen01, Gua02]. Currently there exists no standard, W3C-compliant approach for representing the matching classes between ontologies, as is necessary for facilitating interoperability [van08, Mil10].

As described in Section 3.2, in Step 1 of ontology mapping, the output of an ontology matching algorithm is a set of matching entities between two ontologies. [Euz04] proposes that this set can be specified in the form of a list in a file. A specific format for such a file is proposed, and an API (Application Programming Interface), which can process this format, is implemented in Java. This API can be used by Java programmers to manipulate the matching entities, when the matches are specified in such a format. Since this format is not a part of the W3C recommendations, standard ontology tools can not use it without extra implementation effort. However, our skeleton (refer to Section 4.2) is compliant with W3C recommendations.

[Bou04] proposes an extension to the syntax, and the semantics of OWL, named Context OWL (COWL). The extensions allow the localization of ontology's content (making it not visible to other ontologies), and allow bridge rules for controlled forms of global visibility. These extensions are not a part of the OWL W3C recommendation, and as a result, they are not implemented in standard ontology reasoners. In that work, there is also no algorithm for creating these rules. [Bou04] states that this is a first step, and a lot of research remains to be done, to address the core issue at stake, which is: the tension between how much should be shared and globalized (via ontologies), and how much should be localized with limited and totally controlled forms of globalization (via contexts).

In 2004, the World Wide Web Consortium (W3C) announced the final approval of two key Semantic Web technologies, the revised Resource Description Framework (RDF) and the Web Ontology Language (OWL). RDF and OWL are Semantic Web standards that provide a framework for asset management, enterprise integration and the sharing and reuse of data on the Web. These standard formats for data sharing span application, enterprise and community boundaries, and allow all of these different types of users to share the same information, even if they don't share the same software.

The RDF Primer is an introduction to, and tutorial on how to use, RDF and RDF Schema [RDFp]. RDF Vocabulary Description Language describes how to use RDF to describe application and domain specific vocabularies [RDFv]. [OWLu] presents the use cases and requirements that motivated OWL. [OWLo] is an overview document, which briefly describes the features of OWL and how they can be used. [OWLg] is a comprehensive guide that walks through the features of OWL with many examples of the use of OWL features.

[van08] provides an interesting overview of the semantic interoperability problem and reviews some of the core issues involved. van Harmelen states that with the rapid growth of the Internet and the Web, more principled mechanisms to facilitate semantic interoperability (i.e. facilitate querying of data) across organizational boundaries have become necessary. He emphasizes that despite many years of work on the semantic interoperability problem, this old problem is still *open* and has acquired a new urgency, now that physical and syntactic interoperability barriers have largely been removed. Physical interoperability between systems has been solved with the advent of hardware standards, such as Ethernet, and with protocols, such as TCP/IP and HTTP. Also, syntactic interoperability between systems has been largely solved by agreeing on the syntactic form of the data that we exchange, particularly with the advent of eXtendible Markup Language (XML). For semantic interoperability between systems, we not only need to know the syntactic form (structure) of the data, but also the intended meaning of the data. Note that the skeleton addresses the semantic interoperability problem, i.e. the skeleton enables users to query the data across organizational boundaries.

[Isa09] is an interesting and recent work on ontology matching (thesaurus alignments). [Isa09] explores common real-world problems in a library domain and identifies solutions that would greatly benefit from a more application-embedded study, development and evaluation of matching technology. Knowledge-based data integration can potentially provide great benefits in various applications. These benefits are being investigated in real-world scenarios by [Bod06, Bec07, Ala08, Isa09]. We discussed the issue of instance matching in Section 3.8. In OWL, the owl:sameAs construct is used to specify that two instances in two ontologies are the same. [Hal10] outlines four alternative readings of owl:sameAs, showing with examples how it is being (ab)used on the Web of Data. Then, they present possible solutions to this problem by introducing alternative identity links that rely on named graphs.

In summary, interoperability between autonomous information sources is at the core of the Semantic Web vision [Ber01, Sha06, Biz09]. In essence, the Semantic Web is about applying the advances in the knowledge representation domain (developed by the AI community, e.g. [Sha91, Gru93, Hen95, McG03]), to the exciting and industrial-scale information integration applications (which is the Holy Grail of the database community [Doa01, Hal05]). All this is embedded in the infrastructure that has flourished to form the current World Wide Web.

5.5. Schema Matching and Information Integration

The work on schema matching is related to ontology mapping for interoperability, as outlined in Section 3.5. The need for communication between autonomous and distributed information systems is increasing with the wide usage of the Web [Hal00]. Nowadays, data sharing across resources and enterprises is no longer a desirable feature, but a necessity. Considerable amount of research on data integration and schema mapping over the last three decades have lead to improvements in this area [Ber81, She90, Hal05].

[Rah01] provides a survey of various schema matching approaches in the database literature. [Kno02] is a comprehensive tutorial that motivates the need for information integration on the Web, which falls between the database and AI areas. [Kno02, Hal00] emphasize the roles that AI technology can play in supporting information integration. [Hal05] provides a collection of discussions on the topic from various perspectives.

[Doa01] proposes to learn the mappings between a source schema and a mediated global schema, in information integration systems. Creating these mappings by human users for information integration is time consuming. Their system learns the mappings from a few initial mappings, provided by users. Different learning mechanisms are utilized on the source schemas or on their data, and the results are combined using a meta-learner. [Pot03] describes an algorithm for the merging of models (schemas or ontologies), when a set of correspondences is given. It also provides an elaborate treatment of conflict resolution, since the merging of models generates various types of conflicts. [Len02] is a detailed treatment of the information integration problem from a theoretical standpoint.

