Multi-layered approach to aligning heterogeneous ontologies

William Sunna

Department of Computer Science, University of Illinois at Chicago
851 S. Morgan St. (M/C 152), Chicago, IL 60607, USA

Abstract

In various domain specific applications with heterogeneous databases, an ontology-driven approach to data integration relies on the alignment of the concepts of a global ontology that describes the domain, with the concepts of the ontologies that describe the data in the local databases. Once the alignment between the global ontology and each local ontology is established, agreements that encode a variety of mappings between concepts are derived. In this way, users can potentially query hundreds of distributed databases using a single query that hides the underlying heterogeneities. Using our approach, querying can be easily extended to new data sources. In this paper, we propose our "multi-layered" ontology alignment approach and the underlying algorithms that are used to establish mappings between the concepts. We also present the AgreementMaker, our ontology alignment tool that implements our approach, our tool displays the ontologies, supports several "mapping layers" visually, presents automatically generated mappings, and finally produces the agreements.

1 Introduction

Database concepts in various domains of knowledge are often categorized and described using ontologies. Such ontologies may be created independently by domain experts who have minimum or no communication among them. As a result, similar concepts can be described differently in a given domain of knowledge and their categorization can result in heterogeneous ontologies. Querying distributed databases represented by heterogenous ontologies collectively can be very challenging since it requires the querying system to be aware of each ontology that is used in every system and what each concept each ontology refers to. For this we have proposed an integration framework to facilitate the access to the information

Email address: [wsunna]@cs.uic.edu (William Sunna).

that is contained in distributed and heterogeneous databases [11]. Our approach relies on the alignment of ontologies, that is, on establishing mappings among related concepts in two heterogeneous ontologies. When such mappings have been established, we say that the two ontologies are aligned or matched. We consider two different network architectures: a centralized architecture and a peer-to-peer (P2P) architecture. In the former architecture, there is a global ontology. Each distributed ontology is aligned with the global ontology. As a consequence, a query expressed in terms of the concepts of the global ontology can be translated into a query to one of the distributed or local databases using the mappings that are established during the alignment process. In the latter architecture, a query to one of the peers, the source peer, can be translated into a query to another peer, the target peer, provided that the ontologies of the two peers have been aligned. Whichever the architecture, querying can be easily extended to new databases. In this paper we refer to a global ontology or a source peer as a source ontology and we refer to a local ontology or a target peer as the target ontology.

In order to resolve heterogeneities and bridge the gap between distributed systems, we propose our multi-layered approach to ontology alignment, whose functionality extends that of our previous work [11,13,12]. In our approach, mappings are determined semi-automatically, using both automatic and manual methods. To establish such mappings, our approach uses four mapping layers. Three of these layers use automatic methods and the other one uses manual methods.In the first layer, the concepts of the ontologies are automatically mapped to each other based on their definition and relative locations in the ontological structures. The user manually maps concepts in the second alignment layer where mapping types are used to specify the relationships between mapped concepts. In the third layer, an automatic procedure is invoked which deduces more mappings by looking at previous mappings in the context of the ontologies. Finally, in the fourth layer, an automatic procedure is used to consolidate all the mappings produced from previous alignment layers. To apply our multi-layered approach, we created a visual tool, the AgreementMaker, which implements our four layers of mappings. Our tool allows the user to load two ontologies that get displayed side by side to be aligned. Upon aligning the ontologies, our tool generates an agreement document which stores the results of the mappings from the four layers to be used with end applications such as those providing querying capabilities across distributed systems. Our tool provides a graphical user interface that allows the user to perform manual mappings and invoke automatic mappings in the the various mapping layers. In creating the graphical user interface, we took into consideration many visual issues such as how to display large ontologies, how to present the results of the mapping layers to the user, and how to avoid congestion of the displayed information.

Several ontology alignment techniques and strategies have been proposed to achieve the goal of database interpretability. On one hand, some of these techniques make use of the labels, the properties, and the definitions the concepts to determine the mappings between them. On the other hand, other techniques make use of the location of the concepts in the ontological structure to determine the mappings. In establishing mappings between concepts, similarities are determined to measure how close or accurate the mappings are relative to reality. Achieving high measures of similarities will increase the level of confidence in the

alignment technique, therefore designers of such techniques always try to make use of whatever information they can obtain about a concept in the source ontology in order to find a very good match for it in the target ontology. In this paper, we will present an overview of several alignment techniques and alinement tools in Section 2. As far as our alignment approach is concerned, we will concentrate on our first layer of mapping where we propose three similarity calculation algorithms that are used to map concepts. The first algorithm simply measures how similar concepts are to each other by examining their definitions as provided by a dictionary, the second algorithm reconfigures the similarity results from the first algorithm by considering the similarities of the parent concepts. Finally, the third algorithm reconfigures the similarity of the first algorithm by considering the similarity of sibling concepts.

The rest of this paper is organized as follows. In Section 2, we give an overview of related work in the area. We present our multi-layered approach to ontology alignment and an overview of our alignment tool in Section 3. In Section 4 we present our automatic similarity algorithms that support the first layer of mapping in our multi-layered approach along with the results of applying these algorithms on four sets of heterogenous ontologies. Finally, in Section 5, we outline our future work.

2 Related Work

In this section, we cover several ontology alignment(matching) techniques that are close to our alignment techniques. In addition, we survey several visual ontology alignment tools.

2.1 Ontology Alignment Techniques

In their survey paper, Shvaiko and Euzenat [33] provide a comparative review of recent schema and ontology matching techniques in the context of a new classification system they propose. In their system, they present a classification system of ontology based matching techniques (strategies), the techniques are classified as element level or structure level. In the element level, the techniques are further classified into string based, language based, linguistic based, constraint based, or alignment reuse based. In the structure level, the techniques are further classified as graph based, taxonomy based, or model based. In order to derive mappings between concepts during the alignment process of ontology, the element based techniques consider the labels of concepts, their definitions, the language they are expressed in, and any possibility to reuse previous mappings to derive new ones. Likewise, the structure based techniques consider the location of the concept in the ontological structure whether it was a tree or a graph or any model that the ontology is expressed in and how the mappings of concepts can contribute to the mappings of adjacent concepts. According to their classification system, our alignment techniques fall into a subset of element level category because of our definition mapping layer, and structure level category because of

our context mapping layer. We will present our alignment techniques in more details in Section 4. Next we present a survey of some of the ontology based matching techniques and where they fall in Shvaiko and Euzenat classification.

OLA [17] is an alignment tool that was developed by teams at the University of Montreal and INRIA Rhone Alpes. The main purpose of this tool is to align ontologies expressed in OWL [4]. OLA offers parsing and visualization of OWL-Lite and OWL-DL ontologies, in addition it offers computations of similarities between concepts from the ontologies being aligned. OLA employs linguistic element level and structure level based techniques. OLA allows manual construction of alignments by composing entity pairs from the two aligned ontologies and it uses existing mappings as a starter for automated mappings. In OLA, the available knowledge about the concepts in the aligned ontologies is taken into consideration prior to the alignment process, this requires the tool user who is performing the alignment to make some selections in choosing appropriate alignment methods available in the tool. In designing OLA, one main goal is to achieve the highest level of automation since full automation cannot be achieved. For this goal to be satisfied, OLA expects the user to provide a minimal set of parameters at the initial steps of the alignment process and then leave it for the tool to accomplish the full alignment task at the end without any human intervention. OLA follows category-dependent comparison strategy that is concepts are categorized as classes, objects, properties, and relations, only concepts of the same category of classification will be compared. Furthermore, for each category there are sets of customized similarity functions that are used in establishing mappings between the concepts in the category, unlike what we have in our approach, OLA has the limitations that similarities between concepts do not contribute or affect the similarities of their neighbors. The first step in OLA's alignment process is to parse the input OWL ontologies and build a labeled OL-Graph [18], vertices on the graph are further categorized to classes, objects, relations, properties, property instances, data types, data values, or property instances. The OL-Graph allows the expression of specialization between classes and relations, instantiation between objects and classes, attribution between classes and properties, and valuation between properties and objects. The second step after building the OL-Graph is to apply the category oriented similarity functions, some of these functions employ string distances and lexical distances to compute the similarities with the help of the WordNet dictionary [29].

RiMOM [34] (Risk Minimization based Ontology Mapping) is a system that intends to combine different strategies to achieve optimal alignment from a source ontology to a target ontology that are input to the system. There are two types of defined strategies in the system: linguistic based techniques (includes edit-distance and statistical-learning), and structure based techniques (includes similarity-propagation, property-to-property propagation, and concept-to-property propagation). RiMOM first examines the structural similarity of the ontologies and the label similarity of the concepts in the ontologies to determine which strategies to use in the alignment process. For example, if there is a high similarity in labels, RiMOM will rely more on linguistic based strategies to find the matchings between concepts. RiMOM then applies the selected alignment strategies; each strategy outputs its own independent results, the results are then combined using a linear-interpolation method. Finaly, RiMOM applies a refinement procedure to prune alignments which represent bad matchings

between concepts. RiMOM aims to address mainly three types of challenges: (1) achieving high quality alignments using automatic matching algorithms; (2) enhance performance of the system to find the alignments efficiently; and (3) automatically adjusting the selection of alignment strategies for different sets of source and target ontologies. Compared to our approach, we are also using multiple matching techniques, some of the differences of our approach is that we take into consideration the relationships between the various concepts in the ontology in deciding the similarities. That is to say that if two concepts match, then most likely their children will match with a high similarity and their grand children will match with a lower similarity measure. Compared to their approach in combing results of the various strategies, our tool allows experts to review the results of the mapping strategies and determine the importance of each one, this way we reach better end results and allow the user the flexibility to rank the results from our mapping strategies in different ways.

