Finding Ontological Correspondences for a Domain-Independent Natural Language Dialog Agent

Hamid Haidarian Shahri, Donald Perlis

Department of Computer Science
University of Maryland, College Park, USA
{hamid, perlis}@cs.umd.edu

Abstract

Currently, our interactions with devices are constrained, as we need to program/configure devices, primarily through some artificial interface, instead of interacting through a dialog agent. This limitation in human-device interactions is a major obstacle to the integration of devices (e.g. PDA, GPS) in our daily activities. Considering the numerous advantages of using a dialog agent for communicating with devices, we are building a cognitive dialog agent that facilitates this communication. The agent has semantic knowledge about concepts and available commands of a device, and this knowledge is represented using an ontology. The agent gathers the necessary information, for execution of commands, by engaging users in a meaningful communication. Finally, the commands are sent to the device for execution. Consequently, the agent relieves users from the burden of acquiring the knowledge, which is necessary for operating the device. In order for the agent to process the human utterance, and make sense of the concepts that the human relates to, it requires a mapping between the concepts that the human is using, and the concepts that are understandable for the device. In this paper, we provide a novel algorithm to determine these matching concepts. The algorithm exploits various similarity metrics to disambiguate and match concepts. We formally define these metrics, and analyze the time complexity of computing the metrics, algorithmically. In addition, the effectiveness and scalability of the metrics are evaluated empirically, on real-world ontologies, through different experiments. In essence, the dialog agent interweaves the individual threads of meaning between human and device, using the ontology mapping algorithm, and prevents miscommunication between them.

1. Introduction

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 an *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 tremendous advantages of communicating with devices through a flexible agent, instead of learning the exact operation of a device by human, we 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. 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.

In this paper, we provide this essential matching *algorithm* for the dialog agent, and formally define the various concept similarity metrics that we have used, for the matching process. The metrics are, namely, lexical similarity, extensional similarity, extensional closure similarity, and global path similarity. Additionally, the time complexity of computing each metric is analyzed, algorithmically. Finally, the *effectiveness* and *scalability* of these metrics are evaluated empirically, on real-world ontologies, 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. We will elaborate on this issue in the next section.

The principal contributions of our work are the following. Most 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, whereas our work involves *semantics of dialog* as used by humans (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, 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 concepts that humans employ in their utterance. Furthermore, 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 system. There are a few application-oriented dialog systems that interact with humans, e.g. [Bob77, 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.

The rest of the paper is organized as follows. In Section 2, we provide an overview of the architecture of the system. Section 3 is the required preliminaries to formalize our approach for ontology mapping. In Section 4, we propose the various similarity metrics that we have used, for finding the matching concepts in different ontologies, and then provide an algorithm that utilizes these metrics for concept matching. Section 5 contains the empirical evaluations for finding the matching concepts. Section 6 is the related work, and Section 7 is the conclusions.

2. Overview of the Architecture

Our design for the dialog agent, described in this paper, 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. A detailed discussion of the above issues is *out of the scope* of this paper, since this paper focuses on the *ontology matching* algorithm, and various concept similarity metrics. However, 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. This paper provides 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. One of the domains and experimental test beds that we are currently using is a Mars rover application. The general architecture of the system is depicted in Figure 1. 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 1). 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* to keep track of user's utterances and requests (see Figure 1). The dialog agent starts with a simple *ontology*. It may augment the ontology with various terms, as the conversation proceeds, since the