[Mel02] introduces similarity flooding, a generic graph matching algorithm, which computes the correspondences of nodes in graphs and applies it to schema matching. Graph-based approaches have also been utilized in [Mil98] for schema matching. [An05] provides an algorithm that creates data transformations from a source relational database to a target ontology, using a given set of simple correspondences. Our work is focused on facilitating interoperability between ontologies, and it does not involve any transformation from databases to ontologies. [Li00] presents a tool (SEMINT) that uses neural networks to find correspondences between attributes in heterogeneous databases. It uses both schema and data to produce matching rules automatically.

[Hal06] introduces a set of principles for future data integration systems. Unlike traditional integration systems, in their pay-as-you-go model of integration, they assume that full semantic mappings between schemas are not available. However, the mappings are created incrementally and based on user needs. They advocate the opportunities for learning from existing mappings and also learning from human attention. [Ber07] proposes the model management vision, as a generic approach to solving the problems of data programmability, where precisely engineered mappings are required. The goal is to develop an engine that supports operations to match schemas, compose mappings, diff schemas, merge schemas, translate schemas into different data models and generate data transformations from mappings.

5.6. Duplicate Elimination and Data Cleaning in Databases

The work on duplicate elimination in databases is closely related to instance matching in ontologies, as discussed in Section 3.8. The same real-world entity, which is represented as a record, usually has different representations in different databases. Records (tuples) in a database consist of various attributes (fields). Duplicate records refer to the same entity, but do not share a common key, or they contain errors. This makes the matching of duplicate records a difficult task.

Differences between duplicates could arise from incomplete information, lack of standard formats or any combination of these factors. Many different attribute similarity metrics and record similarity metrics have been proposed for duplicate elimination and data cleaning in databases. Approximate duplicate elimination has been studied for the last five decades and was initially of interest to the statistics community [New59]. The problem is also known as *record linkage* and *entity resolution*. [Elm07] is a recent survey of the literature on detecting duplicate records.

One of the early works that mention the analogy between the issues of instance matching in ontologies and duplicate elimination in databases is [Hai06]. Various data cleaning approaches, studied in the database literature, can be useful for instance matching in ontologies. As described in Section 3.8, instance matching is required, when ontologies are merged. It is also necessary in class matching for interoperability, when computing the extensional similarity metrics. [Ala02] proposes a stepwise process for detecting unique instances, which is to be used for merging, populating and maintaining ontologies. They refer to the problem as identifying coreference or referential integrity.

[Her98] proposes an effective rule-based approach to finding duplicates, where the user specifies a set of rules, using various string similarity metrics and thresholds. The rules are coded in some programming language, and it is time consuming to repeatedly create these rules for each data schema that needs to be cleaned. These rules are static and can not be tuned using machine learning algorithms. [Bil03] uses adaptive similarity metrics to learn the parameters for string comparison. Hence, when training data (labeled duplicate records) are available, the system can be trained. However, without training data, such adaptive systems are ineffective, as their performance is tied to the existence of training data. Also, the training data needs to have a good coverage of problem states, and such data can be difficult to obtain.

Chapter 6: CONCLUSIONS AND FUTURE DIRECTIONS

6.1. Conclusions

There is tremendous potential in enabling general users and scientists to search, browse, and discover information from independent knowledge bases (Linked Data sources) on the Web, in ways that were not anticipated by the developers of the knowledge bases, at the time when the knowledge bases were designed. We demonstrated the differences between searching for data in the RDF model and relational model in terms of user interaction and behavior. In this dissertation, we laid out the foundations of semantic search in Linked Data and provided effective mechanisms to facilitate data interoperability across organizational boundaries.

The key contributions of this dissertation are as follows:

- An overarching definition of ontology mapping that allows the systematic analysis and distinction of different goals of ontology mapping. The analysis of the goals provides a guideline for performing ontology mapping and influences the design of tools and algorithms for ontology mapping.
- Facilitating interoperability between ontologies is rigorously compared with information integration in databases. Based on this comparison, class

matching is emphasized as a critical part of facilitating interoperability. Then, various class similarity metrics for class matching are formalized and evaluated on real-world ontologies.

- A novel W3C-compliant representation, named skeleton, is developed and evaluated against other existing approaches. The skeleton encodes the correspondences between ontologies and facilitates interoperability between them. An algorithm for creating the skeleton is provided.

With semantic search in RDF, user's intentions can be accurately narrowed down in a more robust way than keyword search in the relational model. The comparison of semantic search and keyword search clarified the advantages of using RDF instead of the traditional database. We demonstrated that it is difficult to retrofit a robust semantic search *behavior* on the relational data model and find answers to questions about an entity (as available in RDF). In semantic search, typical users (who have not had any special training) can *interactively* explore the data and navigate through sets of entities, by utilizing their knowledge of a domain. Also, they do not need to use unintuitive and complex SQL statements, as in the relational model.

We demonstrated why it is difficult to simulate semantic search with keyword search on the relational data model. In other words, the explicit encoding of semantics (via typed relations) and the global referencing of entities in RDF (via links or URIs) are the two critical enabling *features* that make RDF suitable for robust search and information integration across different enterprises. The comparison of semantic search and keyword search also revealed some of the crucial research challenges that need to be addressed for scalable semantic search.