In their paper [28], Melnik *et al.* propose a simple structural model based level technique that can be used in matching a variety of data structures (referred to as models). Models can be data schemas, data instances, or a mixture of both. In their approach, models are converted to directed labelled graphs. Their matching algorithm takes two graphs to be aligned as an input and generates mappings between their corresponding concepts. For their algorithm to work, they rely on the fact that concepts from the two graphs are similar when their adjacent concepts on the graphs are similar, the algorithm starts with obtaining initial mappings between concepts in the two input graphs using a string matching function which returns initial similarities between matched concepts. Having established the initial mappings, the algorithm proceeds in iterations to establish more mappings between other concepts, this is done based on the assumption that in every iteration whenever any two concepts in the input models matched with some similarity measure, the similarity of their adjacent concepts increases. The iterations continue "flooding" the similarities across the concepts in the graphs until a fixed point is reached where similarity measures for all concepts have been stabilized.

FOAM [5] (Framework for Ontology Alignment and Mapping) is proposed by Ehrig *et al.* In their framework, they try to address several requirements that may have been overlooked by existing alignment methods. Theses requirements are: (1) obtaining high quality results; (2) Achieving high level of efficiency when aligning two ontologies; (3) allow users to interact when needed; (4) provide flexibility with respect to use cases; and (5) allow for adjustment and parametrization. To meet the requirement of achieving high quality alignment results, FOAM employs structure level based techniques. For example, entities are considered similar to each other if their super concepts are similar. This resembles our mapping by context layer approach where mappings propagate automatically from concepts to their parents in the ontological trees. To meet the efficiency requirements, FOAM implements an intelligent selection algorithm to single out concepts that are good candidates to be matched. In order to provide flexibility in ontology alignment based on use cases, the system can automatically select between mapping algorithms for its input ontologies, for example, if the ontologies are very large, the system selects efficient approaches to be used for the alignment of these large ontologies.

In their paper [24] Lopez *et al.* shift the focus of the reader away from traditional ontology

mapping to other requirements that have been neglected. Some examples of these requirements are: (1) matching multiple online ontologies instead of two only; (2) mapping ontologies that are not close to one another; (3) focus on efficiency not only the quality of matching like most approaches concentrate on; and (4) concentrate on instance level matching as it has been ignored because the focus was always on concept level matching. They introduce their PowerMap algorithm which represents a new generation of algorithms where ontologies get mapped at run time instead at the design time. To allow for this, semantic search engines were built (such as Swoogle [15]) which can crawl and index the online sematic data to enable the PowerMap algorithm to go online and retrieve ontologies that are related to the user's query. The PowerMap algorithm is the core component of PowerAqua ontology based question answering application [23] that tries to answer questions asked in natural language by making use of the available online semantic data. The PowerAqua application proceeds with determining the ontologies that are relevant to the question asked by the user, element based matching techniques are used at this step to identify such ontologies. Once the ontologies are retrieved, PowerAqua applies a filtering algorithm to exclude ontologies that may be ambiguous and only keep those who can potentially provide the most amount of information in answering the user's question. To allow for this, PowerAqua uses more complex algorithms, for example, if the user question contains the term "Capital" which refers to the geographical city where the government of a county is located, then PowerAqua will exclude ontologies that contain information related to "Capital" which refer to an asset or wealth. After excluding the ambiguous ontologies, the application will return the remaining sets of ontologies that jointly contain enough information to provide the answer to the question posed by the user. Our approach is widely different from PowerMap, we concentrate on mapping two ontologies to each other, while PowerMap concentrates on mapping the user's questions to a collection of online ontologies on run time. However, some of the matching techniques that are used in the PowerMap algorithm are similar to the one we are using such as relying on WorldNet [29] and syntactic matching algorithm. One difference is that our system also relies on deducing mappings based on previous mappings, and taking advantage of similarities of concepts in determining the similarity of their neighbors.

Silva *et al.* [6] discuss the situation when different mapping agents establish different semantic bridges between concepts of two ontologies, namely the source ontology and the target ontology. A mapping agent is defined as an approach or a strategy that is used to establish mapping from the concepts in a source ontology to the concepts of a target ontology. Each mapping is referred to as a semantic bridge. Due to the inherent and subjective nature of ontologies, different agents establish different semantic bridges for the same set of ontologies. This may cause conflicts when interpretability occurs between such agents when they are used to align two ontologies. To address this issue, they propose an approach to ontology mapping negotiation where various agents are able to achieve consensus among each others. Their approach is based on utility functions that evaluate the confidence of the mapping rules established by the agents and according to the confidence measure, the mapping rule proposed by an agent is accepted, rejected, or negotiated. Negotiation of mapping rules requires relaxation (increase) of confidence values so the mapping rules get accepted. A utility function is used to determine whether the relaxation should take place or not based on evaluating the effort needed to do that. While expert intervention is needed for the negotiation

process, Silva *et al.* try to establish an automatic consensus building algorithm among two agents. In our approach, multiple mapping strategies are used (alignment layers), each layer proposes a set of mappings between the source ontology and the target ontology. At the very end, a consolidation mapping layer is applied where users supply a priority measures for the other mapping layers, if the mapping layers are in conflict, the layer that has the highest priority measure will dominate and considered for the final results.

2.2 Visual Ontology Alignment Tools

In this section, we present a survey of some visual ontology alignment tools and the features they present.

Chimaera [27] is a software tool developed by the KSL group at Stanford, and provides tools for merging different ontologies created by different authors and for diagnosing individual or multiple ontologies. Chimaera is implemented as a web-based, its graphical user interface consists of a set of commands accessible via spring loaded menus and supports drag and drop editing. The interface displays the classes and slots in current knowledge base being edited and allows users to check the automated merging procedure by highlighting the classes that need the user's attention.

COMA++ [7] is a schema and ontology mapping tool which utilizes several matching algorithms. The tool is mainly composed of five parts: the Model and the Mapping Pool which provide data management for the ontologies and their matchings in the memory, the Repository which stores match-related data, the Match Customizer which configures the matchers and match strategies, and finally the Execution Engine which performs the match operations. COMA++ supports iterative automatic matching of schema or ontology components: first, the candidate components for mappings are identified, then several matching algorithms are performed in the Execution Engine to calculate similarities. Finally the results of the similarity calculations are combined to derive correspondences between the candidate components. The automatic matching process proceeds in iterations where the results of every iteration can be used as an input to the next iteration. COMA++ implements several matching strategies such as: (1) the fragment based divide and conquer matching strategy which decomposes the ontologies into fragments and then tries to match the fragments first before matching their components. (2) reuse oriented matching which makes use of previous mappings in deciding new ones. COMA++ supports multiple schema and ontology formats such as OWL, XSD, and XML. In comparison, we have a similar approach because we also use several matching algorithms in our mapping process. In addition, we reuse previous mappings to discover new mappings in an iterative fashion. The graphical user interface of COMA++ is quite similar to ours. Besides a menu, there are three main areas; a panel for management and match functions on the left, and two panes for displaying the source and target schemas.

Falcon-AO [22] is an automatic ontology alignment tool that matches based on linguistic and graph matching of the ontologies. Falcon-AO's graphical interface provides users with the

view of entities from both the ontologies being mapped and also the result of the automatic mapping performed on them. Unlike our tool, Falcon-AO does not represent the input ontologies as tree structures. Similar to our tool's definition mapping strategy, Falcon tries to align ontologies using linguistic similarity between two entities relying on their names, labels, comments and other descriptive information. Falcon-AO's linguistic matcher relies on lexical comparison and statistical analysis. The graph matcher in Falcon-AO uses directed bipartite graphs to represent ontologies and measures the structural similarity between graphs.

Clio [20] is a graphical tool used for semi-automatically mapping simple relational and XML schemas. In contrast, our mapping tool supports mapping of different types of ontologies such as OWL [4] in addition to mapping XML schemas. Using Clio, the user loads a source schema and a target schema and establishes connections between objects in both schemas graphically. Such connections are referred to as value correspondences, which express how an object or more in the source schema are transformed into a target value. Clio has a mapping engine that incrementally produces SQL queries that realize the mappings implied by the correspondences. Clio systematically ranks and generates mappings and helps the user in selecting between alternative mappings if needed. Our mapping tool also provides automatic help to the user by evaluating previous mappings to produce new ones. Clio's graphical user interface operates in two modes: (1) the Schema View mode which allows the user to see a representation of the schemas and edit any of them. (2) the Data View mode which helps the user to see the actual data related to the schema.

MapOnto [3], inspired by Clio, is a research prototype for semantically mapping between database schemas and ontologies as well as between different database schemas. MapOnto works in an interactive and semi-automatic manner, taking input from user for creating simple attribute-to-attribute correspondence and finally selecting the best mapping out of a list of logical formulas generated. These logical formulas are generated by the tool using knowledge embedded in the ontologies. Also, these logical formulas are ordered to suggest to the user the most reasonable mapping between the two models. MapOnto's graphical interface is based on Protégé [19]. Unlike our tool, the correspondences between attributes are not represented by lines in the interface, but as logical formulas displayed in a separate pane.

