The Use of Machine-Generated Ontologies in Dynamic Information Seeking

Giovanni Modica[1], Avigdor Gal[2], and Hasan M. Jamil[1]

[1] Mississippi State University, Mississippi State University MS 39762, USA
{gmodica, jamil}@cs.msstate.edu,
WWW home page: http://www.cs.msstate.edu/~{gmodica, jamil}

[2] Technion, Israel Institute of Technology, Technion City, Haifa 32000, Israel
and Rutgers University, Piscataway, New Jersey 08854, USA
avigal@ie.technion.ac.il
WWW home page: http://ie.technion.ac.il/~avigal

Abstract. *Information seeking* is the process in which human beings recourse to information resources in order to increase their level of knowledge with respect to their goals. In this paper we offer a methodology for automating the evolution of ontologies and share the results of our experiments in supporting a user in seeking information using interactive systems. The main conclusion of our experiments is that if one narrows down the scope of the domain, ontologies can be extracted with a very high level of precision (more than 90% in some cases). The paper is a step in providing theoretical, as well as practical, foundation for automatic ontology generation. It is our belief that such a process would allow the creation of flexible tools to manage metadata, either as an aid to a designer or as an independent system ("smart agent") for time critical missions.

1 Introduction and Motivation

Information seeking is the process in which human beings recourse to information resources in order to increase their level of knowledge with respect to their goals. Dating years back, information seeking has affected the way modern libraries operate (using tools such as catalogs, classifications, and indexing) and perpetrated the World Wide Web in the form of search engines. While the basic concept of information seeking remains unchanged, a growing need of automation of the process has called for innovative tools to propagate some of the tasks involved in information seeking to the machine level. Therefore, databases are widely used for the efficient storage and retrieval of information. Also, techniques from the area of Information Retrieval [21] were refined over the years to predict the relevance of information to a person's needs and to identify appropriate information for a person to interact with. Finally, the use of computer-based ontologies [25] was suggested to classify the available information based on some natural classification scheme that would allow a more focused information seeking.[1]

[1] For example, ontology.org is an independent industry and research forum, formed in 1998, that is focused upon the application of ontologies in Internet commerce.

Most Internet portals (including Yahoo! and OpenDirectory) use "cybrerians" to maintain Internet Directories. Common practice nowadays assume that once ontologies are created, computer-supported tools can utilize them as part of the information seeking process. The next natural step in then to let the machine generate the ontologies. One may consider two incentives in doing so. The first is rooted in the initial creation of ontologies, which is a tedious, time-consuming process. The second incentive is rooted in the rapid evolution of ontologies. If ontologies are managed manually, any change to them requires human intervention. This can bring to a halt an electronic process in the absence of constant human support. While the latter attracted a little attention in previous years, it has become a major sticking point with the introduction of eCommerce and electronic exchange markets, a rapidly changing environment in which virtual manufacturers, retailers, and consumers join in to perform activities in cyberspace.

It is the evolution of ontologies which we offer to automate. In particular, we suggest utilizing ontologies in supporting a user in seeking information using interactive systems. As an example, consider a researcher who is interested in renting a car to attend her favorite conference (*e.g.*, CoopIS) in her favorite city (say Trento, Italy). Using Web services, the researcher attempts at comparing available rates from many different car rental companies, in order to reach an educated decision in obtaining her goal. Alas, this process of information seeking is tedious as well as frustrating. Information has to be typed in over and over again, and in most cases a manual comparison of terms and conditions is needed in evaluating the outcomes. An alternative exists in the form of car rental portals (*e.g.*, Travelocity.com). However, as most general-purpose tools, such portals cater to popular needs, and therefore may only offer a limited set of options (such as the cheapest car available). Therefore, if our researcher is interested in a deal which offers no mileage constraints (since our researcher will attend another conference, say in Rome) and no constraints on cross-border usage (for the purpose of vacating at Switzerland once the conference is over), she has to resort to manual search of terms and conditions.