Although the ontology mapping process for the Semantic Web has been studied in the past decade, the underlying principles that drive this process have remained murky [van08, Mil10]. We provided an overarching definition of ontology mapping and distinguished its goals using precise use cases. The ontology mapping procedure for two separate and autonomous ontologies, O_1 and O_2, consists of the following steps: *Finding* corresponding entities in ontologies O_1 and O_2, and then *representing* the found correspondences and *using* it to achieve some goal. Hence, the goal of ontology mapping determines what candidates to consider, when we are finding the correspondences. The goal also determines how to represent the correspondences.

One possible goal of ontology mapping is ontology development; that is when an ontology is being designed or engineered by an organization. For example, consider that a supermarket is using an ontology to categorize its items. After a while, the supermarket may expand its business and offer new products. The current ontology then needs to be changed and expanded based on the changes in the business model. In this ontology development process, existing ontologies can be merged into the current ontology and used for creating a new ontology.

The other possible goal of ontology mapping is facilitating interoperability; that is when there are various parties, which are using different ontologies, and users need a mechanism to be able to query the information, which resides in the ontologies. For example, consider that two different universities are using ontologies to categorize their courses. Users may then want to find all the courses that are related to Computer Science in both universities.

We studied these two goals of ontology mapping across seven dimensions, namely: representation of mappings, inconsistency between ontologies, automation of the mapping process, importance of isolating the organizations, class matching and its role in interoperability, tractability of skeleton creation, and the tight coupling between the interoperability goal and the Semantic Web vision. The dimensions serve as a guideline for performing ontology mapping, and they influence the design of tools and algorithms for ontology mapping, e.g., what tasks can be automated and what tasks should involve a human in the loop, what candidates to consider for finding correspondences, and how to represent the correspondences.

In order to understand the goal of facilitating interoperability between ontologies more thoroughly, we compared it with the information integration problem in databases. We showed that in databases, the final output of the information integration process is a mapping between columns in different local schemas. We illustrated that columns in the relational model correspond to classes in the RDF model. The comparison also revealed that to facilitate interoperability between ontologies, the classes in the ontologies need to be mapped to each other.

The ultimate objective of interoperability is to query and correctly retrieve data instances across various ontologies. The data resides in the classes in the ontologies. Hence, as long as a correct mapping between the classes in the ontologies exists, users can query and correctly retrieve the data instances across various ontologies.

After identifying that class matching is a critical part of ontology mapping for facilitating interoperability, we formally defined a set of class similarity metrics that utilized different clues to compute the similarity of classes. Then, we presented an algorithm for class matching, which directly measures the similarity of classes rather than determining subclass relationships.

The lexical similarity metric measures the similarity of the strings that represent the name of the classes. The extensional similarity metric considers the direct instances of two classes that are in common between the two classes. The extensional closure similarity metric considers not only the direct instances that are in common, but also the instances of the subclasses that are being compared. The global path similarity metric provides clues about the position of classes in the class hierarchy. We then experimentally evaluated the effectiveness of these metrics on real-world ontologies.

In order to achieve interoperability between ontologies, the class correspondences need to be represented in a suitable form. We provided a novel W3C-compliant representation, named *skeleton*, which enables users to search and discover knowledge from different ontologies, more generally and effectively than existing alternatives. More specifically, the skeleton provides a uniform representation of mappings across ontologies and is stored independent of those ontologies. A user's query, in a Linked Data browser, may then be largely guided and managed via the uniform skeleton representation, freeing the user from the burdensome and time-consuming task of mapping during the querying process. We also evaluated the skeleton experimentally and provided an algorithm for creating the skeleton.

Our design for the skeleton representation is compliant with the W3C recommendations. In other words, when using the skeleton, query answering and query expansion can be performed using standard tools, without any ontology reasoner adjustments and extra implementation effort.

Creating a skeleton isolates the original ontologies from changes, and therefore each autonomous organization uses its own business model for everyday operations. This isolation, in turn, eliminates the possibility of the inconsistencies that arise, when ontologies are merged. Moreover, once the class matching is done and the correspondences have been determined, the process of creating the skeleton is fully automated and performed using our algorithm without any human user involvement.

Our analysis shows that there is sufficient evidence to indicate that the F1 of our MATCH algorithm is larger than the F1 of S-Match and OMViaUO ($p<=0.05$), and this is not due to chance. No conclusion can be made with regard to the other four tools. The MATCH algorithm saves not only time but has a much greater F1 than two human trials, on the university ontology pair. There seems to be a possible correlation between ontology topic and optimal threshold. These items would be good for future study.

6.2. Future Directions

This dissertation has made significant inroads into understanding semantic search and facilitating interoperability across organizational boundaries. In this section, we discuss some of the critical challenges that remain for further research toward widely deploying semantic search on the Web.

6.2.1. Effective User Interfaces

In general semantic search interfaces should enable users to search and browse the data without knowledge about the schema of the data. There are a number of issues that need to be considered for creating a robust user interface for semantic search. Users are often more familiar with traditional Web search engines than semantic search engines, for example some users find the use of classes, class hierarchy, and instances to be confusing. The user interface should be designed such that it can be used easily by general users, when looking for answers to everyday questions, as well as scientists, when browsing domain-specific knowledge bases.

Given the diversity of entities that need to be browsed on the Web of Data, the interface should support various types of data visualizations. The interface should also have minimal configuration settings in order for it to be appealing and usable, similar to the simple interface of current Web search engines for the Web of documents. Since the volume of data that is being browsed can be large, scalability issues such as incremental loading of data need to be considered. Semantic search interfaces should also allow logging of user interactions and exporting of results, as the results may need to be used by other applications.

Some other points that should be considered are effective presentation of sets of entities, user interactions to support the selection of some of the entities from the result of a query, presentation of relationships (at both class and instance level), and intuitive ways of specifying SPARQL queries.