3 The AgreementMaker Framework

We have been working on a framework that supports the alignment of two heterogenous ontologies. In our framework, we introduce an alignment approach which utilizes different matching techniques between the concepts of the aligned ontologies. each matching technique is embedded in a what we refer to as mapping layer [14]. We currently have four layers in our framework with the possibility of adding more mapping layers in the future. Unlike all the alignment techniques mentioned in Section 2, the motivation behind our framework is to allow for the addition of as many mapping layers as possible in order to capture a wide range of relationships between concepts. Our current mapping layers utilize element based

alignment techniques (first layer) and structural based alignment techniques (first layer and third layer). This is in addition to allowing domain expert to utilize their knowledge in order to contribute to the alignment process (second layer). We also developed a software tool which implements our approach, the name of our tool is the AgreementMaker. Next we present a summary of our multi-layered approach and our alignment tool.

3.1 Multi-Layered Approach

In our ontology alignment approach, the mappings between concepts of two heterogenous ontologies are determined semi-automatically, that is, they are partly established manually by the user and partly determined using several automatic procedures. We refer to this semi-automatic approach as a multi-layered approach which consists of four mapping layers. Three of these layers are automatic and one is manual. In the first layer, the concepts of the two ontologies are automatically compared to each other based on their definition and relative location to the other concepts in the ontological structures as will be explained in Section 4. The user manually matches concepts in the second alignment layer where mapping types are used to specify the relationships between mapped concepts. In the third layer, an automatic bottom-up deduction procedure is invoked which deduces more mappings by looking at previous mappings in the context of the ontologies [10]. Finally, in the fourth layer, an automatic procedure is used to consolidate all the mappings produced from previous alignment layers. In contrast to our third layer of mapping, we are considering adding a layer that embeds a top-bottom deduction approach as described in [32]. Depending on the nature of the application, some layers may play more important role in aligning the ontologies modelling the application than other layers. In this paper, we concentrate on the first layer where we explore how similarities between concepts contribute to the similarity of their adjacent concepts in many different ways. For more details about our multi-layered approach, please refer to [14].

3.2 The AgreementMaker

The AgreementMaker is our software tool which implements our multi-layered approach to ontology alignment. The interface of our tool allows for the user to load two ontologies side-by-side and display each one as a tree of concepts as shown in Figure 1. We refer to the first ontology as the source, and to the second ontology as the target. After loading the ontologies, users can start the alignment process by mapping corresponding concepts manually and/or by invoking procedures that map them automatically. The mapping information is displayed in the form of annotated lines connecting nodes. The source ontology is displayed on the left hand side and the target one is displayed on the right hand side. The interface allows for selective expansion or contraction of parts of the ontological trees to facilitate browsing and mapping. Concepts names are displayed in rectangular nodes on the ontological trees, while additional information can be either displayed or hidden to avoid visual overloading. Users

can customize the interface by changing the background color of the canvas, the color of the ontological trees, and the color of the connecting lines.

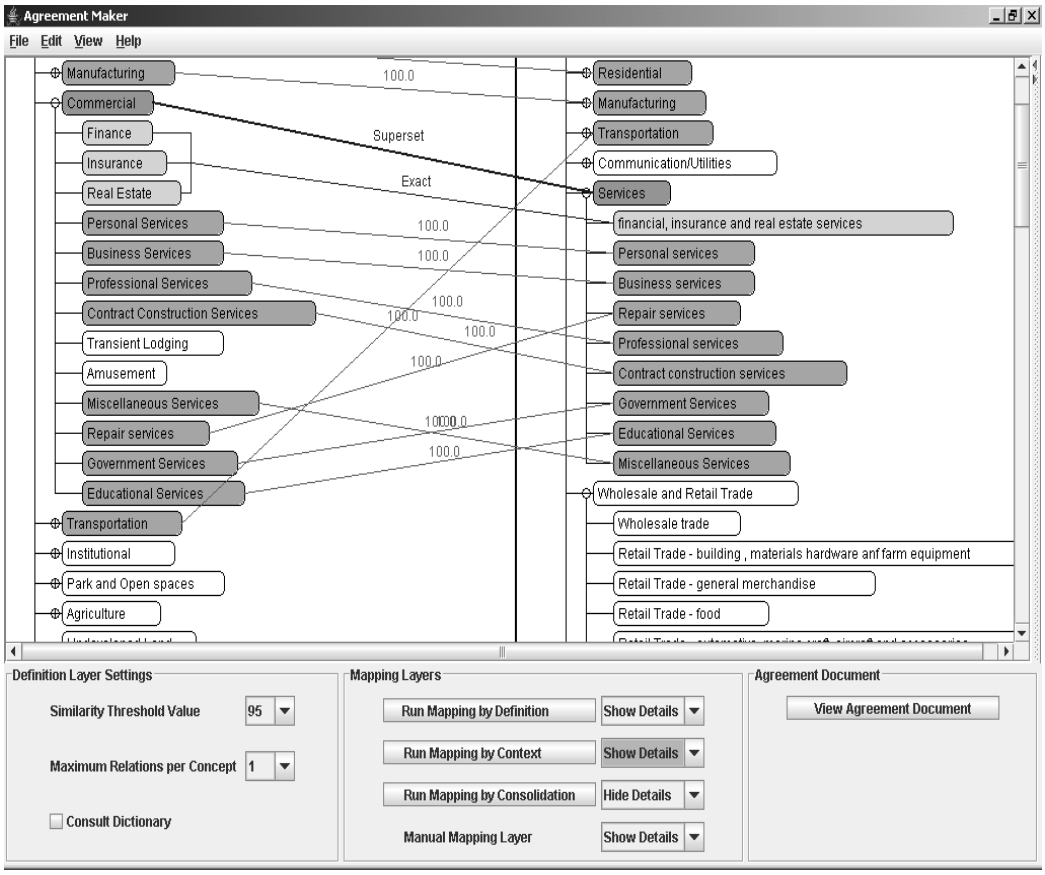


Fig. 1. Results of running three of the mapping layers.

The underlying infrastructure of the AgreementMaker consists of four main modules: The *Graphical User Interface*, the *Ontology Parser*, the *Mapping Engine*, and the *Agreement Document Generator*. Figure 2 shows a diagram of the architecture of our tool. Following, we describe each module in our tool:

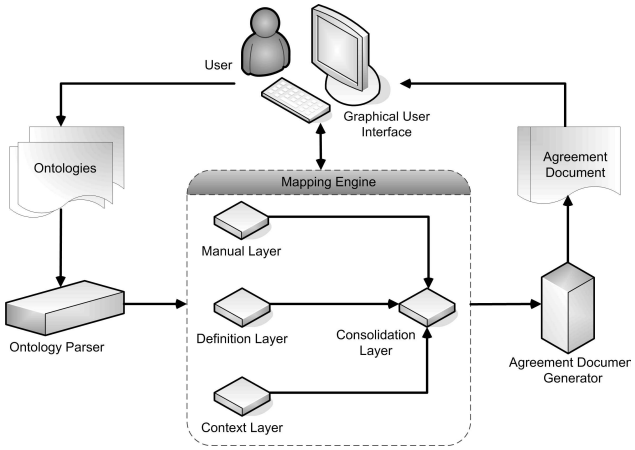


Fig. 2. System architecture of the *AgreementMaker*.

3.2.1 Graphical User Interface

The graphical user interface (Figure 3) assists the user in making the mapping decisions. It is customizable, allowing the user to select colors for the various objects and to expand or collapse parts of the ontologies. The menu bar supports standard operations such as opening files, undoing and redoing actions, and getting help. The central part of the user interface displays the rendered ontologies. It also displays the results of the mappings that result from the various mapping layers. In addition, the interface is capable of displaying information such as comments, properties, and class relationships for any selected concept in a separate pane (hidden in the figure). This information can help the user when mapping concepts manually. Finally, the control panel on the lower part of the interface contains buttons to invoke the various automatic mapping procedures and to select whether to show or hide results of any of the four mapping layers.

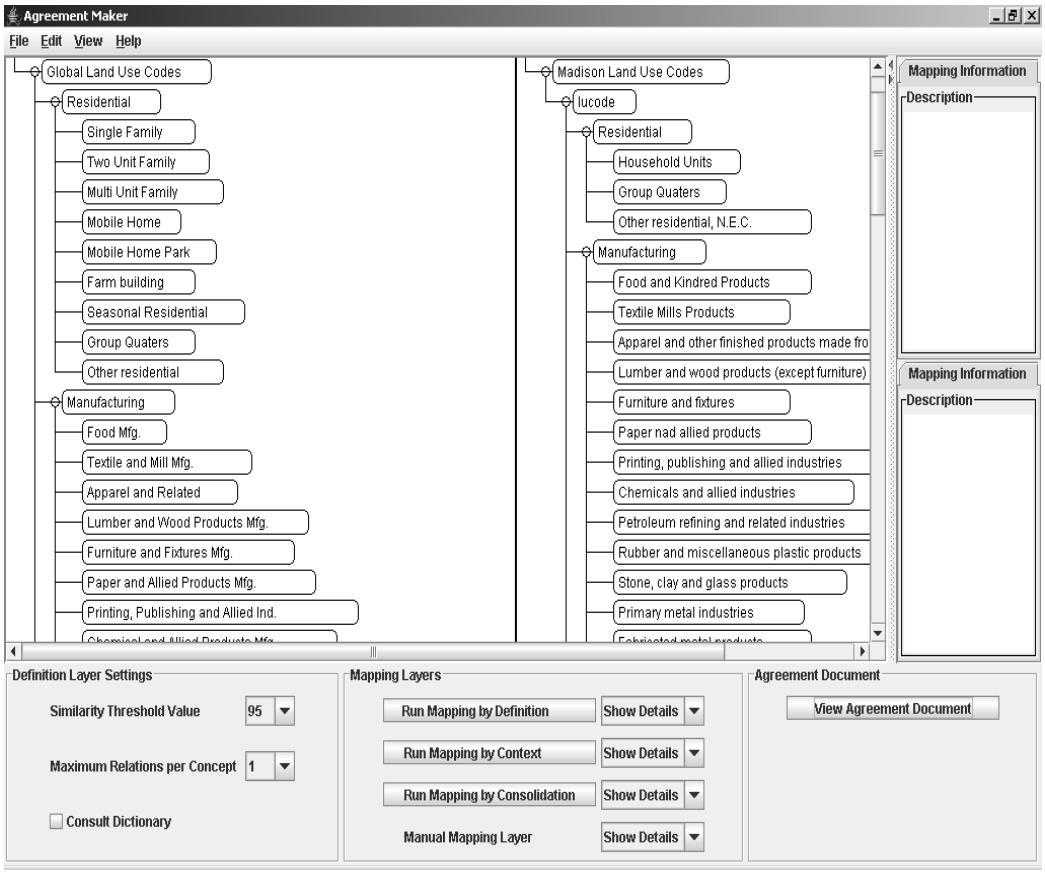


Fig. 3. The User Graphical Interface of the *AgreementMaker* showing the Menu Bar on the top, the Description Pane on the right, the Control Panel on the bottom, and the Canvas in the middle where the global ontology is displayed on the left and the local ontology on the right.