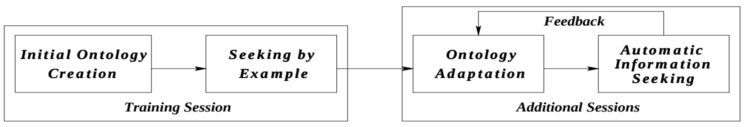


Fig. 1. The process of ontology adaptation

Figure 1 outlines the various stages of ontology creation and adaptation, as suggested in this paper. An initial ontology is created, using either extraction tools or an existing ontology. Equipped with the ontology, the user performs an information seeking session, in which the machine captures the inserted data and matches it with the ontology. This step is followed by an iterative process, in which new information seeking sessions are performed automatically. Each such session requires the fine tuning of the available ontology to the one currently in use, to be followed by an automated information seeking. The results become available to the user and additional feedback is used to enhance the existing ontology and to improve the system's capabilities for the next session. We term the initial session a *training session*, since this is the session in which the machine learns the data needs of the user. There is also a continuous learning process, which enables the machine to improve the ontology with each additional session.

We shall use as a running case study the Web sites of three car rental companies, namely Avis, Hertz and Alamo. We shall use the former as the training Web site for creating the initial ontology and the latter to perform ontology adaptation. In our experiments we extract ontologies from HTML documents. We recognize the fact that XML may serve as a better candidate for ontology exploration. In fact, while one can **exarch** terms and structure from XML documents, one has to **mine** for ontologies in HTML documents. However, current trends in deploying XML as part of the organization data management scheme suggest that while XML may be used for B2B communication, and to some extent as a storage mechanism, interactive sessions still use HTML. Therefore, it is possible that XML data on the server side is "translated" into HTML before being shipped out to the client. It is also worth noting that once an ontology is extracted (from either XML or HTML documents), the process of ontology adaptation remains unchanged.

1.1 Related Work

The problem we tackle in this paper falls into the category of *semantic heterogeneity*, which is well documented in the literature. The area of information science has an extensive body of literature and practice on ontology construction using tools such as thesauri and on terminology rationalization and matching of different ontologies [3, 22, 26, 28]. In the area of databases and information systems many models were suggested to support the process of *semantic reconciliation*, including the SIMS project [4], SCOPES [18], dynamic classificational ontologies [14], COIN [16], and CoopWARE [13], to name a few. What is common to these solutions is their reliance on the designer's intervention, rather than supporting a fully automatic semantic reconciliation.[2] However, redesign and re-implementation of metadata can incur tremendous cost. Therefore, automatic reconciliation becomes a must in such an a environment.

Database research has extensive literature on data integration, including [8], [2], and [15], yet there is little impact of this research on the state-of-the-art in commercial systems. We believe this chasm can be attributed to the fact that most of these approaches rely on semantic reconciliation to be resolved first (probably manually), before attending to the more "technical" aspects of the integration. However, researchers and practitioners alike are coming to realize that there can be no solution to the delivery of integrated information unless one tackles head-on the semantic heterogeneity problem [19]. This research works towards this goal.

Some research was devoted to automatic schema analysis and integration (*e.g.*, [23], [17], and [9]). In [23], the analysis is based on a hand-crafted attribute hierarchy, which we avoid. The work of [9] and [17] are similar in that they analyze a schema,

[2] The slogan of OpenDirectory, "Humans do it better," reflects this approach.

given in an abstract form of a graph, using formal methods of graph analysis. The tools and methodologies suggested in [9], when applied to schema integration are "not sufficient and must be enriched with semantic consideration, such as the interpretation of terms within an application domain in order to correctly compare elements." Our experiments, as shown in this paper, show that it is possible to automatically (and correctly) derive matchings, without reverting to manual interpretation. In [17], it is shown that the process of finding an optimal typing for semi-structured data is NP-hard. Therefore, a method is presented based on heuristics to approximately type a large collection of semi-structured data. No extension of the method to deal with schemata matching is given in [17].

Like many before us, we attempt to perform semantic reconciliation using syntactic comparisons. However, we also enhance our model to include a measure of accuracy, which becomes a powerful tool whenever automated reasoning is involved. The provision of a measure of accuracy allows a user to determine her own tolerance to imprecision and to instruct the system to request for help once imprecision becomes too great. As our experiments demonstrate, if one narrows down the scope of the domain, ontologies can be extracted with a very high level of accuracy.

The rest of the paper is organized as follows. Section 2 outlines our methodology in creating ontologies from dynamic Web pages. Ontology adaptation is discussed in Section 3. Section 4 provides a brief overview of the system's architecture. Finally, some experiment results and future research directions are discussed in Section 5.