6.2.2. Ranking of Results and Data Quality

When searching for objects on the Web of Data, the results come from various data sources (namespaces). One of the challenging issues is the ranking of the resulting objects. Ranking of objects in this context not only depends on the relevance of the object to the query, but also the quality of the data source that provides the object. A number of criteria can be considered to evaluate the quality of data in a Linked Data source, for example consistency, freshness in regard to time, comprehensibility in terms of documentation and commenting, validity of URI's, amount of data, licensing, and performance of the data store. Quantifying these factors and integrating them into the ranking of objects require further research.

A similar problem to the ranking of objects occurs, when searching the Web of documents, where the most reliable documents need to be returned. With current technology, in Google the most popular Web pages are essentially returned as top results. This approach may be adapted for the ranking of objects in the Web of Data, but when searching for critical scientific data (e.g. in health care domain), other approaches may need to be used, in order to ensure the retrieval of accurate results from authoritative sources regardless of the popularity of the data source.

6.2.3. Dealing with Distributed and Dynamic Data

At the moment most applications that use Linked Data are limited to a pre-specified set of datasets, i.e. they do not use all possible data sources on the Web. Various challenges arise in developing applications that consume data from the unbounded Web, in regard to querying and accessing an open number of data sources. These challenges are largely unresolved at this time. One challenge is related to the fact that entities (instances) that result from a query come from distributed sources. Hence, the same entity can have multiple URIs (co-references). This leads to problems in the aggregation of results in a SPARQL query. In other words, such entities need to be reconciled and integrated.

Another challenge is in resource discovery, when we need to discover the subjects of an object in a SPARQL query. This requires the traversal of a relation in reverse direction, which is not easily possible. By analogy, resource discovery is similar to finding all the documents that link to a specific document on the Web of documents. As the Web of Data becomes more interlinked, answering some queries requires accessing multiple datasets. Such queries that span multiple datasets can not be executed readily and pose a challenge. These queries can be answered using URI resolution or SPARQL, and each approach has its benefits and pitfalls.

Since data sources on the Web are often dynamic and autonomous, the data and schema can change. Therefore it is necessary to develop techniques to monitor the changes and maintain the correct correspondences. This problem is often referred to as mapping maintenance, and while it is important in practice, it has not been fully addressed. Also, some of the Linked Data on the Web originates from existing databases and Web pages. Hence such Linked Data is not updated frequently, which causes problems in different applications. In other words, dealing with the dynamic nature of data on the Web requires further research.

6.2.4. Scalability of Storage and Retrieval

Storage and retrieval of data in the relational database model has become highly optimized over the last three decades. Similar performance levels are necessary for RDF to enable large-scale semantic search. Considering that the representation of data in the RDF and relational models are different, there are various storage and indexing issues that need to be studied for achieving better performance. These issues include: native storage mechanisms for RDF or efficient storage of RDF in the relational model, indexing and retrieval of RDF data, and optimization of queries specified in SPARQL.

Research efforts along this line are underway, e.g., [Aba07] studies the physical design issues and proposes the vertical partitioning of data in the relational model, for RDF storage. [Neu09] focuses on join processing, since the fine-grained and schema-relaxed use of RDF often entails star-shaped and chain-shaped join queries with many input streams from index scans.

Bibliography

[Aba07] Abadi, D., Marcus, A., Madden, S., Hollenbach, K., “Scalable Semantic Web Data Management Using Vertical Partitioning,” *Proc. of the 33rd International Conference on Very Large Data Bases (VLDB'07)*, Vienna, Austria, September 23-27, 2007.

[Ala02] Alani, H., Dasmahapatra, S., Gibbins, N., Glaser, H., Harris, S., Kalfoglou, Y., O'Hara, K., Shadbolt, N., “Managing Reference: Ensuring Referential Integrity of Ontologies for the Semantic Web,” *Proc. of the 13th Int. Conf. Knowledge Eng. and Knowledge Management (EKAW 02)*, Sigenza, Spain, pp. 317–334, 2002.

[Ala08] Alani, H., Chandler, P., Hall, W., O’Hara, K., Shadbolt, N., Szomszor, M., “Building a Pragmatic Semantic Web,” *IEEE Intelligent Systems*, 23(3), pp. 61-68, 2008.

[All95] Allen, J.F., Schubert , L.K., Ferguson, G., Heeman, P., Hwang, C., Kato, T., Light, M., Martin, N., Miller, B., Poesio, M., Traum, D., “The TRAINS Project: A case study in building a conversational planning agent,” *Journal of Experimental and Theoretical AI*, Vol. 7, pp. 7-48, 1995.

[Amg06] Amgoud, L., Hameurlain, N., “An argumentation-based approach for dialogue move selection,” *Proc. of 3rd Int. Workshop on Argumentation in Multi-Agent Systems (ArgMAS’06)*, pp. 111-125, 2006.

[An05] An, Y., Borgida, A., Mylopoulos, J., "Inferring Complex Semantic Mappings Between Relational Tables and Ontologies from Simple Correspondences." *OTM Conferences (2)*, 2005.

[An05] An, Y., Borgida, A., Mylopoulos, J., "Inferring Complex Semantic Mappings Between Relational Tables and Ontologies from Simple Correspondences." *OTM Conferences (2)*, 2005.

[And05] Anderson, M., Perlis, D., "Logic, Self-awareness and Self-Improvement: The Metacognitive Loop and the Problem of Brittleness," *Journal of Logic and Computation*, 15(1), 2005.

[And08] Anderson, M., Fults, S., Josyula, D., Oates, T., Perlis, D., Schmill, M., Wilson, S., Wright, D., "A Self-Help Guide For Autonomous Systems," *AI Magazine*, 2008.