3.2.2 The Ontology Parser

The AgreementMaker maps ontologies expressed in XML, RDFS, OWL, or N3 [8]. These ontologies are parsed and converted into our own tree structures using this module. The

ontology files are parsed using various application programming interfaces (APIs): Xalan APIs are used for XML schemas, Jena APIs [25] are used for RDFS [26] ontologies, and OWL Pellet [4] is used for N3 and OWL ontologies.

3.2.3 The Mapping Engine

This module is responsible for running the matching algorithms of the four mapping layers on the loaded ontologies. The mapping engine reports the matching results in the form of lines connecting concepts from the source ontology to the target ontology. In a typical alignment session, the user invokes the definition mapping layer, then performs manual mappings, and then invokes the mapping by context layer [10] in an iterative fashion. After the mappings are established using the three layers, the user invokes the mapping by consolidation layer [12] to generate the final results.

3.2.4 The Agreement Document Generator

This module takes the information of the mappings generated by the mapping engine and stores it in the agreement document, which is the final output of the alignment process. The agreement document contains the alignment information that relates the two ontologies in an XML file. In addition, our tool is extendible and can be configured to reformat the agreement document in any format that is convenient for the end systems that use the document.

3.3 User Interaction

In the process of demonstrating the usability of the AgreementMaker in this section, we will use two types of geospatial ontologies. The first type is in the domain of wetland classification systems, and the second type is in the domain of land use code specification. For wetland ontologies, we will use the "Cowardin" wetland classification system [9] which is adopted by the United States as the source ontology , and the the wetland classification system used in South Africa [16] as the target ontology. As for the land use code ontologies, we will use a global land use classification ontology for the state of Wisconsin which describes land use codes on the state level as the source ontology. For the target ontology, we will use a local land classification ontology which describes land use codes for a selected county in the state of Wisconsin [13].

Our tool enables the user to map concepts from the source ontologies to target ontologies using the graphical user interface. Upon loading the ontologies in our tool, they will be displayed in tree like structures as shown in Figure 4. In the figure, the user loaded the "Cowardin" wetland classification as the source ontology which appears on the left hand side, and the the South African wetland classification system as the target ontology which appears on the right hand side. The ontologies are displayed on the canvas in a hierarchal fashion where lower level concepts belong to higher level ones. For example, the concepts

Rock Bottom, *Unconsolidated Bottom*, *Aquatic Bed*, and *Reef* belong to the concept *Subtidal* which in turns belong to the concept *Marine* in the source ontology.

In designing our user interface, we addressed many issues which affect the usability of our tool. We list several of these issues next and outline the rationale for our design decisions

3.3.1 Ontology display

Our ontologies are displayed as outline trees in the main central part of the user interface of the tool as shown in Figure 1. By using the outline trees, ontology browsing becomes similar to directory browsing, which is familiar to most users. We have implemented such trees so as to allow for the selective expansion or contraction of parts of the tree to facilitate browsing and mapping especially in the case when the ontologies are large.

3.3.2 Meta information display

Ontologies, such as those represented in OWL, have properties associated with their concepts, which can be used in the definition layer to compute similarity among the concepts. In addition, by displaying it, such information can be taken into account by the user when establishing manual mappings. If this information were displayed on the main canvas together with the ontologies, then it would cause visual overloading. Furthermore, it could interfere with the readability of the concept names. For these reasons, we provide a description pane (which can be hidden when not needed) that displays the description of any selected concept. For example, the description of the concept *Intertidal* appears in the upper part of the description pane as shown in Figure 4. The upper part of the description pane is dedicated to the display of information that is associated with the concepts of the source ontology while the lower part is dedicated to the display of information for the local target ontology.

3.3.3 Similarity display

Upon invoking the mapping by definition layer, a measure of the similarity among concepts is calculated to determine possible mappings and lines are drawn that display those possible mappings. To increase the clarity of the picture, the user can specify a similarity threshold so that only the lines that have similarity measures greater or equal to the selected threshold will be displayed. In addition, a maximum number of such possible mappings (as associated with each concept) can be specified. Figure 5 shows the result of running the definition layer, as displayed to the user. In this particular case, the user has selected a threshold of 75%. The user also selected to see a maximum of two relations per concept in the source ontology. For example, the concept *Aquatic Bed* of the *Subtidal Estuarine* subcategory in the source ontology is related to the concept *Aquatic Bed* of the *Subtidal Marine* subcategory with a similarity measure of 75%. At the same time, it is related to the concept *Aquatic Bed* of the *Subtidal Estuarine* subcategory in the target ontology with a similarity measure of 100%. As previously mentioned, the user can collapse and expand the ontology trees to display

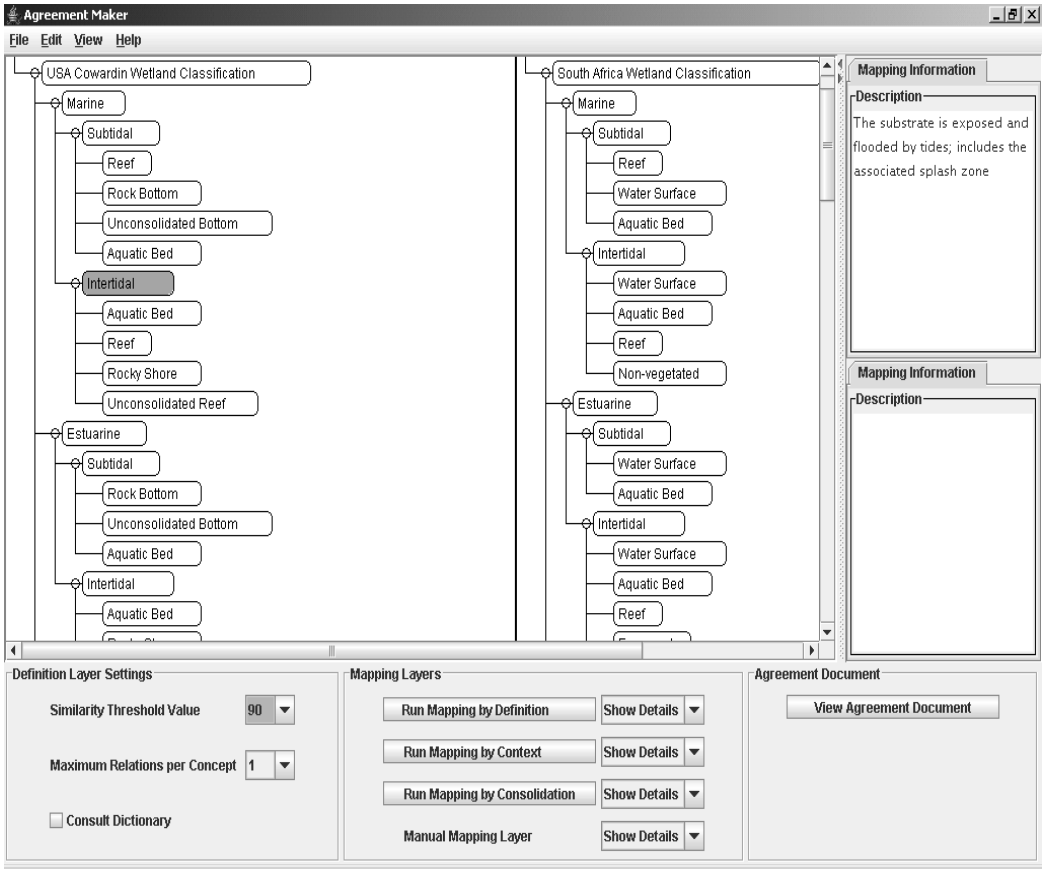


Fig. 4. Description of the selected concept appears in the description pane.

only the concepts of interest and therefore only the associated similarities. To facilitate the reading of the lines that display the matched concepts, they can be highlighted and made bold when the associated concept is selected. For example, in Figure 5 the concept *Aquatic Bed* has been selected and therefore both the matching lines that are connected to it and their similarity measures are highlighted and made bold.