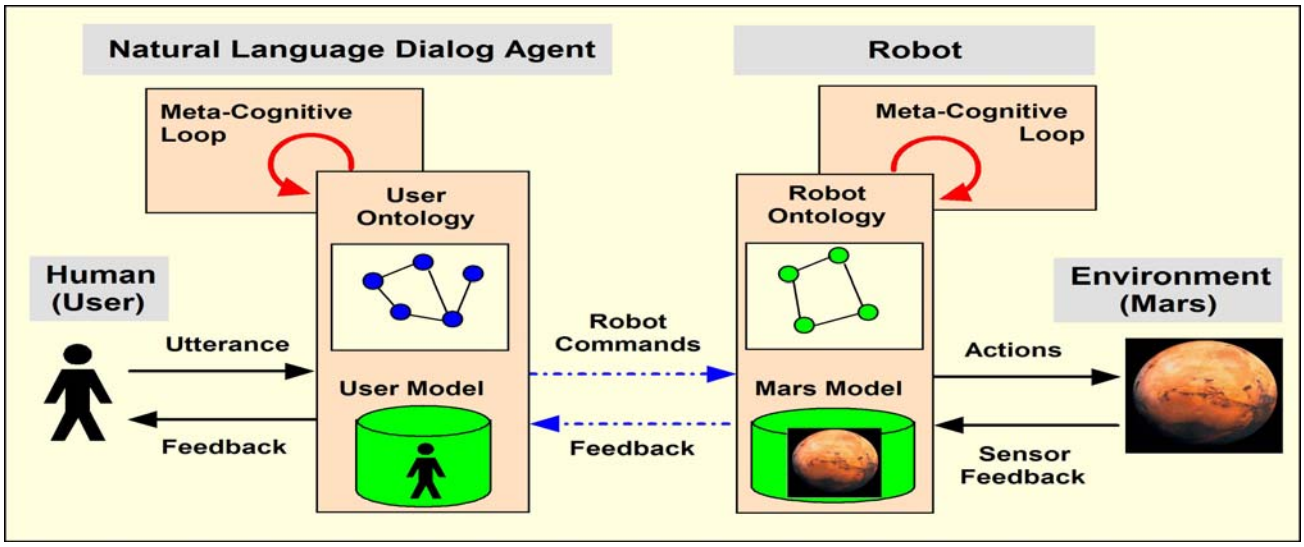

Fig. 1. Overview of the architecture, showing the communication between human user and robot, via a dialog agent.

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 the knowledge, which is necessary for operating the robot (device).

Since the dialog agent is domain-independent, it can be used for interacting with any domain. Also, the ontologies are specified using the OWL or RDF language, which can model any domain. For example, in the online shopping and e-commerce domain, consider that a human user specifies through conversation with the dialog agent that he 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, the term "**journal**" may be used instead of "magazine," in some other ontology. 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, which finds similar concepts in different ontologies, should exploit various concept similarity metrics. 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.

3. Formalization Preliminaries

In this section, we provide the formal definitions for the terminology used in ontology mapping. This formalization is necessary for self containment and a thorough treatment of the topic. More specifically, the formalization will be required for defining various concept similarity metrics, in Section 4.

The definitions are primarily based on RDF (Resource Description Framework), which is a framework and W3C recommendation for representing ontologies 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. By using this standard, the components of our system can interact easily, and we can also leverage existing tools for creation and manipulation of ontologies. Moreover, OWL (Web Ontology Language) is based on RDF. Hence, our 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 things 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 things denoted by 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 (Concept and Instance): *Resources may be divided into groups called concepts (or classes). The members of a concept are known as instances (or individuals) of the concept. Associated with each concept is a set, called the extension of the concept, which is the set of the instances of the concept.*

Example: The concept "Book" could have various instances, which are the titles of books, e.g. "The Art of Computer Programming." The concept "Brand Name" could have different instances, e.g. "Nike," "Adidas," and "Reebok."

Concepts 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 concept. RDF distinguishes between a concept and the set of its instances. If a concept *C* is a subclass of a concept *C′*, then all instances of *C* will also be instances of *C′*. The rdfs:subClassOf property may be used to state that one concept is a subclass of another. The term superclass is used as the inverse of subclass. If a concept *C′* is a superclass of a concept *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 concepts. The instances of a concept 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.*

4. Concept Matching for Communication

In this section, we provide an algorithm for concept matching (i.e. finding corresponding concepts), in ontologies. The algorithm utilizes various concept similarity metrics. We categorize these metrics, formally define each of them, and analyze the time complexity of computing the metric, which is necessary for scalability, in real world settings. Additionally, the role of the reasoner of an ontology for concept matching is explained.

Definition 8 (Mapping for communication): *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 unordered pairs of elements of sets C_1 and C_2, that is:* $C_1 \otimes C_2 = \{(a,b) \mid a \in C_1, b \in C_2, a \neq b\}$.

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. Similarity value is the sum of the following four similarity metrics: lexical, extensional, extensional closure, and global path, which are computed as follows.*

In real world applications, the issue of setting the *threshold*, for identifying corresponding concepts, is an important one. Since identifying correspondences inherently involves *uncertainty*, no algorithm can achieve one hundred percent precision. Fortunately, the dialog agent has access to the human user and can consult with the user and employ his judgment, to make the final decision about whether two concepts correspond or not. However, if we allow approximate answers (similar to web search engine results which are not always relevant), a threshold can be set automatically (using machine learning techniques) or semi-automatically (with a human user involvement).

In principle, ontologies can cover any domain of knowledge and the *nature of data instances* is extremely diverse in real world applications. Hence, it is difficult to provide general guidelines on how to set the threshold for all ontologies (i.e. applications). Essentially, they need to be determined experimentally, for each application.