[Aum05] Aumueller, D., Do, H.H., Massmann, S., Rahm, E., "Schema and ontology matching with COMA++," *Proceedings of the 24th ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*, 2005.

[Bat86] Batini, C., Lenzerini, M., Navathe, S., "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys*, 18(4), pp. 323-364, 1986.

[Bec07] Beck, A.R., Fu, G., Cohn, A.G., Bennett, B., Stell, J.G., "A framework for utility data integration in the UK," *26th Urban Data Management Symposium (UDMS'07)*, Stuttgart, Germany, 2007.

[Ber00] P. Bernstein, A. Halevy, and R. Pottinger. A vision for management of complex models. ACM SIGMOD Record, 29(4):55–63, 2000.

[Ber01] Berners-Lee, T., Hendler, J., Lassila, O., “The Semantic Web,” *Scientific American*, May, 2001.

[Ber06] Berners-Lee, T., et al., “Tabulator: Exploring and Analyzing Linked Data on the Semantic Web,” *Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06)*, Athens, Georgia, November, 2006.

[Ber07] Bernstein, P., Melnik, S., “Model Management 2.0: Manipulating Richer Mappings,” *SIGMOD*, 2007.

[Ber09] Berners-Lee, T., TED 2009, http://www.ted.com/index.php/talks/tim_berners_lee_on_the_next_web.html.

[Ber81] Bernstein, P., Goodman, N., Wong, E., Reeve, C.L., Rothnie, J.B., “Query processing in a system for distributed databases (sdd-1),” ACM Transactions on Database Systems, 6(4), pp. 602-625, 1981.

[Bil03] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S., “Adaptive Name Matching in Information Integration,” *IEEE Intelligent Systems*, vol. 18, no. 5, pp. 16-23, 2003.

[Biz09] Bizer, C., Heath, T., Berners-Lee, T., “Linked Data - The Story So Far,” Int. J. Semantic Web Inf. Syst., 5(3), pp. 1-22, 2009.

[Bla07] Black, E., Hunter, A., “A Generative Inquiry Dialogue System,” *Proc. of 6th Int. Conference on Autonomous Agents and Mutli-Agent Systems (AAMAS'07)*, 2007.

[Bob77] Bobrow, D.G., Kaplan, R.M., Kay, M., Norman, D.A., Thompson, H., Winograd, T., "GUS: A frame driven dialog system," *Artificial Intelligence*, Vol. 8, pp. 155-173, 1977.

[Bod06] Bodenreider, B., Stevens, R., "Bio-ontologies: current trends and future directions," *Briefings in Bioinformatics*, 7(3), pp. 256-274, 2006.

[Bou04] Bouquet, P, Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H., "Contextualizing ontologies," Journal of Web Semantics, 1(4), pp. 325-343, 2004.

[Bro01] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling knowledge representation on the Web by extending RDF schema. In Proceedings of the Tenth Int. World Wide Web Conference, 2001.

[bus07] Q&A with Tim Berners-Lee, 2007, http://www.businessweek.com/technology/content/apr2007/tc20070409_961951.htm

[Che09] Gong Cheng and Yuzhong Qu. Searching linked objects with falcons: Approach, implementation and evaluation. International Journal on Semantic Web and Information Systems (IJSWIS), 5(3):49–70, 2009.

[dAq08] Mathieu d'Aquin, Enrico Motta, Marta Sabou, Sofia Angeletou, Laurian Gridinoc, Vanessa Lopez and Davide Guidi. Towards a New Generation of Semantic Web Applications. IEEE Intelligent Systems, Vol. 23, Issue 3, May/June 2008.

[Das10] R2RML: RDB to RDF Mapping Language, http://www.w3.org/TR/r2rml/

[Dee] http://www.research.ibm.com/deepqa/deepqa.shtml

[Din04] Ding, Li, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal C Doshi, and Joel Sachs (2004). "Swoogle: A Search and Metadata Engine for the Semantic Web". Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management (CIKM). ACM. pp. 652–659.

[Doa01] Doan, A., Domingos, P., Halevy, A., "Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach," *Proceedings of the 20th ACM SIGMOD International Conference on Management of Data (SIGMOD'01)*, 2001.

[Doa03] Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A., "Learning to Match Ontologies on the Semantic Web," *Very Large Databases Journal (VLDB Journal)*, Vol. 12, No. 4, 2003.

[Doa06] Doan, A., Ramakrishnan, R., Vaithyanathan, S., "Managing Information Extraction," *SIGMOD'06*, Tutorial, 2006.

[Dor94] Dorr, B., "Machine translation divergences: A formal description and proposed solution," *Computational Linguistics*, 20(4), pp. 597-633, 1994.

[Ehr04] Ehrig, M., Staab, S., "QOM: Quick Ontology Mapping," *ISWC*, pages 683–697, 2004.

[Elm07] Elmagarmid, A.K., Ipeirotis, P., Verykios, V., "Duplicate Record Detection: A Survey," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Vol. 19(1), pp. 1-16, 2007.

[Etz05] Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., Yates, A, “Unsupervised named-entity extraction from the Web: An experimental study,” *Artificial Intelligence*, 165(1), pp. 91-134, 2005.

[Euz04] Euzenat, J., “An API for Ontology Alignment,” International Semantic Web Conference, pp. 698-712, 2004.

[Euz07] Euzenat, J.,Shvaiko, P., Ontology matching, Springer-Verlag, 2007.

[Fan04] Fan, J., Porter, B., ”Interpreting Loosely Encoded Questions,” *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI’04)*, 2004.