2 Ontology Creation

In this paper we develop a methodology for ontology creation in which the user plays an important role, yet with minimum effort. We hope to make this methodology into a fully automated model for ontology creation at a later stage. The methodology involves two phases, namely the *training phase* and the *adaptation phase*. In the training phase we build an initial ontology in which a user's data needs are encoded. The adaptation phase involve an iterative mergence of closely related ontologies with the current ontology at hand. During each iteration, the current ontology is refined and generalized. The refined ontology at any stage can be used to query the Web sites and gather information in a seamless way.

This section provides an overview of the process of ontology creation. Section 2.1 presents the underlying ontological structures we have utilized in this process, based on [6, 7]. The creation of the initial ontology is given in Section 2.2.

2.1 The Ontological Structures

In our ontological analysis we have used the work of Bunge [6, 7]. We have adopted a conceptual modeling approach rather than a knowledge representation approach (in the AI sense). While the latter requires a complete reflection of the modeled reality for an unspecified intelligent task, to be performed by a computerized system in the future [5], the former requires the minimal set of structures to perform a given task (in our case, assisting a user in handling dynamic information seeking). Therefore, we have chosen a subset of the ontological constructs provided by Bunge for our work and have added a new construct, we term *precedence*, for posing temporal constraints. We recognize the limited capabilities of HTML (and to that effect, also XML) to represent rich ontological constructs, and therefore we have had to eliminate many important constructs (*e.g.*, the class structure), simply because they cannot be realistically extracted from the content of Web pages.

We shall now provide a brief description of the ontological constructs to be used in this paper.

Terms (things): We extract a set of terms[3] from a Web page, each term is associated with one or more form entries. The term is taken from the labeling of the entry and the match is done based on the proximity of the label and the entry. For example, some of the terms we have extracted from the Avis reservation page are `Pick-Up Location Code, Pick-Up Date, Pick-Up Time, Return Date, Return Time,` and `Return Location Code`. The usefulness of this construct is further exemplified in Section 3.

Values: Based on Bunge [6], an attribute is a mapping of things and value-sets into specific statements. Therefore, we can consider a combination of a label and its associated data entry (value) to be an attribute. In certain cases, the value-sets to be associated with a term are constrained using drop lists, check boxes, and radio buttons. For example, the term `Pick-Up Date` is associated with two value-sets, the first is $\{Day, 1, 2, \ldots, 31\}$ and the other is $\{January, February, \ldots, December\}$. Clearly, the former is associated with the date of the month and the latter is associated with the month. Whenever constrained value-sets are available, we can enhance our knowledge of the domain, since such constraints become valuable when comparing two terms that do not exactly match through their labels. For example, the term being used by Alamo for `Return Date` is `Dropoff Date`. Although the terms do not match, and the terms `Return` and `Dropoff` are not synonyms (dropoff is not even considered a word in English, according to Oxford dictionary [1]), our algorithm was able to match these terms using the value-sets, since the term `Dropoff Date` has a value-set of $\{(Select), 1, 2, \ldots, 31\}$.

It is our belief that designers would prefer constraining field domains as much as possible, to minimize the effort in writing exception modules. Therefore, it is unlikely that a field with a dropdown list in one form will be designed as a text field in another form. In the case of a small-sized domain, alternative designs may exist (*e.g.*, AM/PM may be represented as either a dropdown list or radio buttons). Since our algorithm extracts domains and represent them in a unified manner, the end result will remain the same, whether the designer use dropdown list or radio buttons.

Composition: We differentiate simple terms from composite terms. A composite term is composed of other terms (either simple or composite). In the Avis reservation Web page, all of the terms mentioned above are grouped under `Rental`

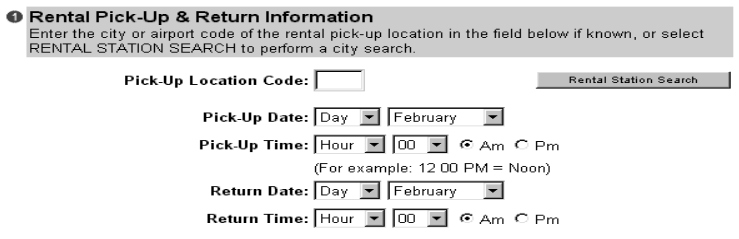
[3] The choice of words to describe ontological structures in Bunge's work had to be general enough to cover any application. We feel that the use of *thing*, which may be reasonable in a general framework, can be replaced with a more concrete description in this application.