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.*

Theorem 1 (Lexical Similarity Complexity): *The 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.*

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*

```
Concept-Matching Algorithm
Input:
   O1, O2: Original ontologies
   C1: Set of concepts in O1
   C2: Set of concepts in O2
Output:
   M: Set of matching concept pairs (c1, c2), s.t. c1∈C1, c2∈C2

1.   for all c1∈C1, c2∈C2
2.     lexSim ← lexicalSim(c1.name, c2.name)
3.     c1.ex ← reasoner.Extensions(c1)
4.     c2.ex ← reasoner.Extensions(c2)
5.     extSim ← extensionalSim(c1.ex, c2.ex)
6.     c1.all ← reasoner.AllExtensions(c1)
7.     c2.all ← reasoner.AllExtensions(c2)
8.     extCSim ← extensionalClosureSim(c1.all, c2.all)
9.     c1.path ← reasoner.GlobalPath(c1)
10.    c2.path ← reasoner.GlobalPath(c2)
11.    gpSim ← globalPathSim(c1.path, c2.path)
12.    if ( lexSim+extSim+extCSim+gpSim > threshold ) then M ← M ∪ (c1, c2)
13.  end for
14.  return M
```

similarity metric for s and t is computed as $|e(s) \cap e(t)| / |e(s) \cup e(t)|$.

Theorem 2 (Extensional Similarity Complexity): *The complexity of computing the extensional similarity metric for concepts s and t is O(\|e(s)\|.\|e(t)\|). The set of individuals e is computed using the reasoner.*

Definition 12 (Extensional Closure Similarity Metric): *Let* $s \in C_1$ *and* $t \in C_2$ *be two classes in the ontologies.Class x is a subclass of class y, is denoted as* $x \sqsubseteq y$. *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* $|e_c(s) \cap e_c(t)| / |e_c(s) \cup e_c(t)|$.

Theorem 3 (Extensional Closure Similarity Complexity): *The complexity of computing the extensional closure similarity metric for concepts s and t is O(\|e_c(s)\|.\|e_c(t)\|). The subclasses of a concept and their respective individuals are computed using the reasoner.*

Based on Definitions 11 and 12, instances of two concepts in different ontologies, need to be matched, i.e. duplicate instances should be identified. When computing the extensional and extensional closure similarity metrics, duplicate instances must be detected, in order to compute the intersection of instances of two concepts correctly. One approach to identifying duplicate instances is the use of approximate *string matching* techniques for the name of instances, similar to the lexical similarity metric used for concepts (Definition 10).

There is an interesting analogy between instance matching in ontologies, and the problem of *approximate duplicate record elimination* in data cleaning in databases. A recent survey on this diverse problem is [Elm07]. Some of the approaches developed for data cleaning in databases are also applicable for finding duplicate instances, when computing the extensional similarity metrics for concept matching.

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 which starts from the root and ends at s, in the RDF graph. 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 which appear in the two paths.*

Theorem 4 (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 concept hierarchy in the worst case), then the complexity of computing the global path similarity metric for concepts s and t is O(||p(s)||.||p(t)||). The concept hierarchy is created from the subclass relationships, using the reasoner.*

Concept Matching Algorithm: Based on the previous definitions, we present our concept matching algorithm, above. The *Concept-Matching* algorithm exploits the concept similarity metrics introduced in this section. 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 is based on the idea of similarity value and threshold, introduced in Definition 9.

5. Empirical Evaluations

In order to evaluate the effectiveness of the concept similarity metrics, introduced in Section 4, we

implemented these metrics in the ontology mapping module of our dialog agent. The running time of computing the similarity metrics was also measured in different experiments, which is important for the scalability of the approach, when using large ontologies. In our implementation, Pellet was used for ontology reasoning [Sir07]. Pellet is an open source reasoner written in Java. Details, of how ontology reasoning is used in the concept matching process was described in Section 4. The experiments were run on a 1.86 GHz Pentium machine with 512 MB of RAM.

The results reported here are from two real-world ontologies that have been developed separately, by different organizations. 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 terms used in bibliographic items and a university organization. The other ontology is from INRIA [Inr], and has been designed by Antoine Zimmermann 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 contain 24 corresponding concepts. Table 1 shows the characteristics of the ontologies with more details.

Table 1. Detailed characteristics of the ontologies.