[Fen01] Fensel, D., *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*, Springer-Verlag, 2001.

[Fin97] Finin, T., Labrou, Y., Mayfield, J., “KQML as an agent communication language,” in Jeff Bradshaw (Ed.), *Software Agents*, MIT Press, Cambridge, MA, 1997.

[FOAF] http://www.foaf-project.org/

[Gar97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. Journal of Intelligent Inf. Systems, 8(2), 1997.

[Giu04] Giunchiglia, F., Shvaiko, P., Yatskevich, M., “S-Match: an Algorithm and an Implementation of Semantic Matching,” *ESWS*, pp. 61-75, 2004.

[Gla09] Glaser, H., Millard, I.C., “RKBPlatform: Opening up Services in the Web of Data,” International Semantic Web Conference, 2009.

[Gol03] Golbeck, J., Fragoso, G., Hartel, F., Hendler, J., Oberthaler, J., Parsia, B., "The National Cancer Institute's Thesaurus and Ontology," *Journal of Web Semantics*, 1(1), December 2003.

[Gri] Dictionary of Philosophy of Mind, http://philosophy.uwaterloo.ca/MindDict/grice.html

[Gri57] Grice, H.P., "Meaning," *The Philosophical Review*, 64, pp. 377-388, 1957.

[Gri68] Grice, H.P., "Utterer's Meaning, Sentence-Meaning and Word-Meaning," *Foundations of Language*, 4, pp. 225-42, 1968.

[Gri69] Grice, H.P., "Utterer's Meaning and Intentions," *Philosophical Review*, 78, pp. 147-77, 1969.

[Gru93] Gruber, T. R., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *Stanford Knowledge Systems Laboratory Technical Report KSL-93-04*, 1993.

[Gua02] Guarino, N., Welty, C., "Evaluating Ontological Decisions with OntoClean," *Communications of the ACM*, 45(2), pp. 61-65, 2002.

[Hai06] Haidarian Shahri, H., Shahri, S., "Eliminating Duplicates in Information Integration: An Adaptive, Extensible Framework," *IEEE Intelligent Systems*, Vol. 21, No. 5, pp. 63-71, 2006.

[Hai08a] Hamid Haidarian Shahri, Donald Perlis, "Finding Ontological Correspondences for a Domain-Independent Natural Language Dialog Agent," Proceedings of the 20th AAAI Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI'08), Chicago, USA, July 13-17, 2008.

[Hai08d] Hamid Haidarian Shahri, James Hendler, Donald Perlis, “Grounding the Foundations of Ontology Mapping on the Neglected Interoperability Ambition,” Proceedings of the AAAI Spring Symposium on Semantic Scientific Knowledge Integration (AAAI/SSKI’08), Stanford University, CA, USA, March 26-28, 2008.

[Hai10b] Hamid Haidarian Shahri, Wikum Dinalankara, Scott Fults, Shomir Wilson, Don Perlis, Matt Schmill, Tim Oates, Darsana Josyula, Michael Anderson, “The Metacognitive Loop: An Architecture for Building Robust Intelligent Systems,” Proceedings of the AAAI Fall Symposium on Commonsense Knowledge (AAAI/CSK’10), Arlington, VA, USA, November 11-13, 2010.

[Hai10c] Hamid Haidarian Shahri, “Semantic Search in Linked Data: Opportunities and Challenges,” Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI’10), Atlanta, Georgia, USA, July 11-15, 2010.

[Hai10e] Hamid Haidarian Shahri, “Foundations of Data Interoperability on the Web: A Web Science Perspective,” Proceedings of the 6th International Conference on Semantic Systems (I-Semantics’10), Graz, Austria, September 1-3, 2010.

[Hal00] Halevy, A., Weld, D., “Intelligent Internet Systems,” *Artificial Intelligence*, 118(1-2), pp. 1-14, 2000.

[Hal05] Halevy, A., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., Sikka, V., “Enterprise information integration: successes, challenges and controversies,” *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD’05)*, Baltimore, Maryland, USA, June 14-16, pp. 778-787, 2005.

[Hal06] Halevy, A., Franklin, M., Maier, D., "Principles of Dataspace Systems," *Proceedings of Twenty-Fifth ACM Symposium on Principles of Database Systems (PODS'06)*, June 26-28, Chicago, Illinois, USA, pp. 1-9, 2006.

[Hal10] Halpin, H., Hayes, P., "When owl:sameAs isn't the Same: An Analysis of Identity Links on the Semantic Web," *Linked Data on the Web Workshop at the World Wide Web Conference (WWW'10)*, 2010.

[Har06] Andreas Harth, Jürgen Umbrich, Stefan Decker. "MultiCrawler: A Pipelined Architecture for Crawling and Indexing Semantic Web Data". 5th International Semantic Web Conference, Athens, GA, USA. November 5-9, 2006.

[Har10] Andreas Harth: VisiNav: A system for visual search and navigation on web data. J. Web Sem. 8(4): 348-354 (2010)

[Hea11] Heath, T., Bizer, C., Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool, 2011.

[Hef01] J. Heflin and J. Hendler, A portrait of the Semantic Web in action. IEEE Intelligent Systems, 16(2), 2001.

[Hen95] Hendler, J., Carbonell, J., Lenat, D., Mizoguchi, R., Rosenbloom, P., "VERY Large Knowledge Bases - Architecture vs. Engineering," *IJCAI 1995*, pp. 2033-2036, 1995.

[Her98] Hernandez, M.A., Stolfo, S.J., "Real-World Data Is Dirty: Data Cleansing and the Merge/Purge Problem," *Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 9-37, 1998.