Pick-Up & Return Information. It is worth noting that these terms are, in themselves, composite terms. For example, Pick-Up Time is a group of three entries, one for the hour, the other for the minutes, and the third for either AM or PM. In this case, since the entries themselves are nameless, we assign the terms within this group using the group name (*e.g.*, Pick-Up Time 1, Pick-Up Time 2, etc.) Another composite term in the same Web page is titled Airline Information (with the terms *Airline Name and *Flight # – the * represents an optional field). The algorithm in Section 3 makes use of composition to overcome granularity differences. It is worth noting that there is a rich body of literature on mereology (*e.g.*, [24, 27]). However, the minimal support of ontological structures in HTML render these differences immaterial in this framework.

Precedence: The last construct we have considered is the precedence relationship among composite terms. In any interactive process, the order in which one provides the data may be of importance. In particular, data given in an earlier stage may restrict the number of options that are available for a later entry. For example, the Avis Web site determines the car groups available for a given session, using the information regarding the pick-up location and the pick-up time. Therefore, once those entries are filled in, the information is sent back to the server and the next form is brought up. Such precedence relationships can usually be identified by the activation of a script, such as (but not limited to) the one associated with a SUBMIT button. In any such case, we compose all simple and composite terms that are separated by script execution and assign a precedence relationship accordingly. It is worth noting that the precedence construct rarely appears as part of the basic ontology constructs. This can be attributed to the view of ontologies as static entities whose existence is independent of temporal constraints. We anticipate that contemporary applications, such as the one presented in this paper, will need to embed temporal reasoning in ontology construction.

2.2 Target Ontology

Before we describe our ontology extraction process, we introduce two quality metrics, namely *recall* and *precision*, exploited for effective ontology creation through dynamic ontology adaptation.



❶ **Rental Pick-Up & Return Information**
Enter the city or airport code of the rental pick-up location in the field below if known, or select RENTAL STATION SEARCH to perform a city search.

Pick-Up Location Code: Rental Station Search

Pick-Up Date: Day February
Pick-Up Time: Hour 00 [✓ Am] Pm
(For example: 12 00 PM = Noon)
Return Date: Day February
Return Time: Hour 00 [✓ Am] Pm

Fig. 2. Rental Pick-Up and Return Information on Avis Web site.

Metrics Following common IR practice for retrieval effectiveness (*e.g.*, [11]), We shall use the following two metrics for performance analysis. The first is the *recall* (completeness) metric, defined as the ratio of relevant terms retrieved for a given ontology over the number of relevant terms in that ontology. The denominator is taken as the number of fields to be filled during the reservation process. A subjective analysis recovers the numerator. For example, consider Figure 2, which provides the rental pick-up and return information on Avis Web site. The relevant terms are all given to the left of the fields (in this case). However, it is possible that an algorithm would use the statement "*(For example: 12 00 PM = Noon)*" as the term for the line below it (assuming it is a header). This will constitute an irrelevant term.

The second metric is *precision* (soundness), the ratio of the number of relevant terms retrieved over the total number of terms retrieved. In our experiments we shall use the combined measure as suggested in [20]. For a precision value P, a recall value R, and an importance measure b, the combined measure E is computed to be

$$E = 1 - \frac{(1+b^2)PR}{b^2P+R}$$

A low E value indicates a higher combined value of P and R.

Ontology Extraction The extraction process begins with an empty ontology. Once the Web site is accessed by the system browser, the page is parsed into a data structure, called the *DOM tree* (short for *document object model*), which identifies the page elements. This W3C standard for hierarchical data structure is basically a tree that represents the whole HTML page. The DOM tree can be used in a fairly straightforward manner to identify form elements, labels, input elements, *etc.* A DOM tree potentially contains "noise" due to incorrect specification of the source HTML code including difference in order of opening and closing tags and missing closing tags. Hence, suitable "cleaning" and filtering is required before achieving an error-free tree. Examples of such cleaning process include elimination of superfluous tags, removal of formatting and scripting tags, *etc.*