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

Evaluation Metrics: The following evaluation metrics are used in this section. *Recall* (*R*) is the number of correct results divided by the number of results that should have been returned. *False Positive* (*FP*) error is the ratio of the number of wrongly identified matches to the total number of identified matches. *Precision* (*P*) is the number of correct results divided by the number of all returned results. Assuming *S* is the set containing all the correct corresponding concepts, and *A* is the set containing the corresponding concepts returned by an algorithm, then recall and precision are computed as: $R = |S \cap A| / |S|$ and $P = |S \cap A| / |A|$. Obviously, the system is performing better, if it has a higher precision, at a given recall rate. 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.

When computing the lexical similarity metric, for comparing the name of concepts in two ontologies, various string similarity measures can be used. We implemented the Jaro-Winkler, Jaccard, Monge-Elkan and Levenstein measures. The results show that the performance of these measures varies considerably, as illustrated in Figure 2. The Jaro-Winkler measure shows a more robust behavior for finding corresponding concepts in ontologies, based on concept name. Precisions below the 60 percent range would not acceptable for many applications, as many of the detected matches would be incorrect. 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.

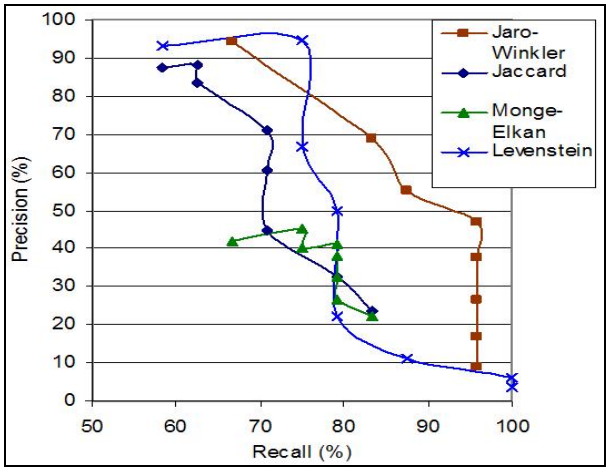

Fig. 2. Performance of various string similarity measure for finding corresponding concepts in ontologies, based on name.

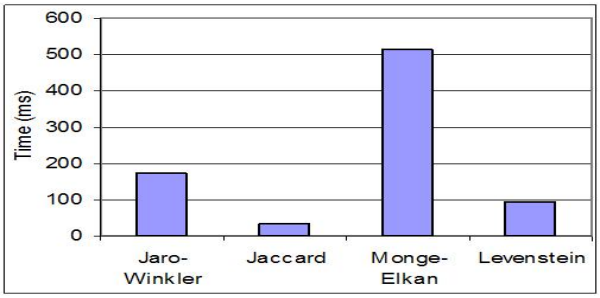

Fig. 3. Running time of computing the lexical similarity of concepts using various string similarity measures.

Figure 3 shows the running time required of computing the lexical similarity of concepts in both ontologies using various string similarity measure. Jaro-Winkler, which showed best performance in the Figure 2, takes 172 ms to compute, and lies approximately in between the other string similarity measure, in terms of running time.

Decreasing the detection threshold for lexical similarity metric would decrease the precision and increase the number of false positives (FP). To tackle this problem and find more matching concepts (without decreasing the threshold), we also employed the other similarity metrics namely, extensional, extensional closure and global path similarity metrics. Our experiments show that utilizing these additional metrics, help in finding more correct matching concepts (true positives). At the same time, they do not introduce many false positives, similar to the case of decreasing the detection threshold, which reduces precision (as in Figure 2).

Computing the lexical similarity of concepts does not require reasoning. However, to compute the extensional, extensional closure and global path similarity metrics, the reasoner must be used. For computing the extensional closure, we need 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 concepts 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, involving comparison of retrieved instances or comparing the concepts in a path, is relatively small. As shown in Figure 4, the running time for computing the lexical similarity is small, compared to the other three similarity metrics, which are approximately the same and require reasoning.

Figure 5 shows that by using the extensional, extensional closure and global path similarity metrics, in addition to lexical, the recall rate increases. This is while the precision remains almost the same. This in turn causes the F1 quality measure to increase, as shown with the darker (purple) bar in Figure 5. It proves that using the additional concept similarity metrics, indeed helps in finding more corresponding concepts and achieving better results.

Returning to the example from the online shopping and e-commerce domain, as explained in Section 2, the dialog agent is using the concept "magazine" in its ontology. When the dialog agent searches the ontology of different vendors on the Web, to find the requested item, the concept "journal" may be used instead of "magazine," in some other ontology. Hence, the dialog agent needs to match the two concepts using any available similarity metric, to facilitate the *communication* between the human user and different vendors on the Web. Different concept similarity metrics like lexical, extensional, extensional closure and global path, serve as evidence for the similarity of the "magazine" and "journal" concepts.