[Hri02] Hristidis, V., Papakonstantinou, Y., "DISCOVER: Keyword Search in Relational Databases," *Proc. of the 28th International Conference on Very Large Data Bases (VLDB'02),* Hong Kong, China, August 20-23, 2002.

[Huy] Huynh, D., Freebase Parallax Software, at: http://mqlx.com/~david/parallax/

[Inr] INRIA ontology, http://fr.inrialpes.exmo.rdf.bib.owl.

[Isa09] Isaac, A., Wang, S., Zinn, C., Matthezing, H., van der Meij, L., Schlobach, S., "Evaluating Thesaurus Alignments for Semantic Interoperability in the Library Domain," *IEEE Intelligent Systems*, 24(2), pp. 76-86, 2009.

[Ive99] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution system for data integration. In Proc. of SIGMOD, 1999.

[Jag07] H. V. Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, Cong Yu: Making database systems usable. SIGMOD Conference 2007: 13-24

[Jos05] Josyula, D., Anderson, M., Perlis, D., "Designing a universal interfacing agent," *Proceedings of the Second Language and Technology Conference (L&TC'05)*, pp. 377-381, 2005.

[Kal03a] Kalfoglou, Y., Schorlemmer, M., "IF-map: an ontology mapping method based on information flow theory," Journal of Data Semantics, Vol.1, No.1, 2003.

[Kal03b] Kalfoglou, Y., Schorlemmer, M., "Ontology Mapping: The State of the Art," *Knowledge Engineering Review*, Vol. 18(1), pp. 1-31, 2003.

[Kal05] Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B., Hendler, H., "Swoop: A Web Ontology Editing Browser," *Journal of Web Semantics*, Vol. 4(2), 2005.

[Kam07] Kambhampati, S., Knoblock, C., "Information Integration on the Web," AAAI Tutorial, 2007.

[Kar] Karlsruhe ontology, http://www.aifb.uni-karlsruhe.de/ontology.

[Kno02] Knoblock, C., Kambhampati, S., "Information Integration on the Web," *Proc. of the Eighteenth National Conference on Artificial Intelligence (AAAI'02)*, July 29, 2002.

[Lae07] Laera, L, Blacoe, I., Tamma, V., Payne, T., Euzenat, J., Bench-Capon, T., "Argumentation over Ontology Correspondences in MAS," Proc. of 6th Int. Conference on Autonomous Agents and Mutli-Agent Systems (AAMAS'07), 2007.

[Lam99] E. Lambrecht, S. Kambhampati, and S. Gnanaprakasam. Optimizing recursive information gathering plans. In Proc. of the Int. Joint Conf. on AI (IJCAI), 1999.

[Len02] Lenzerini, M., "Data Integration: A Theoretical Perspective," *Proceedings of 21st ACM Symposium on Principles of Database Systems (PODS'02)*, June 3-5, Madison, Wisconsin, USA, pp. 233-246, 2002.

[Li00] Li, W., Clifton, C., "SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks." *Data Knowledge Eng*., 33(1), 2000.

[Mae01] A. Maedche and S. Saab. Ontology learning for the Semantic Web. IEEE Intelligent Systems, 16(2), 2001.

[Mar08] Marcus, A., "BlendDB: Blending Table Layouts to Support Efficient Browsing of Relational Databases," MSc Thesis, MIT, 2008.

[McG00] McGuinness, D., Fikes, R., Rice, J., Wilder, S., "An Environment for Merging and Testing Large Ontologies," *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, Breckenridge, Colorado, USA, 2000.

[McG03] McGuinness, D., "Ontologies Come of Age," In Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, editors. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2003.

[McG03] McGuinness, D., "Ontologies Come of Age," In Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, editors. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2003.

[Mel02] Melnik, S., Garcia-Molina, H., Rahm, E., "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching." *ICDE*, 2002.

[Mil00] R. Miller, L. Haas, and M. Hernandez. Schema mapping as query discovery. In Proc. of VLDB, 2000.

[Mil10] Ian Millard, Hugh Glaser, Manuel Salvadores, Nigel Shadbolt, "Consuming Multiple Linked Data Sources: Challenges and Experiences," First

International Workshop on Consuming Linked Data (COLD2010), Shanghai, China, 2010.

[Mil98] Milo, T., Zohar, S., "Using Schema Matching to Simplify Heterogeneous Data Translation," *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, 1998.

[Mit00] Mitra, P., Kersten, M., and Wiederhold, G., "A Graph-Oriented Model for Articulation of Ontology Interdependencies," *Proc. 7th Int. Conference on Extending Database Technology (EDBT'00)*, Konstanz, Germany, 2000.

[Mul10] Varish Mulwad, Tim Finin, Zareen Syed, and Anupam Joshi, Using Linked Data to interpret tables, Proceedings of the First International Workshop on Consuming Linked Data, held in conjunction with the Ninth International Semantic Web Conference, Shanghai, 2010.

[Myl98] Mylopoulos, J., "Next Generation Database Systems Won't Work Without Semantics," Panel Discussion, *Proc. of the 17th ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, Seattle, WA, June 2-4, 1998.

[Neu09] Neumann, T., Weikum, G., "Scalable Join Processing on Very Large RDF Graphs," Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2009), Providence, Rhode Island, USA, June 29 - July 2, 2009.

[New59] Newcombe, H., Kennedy, J., Axford, S., James, A., "Automatic Linkage of Vital Records," *Science*, vol. 130, no. 3381, pp. 954-959, 1959.