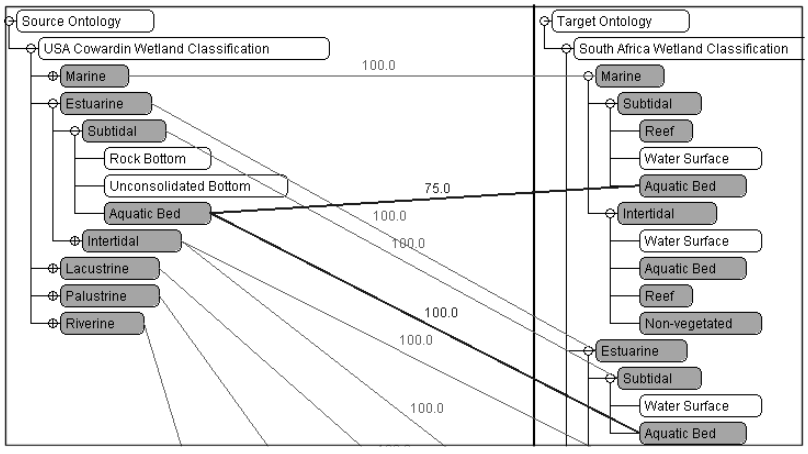


Fig. 5. Results of running the mapping by definition layer.

3.3.4 Manual mapping

To facilitate the manual mappings that are performed by the user, a mapping menu is used that contains various mapping types that are used to relate concepts [13]. The user can select one or more concepts in the source ontology and map them to one or more concepts in the target ontology. Figure 6 shows an example where the user is mapping the concept *Aviation* to the concept *Aircraft Transportation* as an exact match for two ontologies describing land use codes [13]. After manually establishing a connection between those concepts, a menu that displays the different mapping types is displayed, so that the most appropriate mapping type can be chosen.

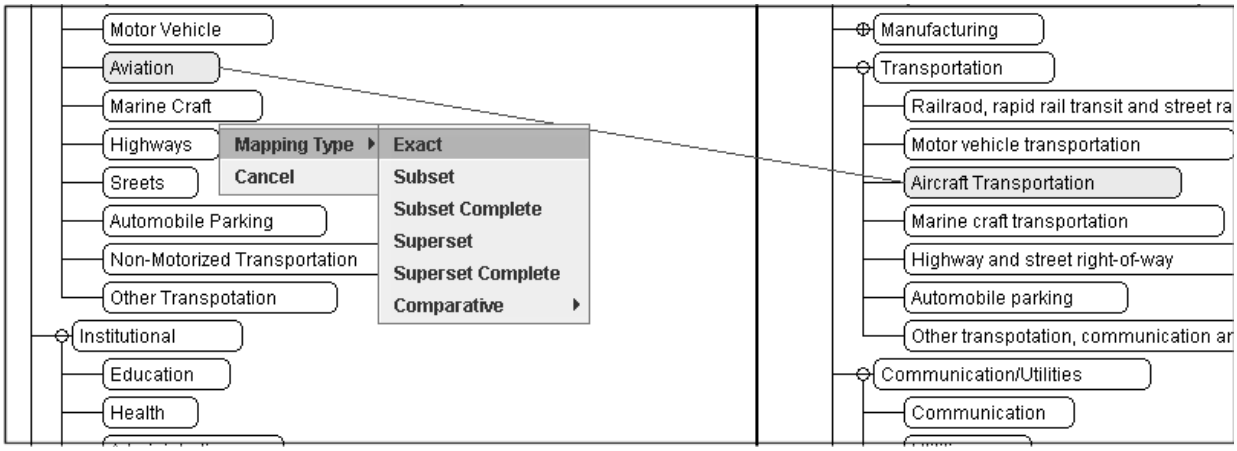


Fig. 6. Performing manual mapping between two concepts.

3.3.5 Simultaneous matching displays

An important issue is related to the presentation of the results of all the mapping layers to the user without visual overloading. To resolve this issue, the concepts that are mapped in a given layer are displayed using the same color. In this way, the user can easily distinguish which concepts are mapped by which layers. Another issue is the display of matching concepts that have been mapped by more than one layer. To improve readability, we allow the user to hide the results of any of the mapping layers and to redisplay them as desired. In this way, users can focus only on a subset of the layers at a time. Figure 1 shows the mappings that result from two of the mapping layers when aligning ontologies describing land use codes [13]. In the figure, lines with similarity measures result from the definition layer. Concepts that were connected in this way include *Personal Services*, *Business Services*, and *Professional Services*. Other concepts such as *Finance*, *Insurance*, and *Real Estate* in the source ontology were mapped manually by the user.

3.3.6 Color selection.

We give the user the flexibility of choosing the colors of the various visual components. This feature may be important for users who suffer from color blindness or other related visual

problems. Our tool enables the user to change the color of the background, of the ontology, and of the highlighted concepts. The tool also enables the user to choose the color of the similarity lines that are produced by the definition mapping layer, by the manual mapping layer, and by the context mapping layer.

4 Automatic Similarity Algorithms

In order to achieve a high level of confidence in performing automatic alignment of two heterogenous ontologies, thorough understanding of the identities of the concepts of the aligned ontologies is highly desired. For this we propose algorithms that investigate the identities of the ontological concepts prior to making a decision on how they should be mapped. In our algorithm we consider the labels and the definitions of the ontological concepts on one hand and the relative positions of concepts in the ontological tree on the other hand. We aim to utilize every piece of information about a concept to realize and understand what entity in reality it refers to. Our alignment algorithm enables the user to select one of the following three matching procedures: (1) applying the base similarity calculations only; (2) applying the base similarity calculations followed by the Descendant's Similarity Inheritance *(DSI)* algorithm; or (3) applying the base similarity calculations followed by the Sibling's Similarity Contribution *(SSC)* algorithm. We apply any of these procedures in our first layer of mapping, next we explain each of these procedures in greater details.

4.1 Base similarity calculations

The very first step in our approach is to establish initial mappings between the concepts of the source ontology and the concepts of the target ontology. These initial mappings will be a starting point for both the *DSI* and *SSC* algorithms as will be explained in Sections 4.2 and 4.3. In order to accomplish this task, we try to find matching concepts in the target ontology for each concept in the source ontology. This is achieved by defining a similarity function that takes a concept in the source ontology and a concept in the target ontology and returns a similarity measure between them. If the similarity measure is equal or above a ceratin reasonable threshold decided by the user, then the two input concepts are considered a match to one another. In what follows we present the details of finding base similarities between a concept in the source ontology and a concept in the local ontology:

- Let S be a the source ontology.
- Let T be a the target ontology.
- Let C be a concept in the source ontology S.
- Let C' be a concept in the target ontology T.
- Let *base_sim(C,C') = M* be a similarity function such that when applied to C and C', it yields a similarity measure M between the values *0%* and *100%*.

- Let *TH* be a threshold value such that if $M \geq TH$, then *C'* is considered a matched concept with *C*.
- $\forall\ C \in S$, find all C'$\in T$ such that *base_sim(C,C')* $\geq TH$
- Let *MS(C)* be a set that contains all concepts in T that match the concept C in S after applying the base similarity algorithm the concept C with all the concepts in T. We will refer to this set as the *Mapping Set* of the concept C

In order to find the similarity measure between two concepts, we utilize their labels and definitions as provided by a dictionary. Following are the steps in determining the similarity between two concepts C and C' by looking at their labels (*label(C)* and *label(C')*):

- If *label(C)* is identical to the *label(C')*, then return a similarity of *100%*
- Else, apply the *treat_string()* function on *label(C)* and *label(C')*, this function separates a composite string that contains multiple words into its components. For example, in an ontology that contains concepts related to types of weapons, if *label(C)* = *"air-to-air-missile"*, then the string of the label will be converted to *"air to air missile"* after applying the function. Similarly, if *label(C)* = *"ServerSoftware"* in an ontology that contains concepts related to computers and networks, it will be converted to *"Server Software"* after applying the *treat_string(label(C))* function. The function also takes care of words separated by the underscore "_" characters.
- After applying the *treat_string()* function to *label(C)* and *label(C')*, check if the two resulting labels are identical, if true, return a similarity measure of *100%*, else proceed to the next step.
- Remove all "Stop words" (such as: "the", "a", "to", ...etc) from label(C) and label(C').
- Retrieve the definitions of every remaining word in the labels from the dictionary. For *label(C)*, concatenate the definition of all the word that make up the label into the string *D*. For *label(C')*, concatenate the definitions of all the words that make up the label into the string *D'*
- Apply the *stemming* algorithm [21] on every word in D and *D'*. On a high level, the algorithm traces words back to their roots, for example it reduces the words "directed" and "directing" to "direct" and "direct" consecutively. This way the two words become comparable to each other.
- Let *len(D)* be the number of words in the string D, and *len(D')* be the number of words in the string *D'*. Let *common_count(D,D')*be the number unique common words between D and *D'*.
- Compute *base_sim(label(C),label(C'))* as: $\frac{common\_count(D,D') \times 2}{len(D)+len(D')}$
- If *base_sim(label(C),label(C'))* $\geq TH$, then add *C'* to *MS(C)* .

4.2 Descendant's Similarity Inheritance (DSI) algorithm

Establishing base similarities between the concepts of the source ontology and the concepts of the target ontology may not be enough to achieve a high degree of precision in relating the two ontologies to each other. Consider this situation for example when aligning ontologies

in the wetland classification domain. The first ontology describes the "Cowardin" wetland classifican system , and the second ontology describes the South African wetland classification system. Figure 7 shows a part of the "Cowardin" system on the left hand side as a source ontology and a part of the South African wetland classification system on the right hand side as a target ontology. When calculating the base similarities between the concepts of the two ontologies, the concept *Reef* which belongs to the *Intertidal* wetland subsystem in the source ontology will yield a base similarity measure of *"100%"* with the concept *Reef* which belongs to the *Intertidal* wetland subsystem in the target ontology. Furthermore, it will also yield a base similarity measure of *"100%"* with the concept *Reef* which belongs to the *Subtidal* wetland subsystem in the target ontology. The later base similarity measure is misleading because it does not correctly express the true meaning of the relationship between the two concepts that should not be related because they belong to different wetland subsystems.