6. Related Work

[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. Most of the work on agent dialog only considers communication between software agents, 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.

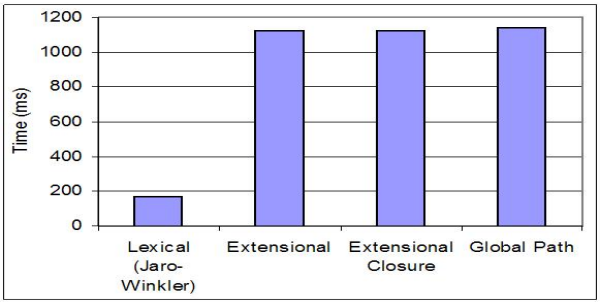

Fig. 4. Running time for computing lexical, extensional, extensional closure and global path similarity metrics.

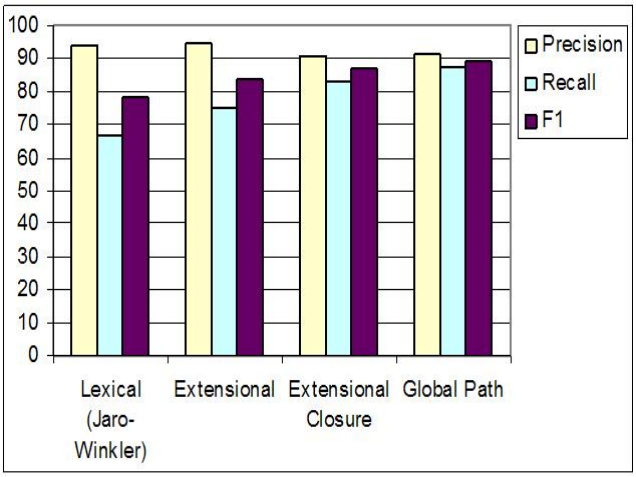

Fig. 5. Using extensional, extensional closure and global path similarity metrics, in addition to lexical, increases the recall and F1 quality measure.

Current research on ontology mapping is mainly in the context of creating new ontologies (ontology engineering), in the Semantic Web community [McG00, Noy00, Stu01]. These ontology engineering efforts produce a *merged* ontology, as the final output of the ontology mapping process. The merging is usually guided

by the ontology designer. Notice that in our approach, when mapping between ontologies, to facilitate communication between the dialog agent and the device, there is no merging of ontologies involved. The ontologies are kept separately, and they are managed by different components of the system. Our algorithm finds the corresponding concepts between ontologies, to allow a more robust interaction between human and device. Also, the work on ontology merging does not deal with a dialog agent, or a human. However, some of the approaches used for ontology merging could potentially help in finding corresponding concepts in ontologies. [McG00] proposes Chimaera, which is one of the early ontology merging tools and supports expressive rule languages. They consider lexical similarity and also structures such as subclass and superclass relations and slot attachments. [Noy00] proposes the PROMPT Suite, which provides tool support for merging ontologies and exploits the graph structure of ontologies. [Stu01] uses the set of shared instances or the set of shared documents annotated with concepts of two ontologies and generates a lattice, to relate the concepts of the ontologies using formal concept analysis.

7. Conclusions

The quality and robustness of human-device interactions is greatly improved, if the human negotiates with the device through a dialog, instead of learning about the commands that are available in the device. To facilitate this interaction, we are building a cognitive dialog agent that engages the human user, in a dialog. The agent has semantic knowledge about the operation of the device, and gathers the required information from the user, before issuing commands to the device for execution.

In order to for the dialog agent to process user's utterances and disambiguate different concepts, it is essential for the agent to create a mapping between the concepts used by the agent and the concepts used in the device ontology. We provide a novel algorithm to compute the mapping of concepts. The algorithm utilizes various similarity metrics to determine the corresponding concepts in different ontologies. The similarity metrics are formally defined and their running time complexity is analyzed algorithmically, for scalability. Furthermore, the effectiveness and scalability of these metrics are evaluated experimentally, on real-world ontologies.

Acknowledgements: We thank James Hendler, Tim Oates, Michael Anderson, Darsana Josyula, Scott Fults, Shomir Wilson, and Matthew Schmill for their invaluable comments.

References

[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.

[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.

[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.

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

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

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

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

[McG00] McGuinness, D.L., 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.

[Noy00] Noy, N.F, 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.

[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.

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

[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.

[Sir07] Sirin E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., Katz, Y., "Pellet: A practical OWL-DL reasoner," *Journal of Web Semantics* (To Appear), 2008.

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

[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.