[Nir02] Nirenburg, S., Somers, H.L., Wilks, Y.A. (Eds.), *Readings in Machine Translation*, MIT Press, 2002.

[Noy00] Noy, N., Musen, M., “PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment,” *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00)*, Austin, TX, USA, July 2000.

[Noy01] Noy, N., McGuinness, D.L., “Ontology Development 101: A Guide to Creating Your First Ontology,” *Stanford Knowledge Systems Laboratory Technical Report* KSL-01-05 and *Stanford Medical Informatics Technical Report* SMI-2001-0880, March, 2001.

[Noy04] Noy, N., “Semantic Integration: A Survey of Ontology-Based Approaches,” *SIGMOD Record*, Vol. 33(4), pp. 65-70, 2004.

[Noy97] Noy, N., Hafner, C., “The State of the Art in Ontology Design: A Survey and Comparative Review,” *AI Magazine*, 18(3), pp. 53-74, 1997.

[OAEI] Ontology Alignment Evaluation Initiative, http://oaei.ontologymatching.org/

[Ome01] B. Omelayenko. Learning of ontologies for the Web: the analysis of existent approaches. In Proceedings of the International Workshop on Web Dynamics, 2001.

[Ont] http://ontologymatching.org/publications.html

[Ope] OpenLink Software, Virtuoso Open-Source License Terms, http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSLicense

[OWLg] Smith, M., Welty, C., McGuinness, D.L., "OWL Web Ontology Language Guide," *World Wide Web Consortium (W3C) Recommendation*, February, 2004.

[OWLo] McGuinness, D., van Harmelen, F., "OWL Web Ontology Language Overview," *World Wide Web Consortium (W3C) Recommendation*, February, 2004.

[OWLr] Dean, M., Schreiber, G., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L., "OWL Web Ontology Language Reference," *World Wide Web Consortium (W3C) Recommendation*, February, 2004.

[OWLu] Heflin, J., "OWL Web Ontology Language Use Cases and Requirements," *World Wide Web Consortium (W3C) Recommendation*, February, 2004.

[Par03] Parsons, S., Wooldridge, M., Amgoud, L., "On the outcomes of formal inter-agent dialogues," *Proc. of 2nd Int. Conference on Autonomous Agents and Mutli-Agent Systems (AAMAS'03)*, pp. 616-623, 2003.

[Par98] C. Parent, S. Spaccapietra. Issues and approaches of database integration. Communications of the ACM, 41(5):166–178, 1998.

[Per98] Perlis, D., Purang, K., Andersen, K., "Conversational Adequacy: Mistakes Are the Essence," *International Journal of Human-Computer Studies*, 48:553–575, 1998.

[Pot03] Pottinger, R., Bernstein, P., "Merging Models Based on Given Correspondences." Proc. VLDB 2003.

[Pru10] Use Cases and Requirements for Mapping Relational Databases to RDF, http://www.w3.org/TR/rdb2rdf-ucr/

[Rah01] Rahm, E., Bernstein, P., “A survey of approaches to automatic schema matching,” *VLDB Journal*, Vol. 10(4), pp. 334-350, 2001.

[Rah03] Rahwan, I., McBurney, P., Sonenberg, L., “Towards a theory of negotiation strategy (a preliminary report),” *Proc. of 5th Workshop on Game Theoretic and Decision Theoretic Agents (GTDT’03)*, pp. 73-80, 2003.

[RDFp] Manola, F., Miller, E., “RDF Primer,” *W3C Recommendation*, 2004.

[RDFv] Brickley, D., Guha, R.V., “RDF Vocabulary Description Language 1.0: RDF Schema,” *W3C Recommendation*, 2004.

[Sar08] Sarawagi, S., “Information Extraction,” *Foundations and Trends in Databases*, Vol. 1, No. 3, pp. 261–377, 2007.

[Say07] Sayyadian, M., LeKhac, H., Doan, A., Gravano, L., “Efficient Keyword Search across Heterogeneous Relational Databases,” *Proc. of the 23rd International Conference on Data Engineering (ICDE’07)*, Istanbul, Turkey, April 17-20, 2007.

[Sha06] Shadbolt, N., Berners-Lee, T., Hall, W., “The Semantic Web Revisited,” *IEEE Intelligent Systems*, Vol. 21(3), pp. 96-101, 2006.

[Sha91] Shadbolt, N., “Facts, Fantasies and Frameworks: The Design of a Knowledge Acquisition Workbench,” *Contemporary Knowledge Engineering and Cognition*, pp. 39-58, 1991.

[She90] Sheth, A., Larson, J., "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, 22(3), 1990.

[sin] http://sindice.com/

[Sir07] Sirin E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., Katz, Y., "Pellet: A practical OWL-DL reasoner," *Journal of Web Semantics*, 5(2), 2007.

[sta] http://www.statisticalengineering.com/index.html

[statsoft] http://www.statsoft.com/#

[Stu01] Stumme, G., Maedche, A., "FCA-merge: Bottomup Merging of Ontologies," *IJCAI 2001*, pp. 225-234, 2001.

[van08] van Harmelen, F., "Semantic web technologies as the foundation for the information infrastructure," In van Ooster, P. (Ed.), Creating Spatial Information Infrastructures, Wiley, 2008.

[Wal01] Walker, M., Kamm, C., Litman, D., "Towards developing general models of usability with PARADISE," *Natural Language Engineering: Special Issue on Best Practice in Spoken Dialogue Systems*, Vol. 6(3), 2001.

[Wei66] Weizenbaum, J., "ELIZA: A computer program for the study of natural language communication between man and Machine," *Communications of the ACM*, Vol. 9(1), pp. 36-45, 1966.