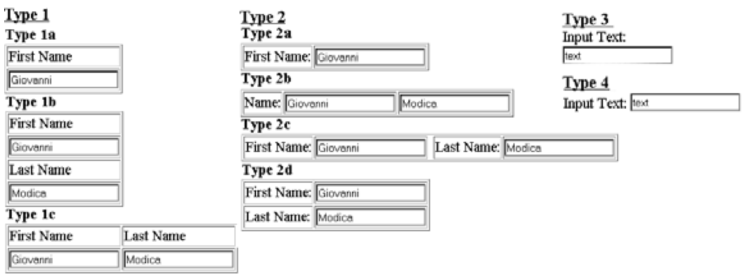
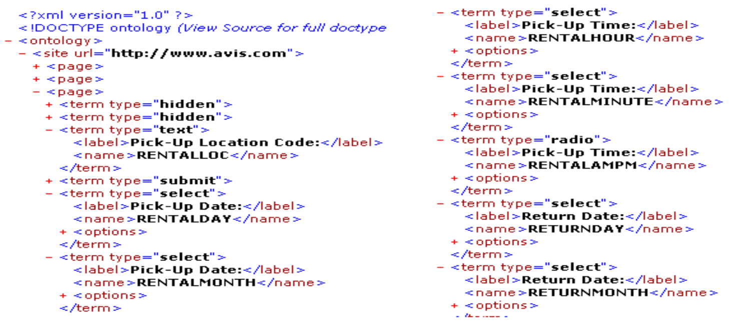


Fig. 3. Possible input layout combinations.

The diversity of layout techniques and principles used in Web design complicates the label identification process for input elements even in a well-structured DOM tree.

We have used a set of heuristics, learned from a training set of HTML documents, to recognize an HTML page layout. These documents depict all possible table and non-table input layouts we have encountered. Examples of input layout include text and image labels for input elements (or forms), and table type and row type label and input field forms. Figure 3 demonstrates some of these combinations used in the training set. Type 1 and Type 2 show tabular input formats for vertical and lateral layouts respectively. Type 3 and Type 4 show the same for plain (non tabular) input formats. Training the system using this example set resulted in a quite successful label identification process (see Section 3 for details). We attribute this success to the use of the ontological structures presented in Section 2.1. In particular, structures such as terms and values have greatly improved the identification by providing a critical semantic interpretations of source HTML documents. The composition and precedence structures have helped locating terms and associating the labels with the names of the elements by identifying semantic proximity and order of page elements.

The extraction process returns all the identifiable input elements as a set of 5-tuples $\langle u, n, l, t, S\rangle$ where u is the URL of the page (source HTML document), n is the name of the input field, l is the label of the input field, t is the type of the field, and finally, S is the set of values that are listed in a select, radio, or check box fields. The end result of the extraction process is an XML document containing the extracted ontology of the site. The XML document properly identifies the ontological structures discussed in this section to facilitates effective information retrieval at a later stage.


```
  <?xml version="1.0" ?>
  <!DOCTYPE ontology (View Source for full doctype
- <ontology>
  - <site url="http://www.avis.com">
    + <page>
    + <page>
    - <page>
      + <term type="hidden">
      + <term type="hidden">
      - <term type="text">
          <label>Pick-Up Location Code:</label>
          <name>RENTALLOC</name>
        </term>
      + <term type="submit">
      - <term type="select">
          <label>Pick-Up Date:</label>
          <name>RENTALDAY</name>
        + <options>
        </term>
      - <term type="select">
          <label>Pick-Up Date:</label>
          <name>RENTALMONTH</name>
        + <options>
        </term>
```

```
- <term type="select">
    <label>Pick-Up Time:</label>
    <name>RENTALHOUR</name>
  + <options>
  </term>
- <term type="select">
    <label>Pick-Up Time:</label>
    <name>RENTALMINUTE</name>
  + <options>
  </term>
- <term type="radio">
    <label>Pick-Up Time:</label>
    <name>RENTALAMPM</name>
  + <options>
  </term>
- <term type="select">
    <label>Return Date:</label>
    <name>RETURNDAY</name>
  + <options>
  </term>
- <term type="select">
    <label>Return Date:</label>
    <name>RETURNMONTH</name>
  + <options>
```

Fig. 4. Extracted ontology in XML.

Example of an extracted ontology is shown in Figure 4. The content of this figure can be explained with the help of Figure 2 for Avis Web site, for which the ontology in Figure 4 was generated. In Figure 4, the term `Pick-Up Date` is an example of a successful extraction for the target ontology, while `submit` is not. It is worth noting that in correspondence with the `Pick-Up Date` lateral tabular choice box in Figure 2, we have two label-name pairs in the extracted ontology, i.e., `Pick-Up Date`-*RENTALDAY* and `Pick-Up Date`-*RENTALMONTH*. The latter is extracted just because it was a term in the Web site which will play no role in the extracted ontology (spurious terms). It is also noteworthy that in case the Web site for which the ontology is being extracted has multiple pages, the ontology is created as a combination of information from all the pages the user has actually visited during the training session.