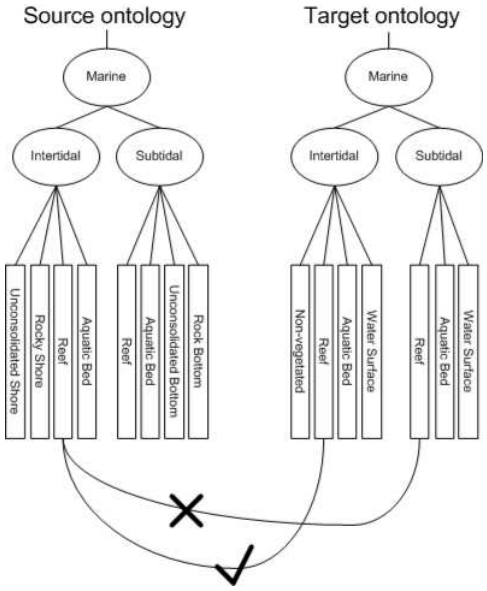


Fig. 7. An example of a case where misleading mappings may occur.

In order to eliminate such situations, we propose the *DSI* algorithm which reconfigures the base similarity between the concepts based on the similarity of their parent concepts. We define the *DSI* reconfigured similarity between a concept C in S and a concept C' in T as *DSI_sim(C,C')*. In what follows we present the details on how to determine *DSI_sim(C,C')*

- Let *path_len_root(C)* be the number of concepts between the concept C in S and the root of the ontological tree S. For example, in Figure 8, *path_len_root(C) = 2*.
- Let *path_len_root(C')* be the number of concepts between the concept C' in T and the root of the ontological tree T. For example, in Figure 8, *path_len_root(C') = 2*.
- Let $parent_i(C)$, be the *ith* concept from the concept C to the root of the source ontology S, where $_i$ is a value between *0* and *path_len_root(C)*. For example, in Figure 8, $parent_1(C)$ is the concept B.
- Let $parent_i(C')$, be the *ith* concept from the concept C' to the root of the source ontology T, where $_i$ is a value between *0* and *path_len_root(C')*. For example, in Figure 8, $parent_1(C')$ is the concept B'.

- We define *DSI_sim(C,C')* as:
  $0.75 \times$ *base_sim(C,C')*$+ \sum_{i=1}^{n}$(*base_sim*(*parent*$_i$(C),*parent*$_i$(C'))$\times \frac{2(n+1-i)}{n(n+1)} \times 0.25$), where
  n = *minimum(path_len_root(C),path_len_root(C'))*.

The main characteristic of the *DSI* algorithm is to allow the parent concepts of a given concept to play a role in the identification process of the concept. The immediate parent of a concept contributes more to the identity of the concept than its grandparent, and the grandparent concept contributes more than the great grandparent, and so on until the root is reached. This can be demonstrated by considering the example in Figure 8. In the figure, we show how the *DSI* similarity is determined between the concept *C* in the source ontology *S* (shown in the left hand side of the figure) and the concept *C'* in the target ontology *T* (shown in the right hand side of the figure) when applying the *DSI* algorithm. The *DSI* similarity is determined by summing 75% of the base similarity between *C* and C' to 17% of the base similarity of their immediate parents (*B* and *B'*) to finally 8% of the base similarity of their grandparents (*A* and *A'*).

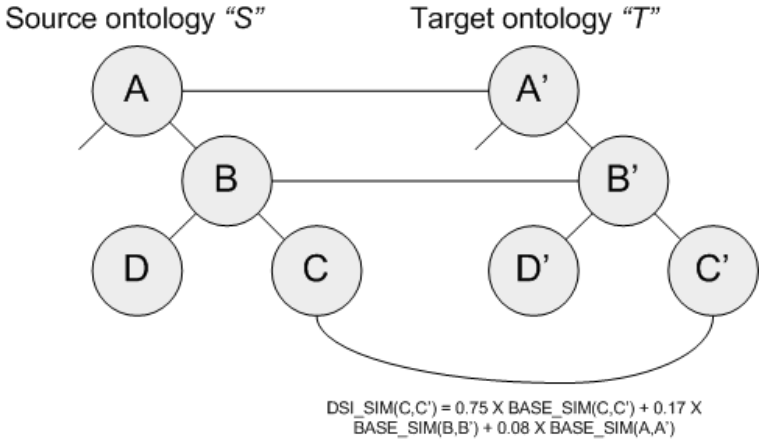


Fig. 8. Applying the DSI algorithm to calculate the similarity between *C* and C'

Considering the case in Figure 7, the base similarity between the concepts *Intertidal* in the source ontology and the concept *Subtidal* in the target ontology is 37%. The base similarity between the concepts *Marine* in the source ontology and the concept *Marine* in the target ontology is 100%. When applying the DSI algorithm, the DSI similarity between the concept *Reef* which belongs to the *Intertidal* wetland subsystem in the source ontology and the concept *Reef* which belongs to the *Subtidal* wetland subsystem in the target ontology will be 88%. Applying the DSI algorithm again between the concept *Reef* which belongs to the *Intertidal* wetland subsystem in the source ontology and the concept *Reef* which belongs to the *Intertidal* wetland subsystem in the target ontology will yield a similarity of 100%. Therefore, we conclude that the later similarity case should be considered when reporting the results. This is one example that shows how the DSI algorithm can be useful in determining more accurate similarity measures between concepts.

In this algorithm, siblings of a concept contribute to the identification of the concept. This may further enhance the quality of the automatic alignment process. Similar to the *DSI* algorithm, the *SSC* algorithm reconfigures the base similarities between concepts. We define the *SSC* reconfigured similarity between a concept C in S and a concept C' in T as *SSC_sim(C,C')*. In what follows we present the details on how to determine this similarity:

- Let *sibling_count(C)* be the number of sibling concepts of the concept C in S. For example, in Figure 9, *sibling_count(C) = 2*.
- Let *sibling_count(C')* be the number of sibling concepts of the concept C' in T. For example, in Figure 9, *sibling_count(C') = 3*.
- Let *SS(C)* be a set of all sibling concepts of a concept C in S
- Let *SS(C')* be a set of all sibling concepts of a concept C' in T
- Let S_i be a the i_th sibling concept of C where $S_i \in SS(C)$, i is a value between *1* and *sibling_count(C)*.
- Let S'_j be a the j_th sibling concept of C' where $S_j \in SS(C')$, j is a value between *1* and *sibling_count(C')*.
- We define *SSC_sim(C,C')* only if both *SS(C)* and *SS(C')* are not empty as: $0.75 \times$ *base_sim(C,C')* $+\frac{0.25}{n} \times \sum_{i=1}^{n} \max(\text{base\_sim}(S_i,S'_1),...,\text{base\_sim}(S_i,S'_m))$, where n =*sibling_count(C) and* m = *sibling_count(C')*.

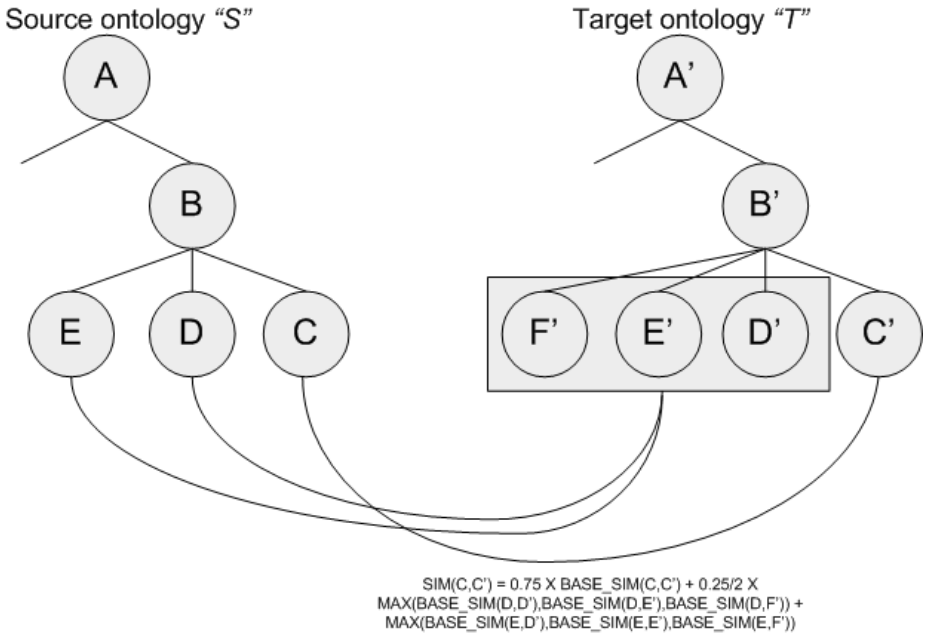


Fig. 9. Applying the SSC algorithm to calculate the similarity between C and C'

The main characteristic of the *SSC* algorithm is allow the siblings of a given concept to play a role in the identification process of the concept equally. This can be demonstrated by considering the example in Figure 9. In the figure, we show how the *SSC* similarity is determined between the concept C in the source ontology S (shown in the left hand

side of the figure) and the concept *C'* in the target ontology *T* (shown in the right hand side of the figure) when applying the *SSC* algorithm. The *SSC* similarity is determined by adding 75% of the base similarity between *C* and C' to 12.5% of the maximum base similarity of *((D,D'),(D,E'), and (D,F'))* to finally 12.5% of the maximum base similarity of *((E,D'),(E,E'), and (E,F'))*.

Similar to Melnik's similarity flooding algorithm described in [28], both our *DSI* and *SSC* algorithms depend on establishing initial similarities between concepts before they can be executed. Unlike Melnik's similarity flooding algorithm, our *DSI* and *SSC* algorithms do not run in multiple iterations that keep reconfiguring the similarities between concepts until the similarities become stable. The penalty paid in running multiple iterations of the flooding algorithm is a possible degradation in the run time performance of the process on the account of obtaining high quality results. In order to confirm this, we are planning to run several test cases in order to compare the run time performance between our algorithms and Melnik's similarity flooding algorithm.

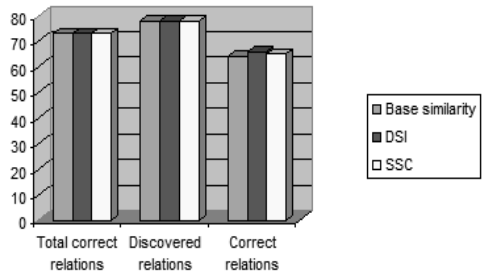
4.4 Evaluation of our Similarity Algorithms

To validate our *base similarity*, *DSI*, and *SSC* algorithms, we present the results of aligning four sets of ontologies provided by the Ontology Alignment Evaluation Initiative (OAEI) [33]. The first set contained two ontologies describing classifications of various weapon types, the second set contained two ontologies describing attributes of people and pets , the third set contained two ontologies describing classifications of computer networks and equipments, and the fourth one contains general information about Russia. Each set contains a source ontology, a target ontology, and the expected alignment results between them. In our validation process, for each set of ontologies, we capture the number of discovered relations between the concepts of the source ontology and the concepts of the target ontologies for each algorithm. Each relationship represents a mapping from a concept *C* in the source ontology *S* to a matching target ontology concept $C' \in MS(C)$ with the highest similarity measure. Obviously, Concepts in the source ontologies with empty mapping sets do not establish any relations with any concepts in the target ontologies. After capturing the discovered relations for each set, we count how many of these relations are valid when compared with the expected alignment results for this set. Having figured the count of correct relations, we calculate the precision and the recall for each test case. The precision is calculated by dividing the number of discovered valid relations to the total number of discovered relations, the recall is calculated by dividing the number of discovered valid relations to the total number of valid relations as provided by the expected alignment results.

Upon aligning the ontologies in the first set, the *DSI* algorithm yielded slightly higher results than the *SSC* which also yielded a slightly higher the the base similarity algorithm as shown in Figure 10. Upon aligning the ontologies of the second set, all three algorithms yielded exact results for recall and precision as shown in Figure 11.

The *SSC* algorithm yielded better recall and precision results than the other two algorithms

Algorithm	Total correct relations	Discovered relations	Correct relations	Precision	Recall
Base similarity	73	78	64	82.05%	87.67%
DSI	73	78	66	84.62%	90.41%
SSC	73	78	65	83.33%	89.04%



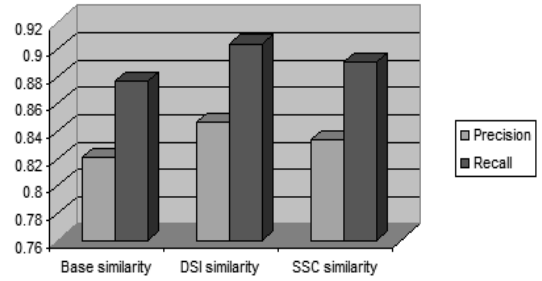
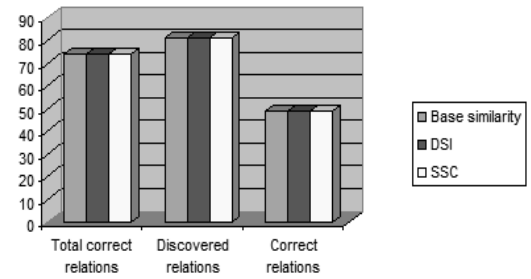


Fig. 10. Applying Base similarity, DSI, and SSC algorithms on ontologies describing weapons

Algorithm	Total correct relations	Discovered relations	Correct relations	Precision	Recall
Base similarity	74	81	49	60.49%	66.22%
DSI	74	81	49	60.49%	66.22%
SSC	74	81	49	60.49%	66.22%



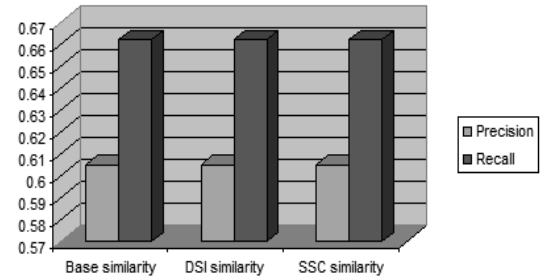
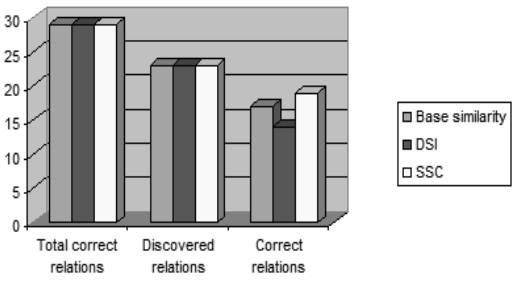


Fig. 11. Applying Base similarity, DSI, and SSC algorithms on ontologies describing people and pets

when aligning the third set of ontologies as shown in Figure 12. Finally as shown in Figure 13, upon aligning the fourth set the *DSI* algorithm yielded the highest results for precision and recall followed by the *SSC* algorithm then the base similarity algorithm.

The differences of the recall and precision values for a given algorithm when applied across different test cases are mainly due to the way the input ontologies are formulated. For example, in the first set and the second set, the relations between the concepts, their parents, and their siblings do not contribute to refining the base similarity results. However, the relationships between the concepts and their siblings added value in refining the base similarity results when aligning the third set. The relationships between the concepts and their parents added value in refining the results when aligning the fourth set. Therefore, the selection of an appropriate matching algorithm should be done after a preliminary examination of the concepts in the ontologies and how they relate to each other. Mochol *et al.* [30] present a methodology on how to select an appropriate matching algorithm for a specific alignment case by having a domain expert fill a questionnaire about the nature of the ontologies to be

Algorithm	Total correct relations	Discovered relations	Correct relations	Precision	Recall
Base similarity	29	23	17	73.91%	58.62%
DSI	29	23	14	60.87%	48.28%
SSC	29	23	19	82.61%	65.52%



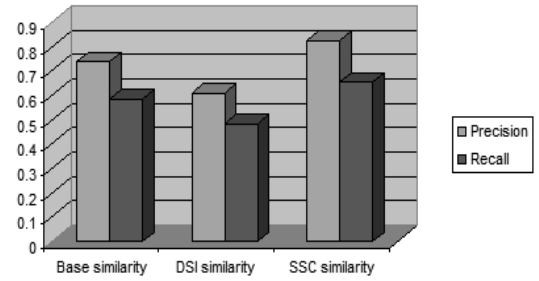
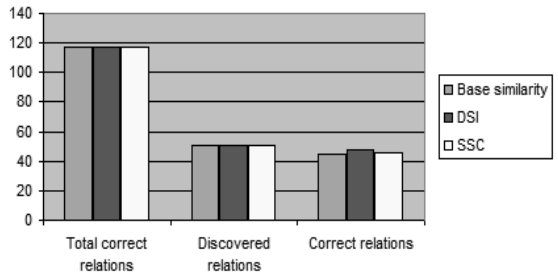


Fig. 12. Applying Base similarity, DSI, and SSC algorithms on ontologies describing computer networks

Algorithm	Total correct relations	Discovered relations	Correct relations	Precision	Recall
Base similarity	117	51	45	88.24%	38.46%
DSI	117	51	48	94.12%	41.03%
SSC	117	51	46	90.20%	39.32%



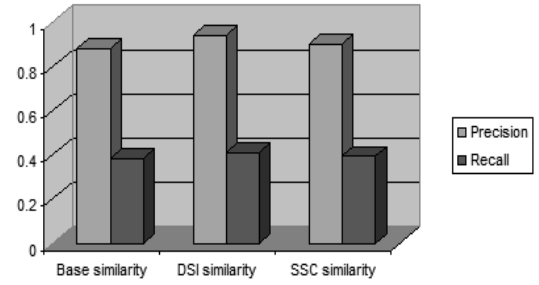


Fig. 13. Applying Base similarity, DSI, and SSC algorithms on ontologies containing general information about Russian

aligned.

5 Future work

Alignment of ontologies is a very hot topic that is pursued by many researchers around the world. The most challenging part of this type of research is the validation of any proposed automatic alignment technique. There are many reasons why the validation process is a grey area for many researchers, in what follows we list some of these reasons:

- There is a lack of real world ontology sets that are good candidates for alignment.
- The evaluation of results from aligning two ontologies may differ from one person to another. The main reason for this is that the concepts can be matched with different

matching criterion, for example, matching the concepts "man" and "woman" is considered valid based on the criterion that both concepts are of the same class "human being" while it may be considered as invalid (bad match) based on the criterion that they are opposite in "sex".

- All the alignment algorithms are semi-automatic, this is mainly because of the previous reason. Ontology Domain experts are always expected to interfere in order to validate the automatically generated mappings, override what they consider "invalid" mappings, and map what was not matched by the automatic alignment process. Different domain experts may end up with different alignment results because they may differ in how they understand or interpret the meanings of of concepts within the context of their ontologies.
- The value of aligning ontologies is still not visible in the real world applications. As discussed in the *Ontology Alignment Evaluation Initiative 2006 Campaign* [1] sessions, the current focus is on exploring more automatic alignment techniques. In the near future, the focus should be shifted to exploring practical ways on how to deploy such techniques in the industry in order to solve real world business problems.

In order to overcome the above obstacles in validating various approaches to ontology alignment, the Ontology Alignment Evaluation Initiative (OAEI) [33] was established as a part of the International Workshop on Ontology Matching which is correlated with the International Semantic Web Conference (ISWC). Every year, the workshop proposes several sets of ontologies to be aligned, each set contains a source ontology and a target ontology. A group of domain experts decide the results of aligning the source and target ontologies in the proposed sets and keep the results hidden from the public to enable for an international contest in aligning these sets. Researchers around the world develop automatic alignment algorithms in order to compete in the OAEI contest, they submit their algorithm in a standardized format to the contest organizers who run them against the proposed sets of ontologies. The results of each algorithm are then evaluated against the expected results that were decided by the experts in order to determine the winners of the contest.

My plan is to compete in the upcoming OAEI contest which will be held as a part of the *Workshop on Ontology Matching* in The *6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference* in Busan, South Korea [2]. The contest will take place in November, 2007. My goal is to obtain reasonably results in the contest in order to prove that my automatic alignment techniques described in Section 4 are capable of solving common ontology alignment problems.

One important point to mention is that the competition does not fully validate my ontology alignment approach described in Section3. The reason is that every competitor is allowed to enter with only one automatic matching algorithm. Therefore, if I introduce more layers in my multi-layered framework, I can only participate with one. Furthermore, any layer that contains an alignment methodology which requires full or partial human intervention cannot be evaluated by the contest. A major part of my research is to propose an extensible semi-automatic ontology alignment framework where different mapping layers complement each other. As depicted from the alignment results in Section 4.4, the input ontologies play a major role in the success of a given alignment algorithm. It is expected that depending

on the nature of the input ontologies, some matching techniques can perform better than others [30]. Therefore there is a necessity of either combing multiple semi-automatic alignment techniques together (as pointed out in [31]) or selectively preferring some techniques over the others to achieve valid and reasonable alignment results. Again, to reiterate, there are many non-deterministic factors in the ontology alignment process such as the nature of the input ontology, the level of heterogeneity of the input ontologies, and the variation of the definition of what is considered a "match" between concepts. All these non-deterministic factors support the need for a semi-automatic multi-layered approach like ours to achieve perfect results in aligning real world ontologies in various domains. This is mainly what distinguishes our approach from the rest of the approaches described in Section 2.

Another major and important challenge which is widely ignored [34] is the enhancement of the runtime performance of automatic alignment techniques. The main focus has been given to obtaining reasonably valid alignment results while little focus has been given to tune the performances of the alignment algorithms. Problems can arise when aligning very large ontologies if the runtime performance tuning of the alignment algorithm is neglected, especially if the algorithm is deployed in production in a highly paced industrial firm that has a small processing window for the algorithm to run. To address this concern, I am planning to propose a methodology that will perform sub-tree matching prior to performing concept level matching. This methodology will significantly reduce the number of comparisons between concepts and enhance the run time of our automatic matching algorithms.

In order to prepare for OAEI competition which will take place in November, 2007. I am planning to do the following:

- Adjust the parameters of my *DSI* and *SSC* alignment algorithms as necessary in order to achieve higher precision and recall values for the test cases I am currently experimenting with.
- Propose and implement a strategy to tune the performance of my alignment algorithms is such a way that will not compromise the quality of the alignment results.
- Run several test cases to compare the run time performance and the quality of alignment results between my alignment algorithms and Melnik's similarity flooding algorithm as mentioned earlier in Subsection 4.3.

Acknowledgments

I would like to thank Prof. Isabel Cruz for being a great advisor, supporter, and academic mentor through out the stages of this research. I would like to also thank Dr. Nancy Wiegand and Steve Ventura, from the Land Information and Computer Graphics Facility at the University of Wisconsin-Madison for the discussions on land use problems. In addition, I would like to thank Sujan Bathala, Mohammad Noman, and Nalin Makar for their active participation in the development of the AgreementMaker.

References

[1] http://oaei.ontologymatching.org/2006.

[2] http://www.iswc07.org.

[3] Y. An, A. Borgida, and J. Mylopoulos. Inferring Complex Semantic Mappings Between Relational Tables and Ontologies from Simple Correspondences. In *OTM Conferences (2)*, pages 1152–1169, 2005.

[4] G. Antoniou and F. van Harmelen. Web Ontology Language: OWL. In *Handbook on Ontologies*, pages 67–92. 2004.

[5] B. Ashpole, M. Ehrig, J. Euzenat, and H. Stuckenschmidt, editors. *Integrating Ontologies '05, Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies, Banff, Canada, October 2, 2005*, volume 156 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[6] B. Ashpole, M. Ehrig, J. Euzenat, and H. Stuckenschmidt, editors. *Integrating Ontologies '05, Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies, Banff, Canada, October 2, 2005*, volume 156 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

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

[8] T. Berners-Lee, D. Connolly, E. Prud'homeaux, and Y. Scharf. Experience with N3 rules. In *Rule Languages for Interoperability*, 2005.

[9] L. M. Cowardin, V. Carter, F. C. Golet, and E. T. LaRoe. Classification of Wetlands and Deepwater Habitats of the United States, 1979.

[10] I. Cruz and A. Rajendran. Exploring a New Approach to the Alignment of Ontologies. In *Semantic Web Technologies for Searching and Retrieving Scientific Data Workshop, International Semantic Web Conference, Sanibel Island, Florida*, 2003.

[11] I. F. Cruz, A. Rajendran, W. Sunna, and N. Wiegand. Handling Semantic Heterogeneities using Declarative Agreements. In *International ACM GIS Symposium*, pages 168–174, 2002.

[12] I. F. Cruz, W. Sunna, and K. Ayloo. Concept Level Matching of Geospatial Ontologies. In *GIS Planet Second Conference and Exhibition on Geographic Information, Estroil, Portugal*, 2005.

[13] I. F. Cruz, W. Sunna, and A. Chaudhry. Semi-Automatic Ontology Alignment for Geospatial Data Integration. In *Proc. 3rd Int. Conf. on Geographic Information Science (GIScience), LNCS 3234, Springer Verlag*, pages 51–66, Adelphi, MD, 2004.

[14] I. F. Cruz, W. Sunna, N. Makar, and S. Bathala. A Visual Tool for Ontology Alignment to Enable Geospatial Interoperability. *Journal of Visual Languages and Computing. (Coming soon)*, 2007.

[15] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a Search and Metadata Engine for the Semantic Web. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659, New York, NY, USA, 2004. ACM Press.

[16] J. Dini, G. Gowan, and P. Goodman. South African National Wetland Inventory., 1998.

[17] J. Euzenat, P. Guégan, and P. Valtchev. Ola in the Oaei 2005 Alignment Contest. In *Integrating Ontologies*, 2005.

[18] J. Euzenat and P. Valtchev. Similarity-Based Ontology Alignment in Owl-Lite. In *ECAI*, pages 333–337, 2004.

[19] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu. The Evolution of Protégé: an Environment for Knowledge-Based Systems Development. *Int. J. Hum.-Comput. Stud.*, 58(1):89–123, 2003.

[20] M. A. Hernández, R. J. Miller, and L. M. Haas. Clio: A Semi-Automatic Tool For Schema Mapping. In *SIGMOD Conference*, 2001.

[21] D. A. Hull. Stemming Algorithms: A Case Study for Detailed Evaluation. *Journal of the American Society of Information Science*, 47(1):70–84, 1996.

[22] N. Jian, W. Hu, G. Cheng, and Y. Qu. FalconAO: Aligning Ontologies with Falcon. In *Integrating Ontologies*, 2005.

[23] V. Lopez, E. Motta, and V. Uren. *PowerAqua: Fishing the Semantic Web*. 2006.

[24] V. Lopez, M. Sabou, and E. Motta. Powermap: Mapping the Real Semantic Web on the Fly. In *International Semantic Web Conference*, pages 414–427, 2006.

[25] B. McBride. Jena: A Semantic Web Toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002.

[26] B. McBride. The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS. In *Handbook on Ontologies*, pages 51–66. 2004.

[27] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An Environment for Merging and Testing Large Ontologies. In *Seventeenth International Conference on Principles of Knowledge Representation and Reasoning (KR-2000)*, pages 483–493, 2000.

[28] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE.02)*, 2002.

[29] G. A. Miller. Wordnet: An Online Lexical Database. Technical report, Prenceton University, 1990.

[30] M. Mochol, A. Jentzsch, and J. Euzenat. Applying an Analytic Method for Matching Approach Selection. In *International Workshop on Ontology Matching (OM-2006) collocated with the 5th International Semantic Web Conference (ISWC-2006), Athens, Georgia, USA*, 2006.

[31] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB J.*, 10(4):334–350, 2001.

[32] M. A. Rodríguez and M. J. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Trans. Knowl. Data Eng.*, 15(2):442–456, 2003.

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

[34] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang. Using Bayesian Decision for Ontology Mapping. *Web Semant.*, 4(4):243–262, 2006.

[35] C. T. Yu and W. Meng. *Principles of database query processing for advanced applications.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.