3 Ontology Adaptation

In the adaptation phase, the user suggests browsing other, similar Web sites. Each such site goes through the extraction process, resulting in a *candidate ontology.* The candidate ontology is then merged with the existing ontology (termed *target ontology*). This process refines and generalizes the existing ontology to include more terms, mapped into the existing ontology, to the extent possible. This process is repeated for each new Web site visited in the adaptation phase.

The adaptation effectiveness is measured in terms of the metrics recall and precision presented earlier. Recall measures the percentage of the candidate ontology terms that were successfully mapped to the terms of the target ontology. Precision, on the other hand, is used to quantify the semantic correctness of ontological mapping from candidate ontology to target ontology. This measure is important because syntactic mappings (matching) of labels are not always semantically correct. Together, recall and precision attempts to quantify the adaptation effectiveness of a set of ontologies.

The adaptation of a candidate ontology with the target ontology involves a series of steps, to be discussed shortly. Several heuristics are used in this process to improve performance. For example, while it is possible to consider case sensitivity, we do not use case sensitivity of labels and terms during adaptation process to improve recall measure. Examples of other heuristics include removal of noise characters, hyphenations, etc. The following steps are performed in succession on the two ontologies in the order they are listed.

Ontologies are merged pair-wise and the best match for each existing term in the target ontology is selected. The terms that are not matched or are poorly matched (below a threshold) are usually not selected and discarded. However, a choice is given to the user during the merging process (see discussion in Section 4) to select any unmatched term for the inclusion in the merged ontology through the Form Viewer module. During a query session, use of such terms in the query may result in accessing only the sites (through the Navigation Module discussed in Section 4) that contain these terms. However, unmatched terms may be matched by declaring them to be synonymous through the use of a thesaurus as we explain later in this section and also in section 4.

Textual Matching: In this step all the terms are compared pair-wise and tested for identical textual match (equality test). Usually the recall after this step is very low as labels and terms are unlikely to be named identically although they represent the same term.

Ignorabale Character Removal: Characters such as '*', '/', '-', *etc.* are treated as "noise" and are considered dispensable, and as such are removed from the terms. Hence, after this step, terms such as "*Country" and "country" will be considered identical. The argument here is that such characters do not contribute in creating a meaningful identifier for a database field name.

De-hyphenation: Labels such as "pick-up" and "pick up" are considered identical (*e.g.*, [10]). Hence, hyphens in labels are removed to improve matching. From our experiments, it turned out that by merging the hyphenated words, one yields better recall than replacing hyphens with white space. Hence, we merge hyphenated words by removing the hyphens. For example, "pick-up" will be replaced by "pickup".

Stop Terms Removal: Common terms such as 'a', 'to', 'in', and 'the' are considered *stop terms* (*e.g.*, [12]). The removal of stop terms improve recall and does not adversely affect the precision.

Substring Matching: Labels are matched pair-wise for substring matching. A parametric threshold for term matching is used.[4] The match effectiveness for two terms t_1 and t_2 is defined as the ratio between the number of words in term t_2 that are substrings of terms t_1 and the number of words in term t_2, providing a measure of the semantic similarity of these two terms. If one term contains more of the words of another term, the more similar they are. For example, the match effectiveness of t_1=`Pickup Location` and t_2=`Pick-up location code` can be computed at 66%.

Content Matching: Fields with select, radio, and check box options are processed using their value-sets. A match effectiveness is applied here too, calculated as the number of options in the second term that match (using substring matching) with the options in the first term, divided by the number of options in the second term. For example, suppose that t_1 is a `Return-time` term and t_2 is a `Dropoff-time` term with values such as {10:00am, 10:30am, 11:00am} and {10:00am, 10:15am, 10:30am, 10:45am, 11:00am} respectively. Then, if we inspect each value of t_2 for a match in t_1 (using substring matching technique already described), we will not find a match for values such as 10:15am, 10:45am, and so on. Hence the match effectiveness of t_2 (with respect to t_1) will be calculated as $\frac{3}{5} = 0.60\%$ for this example. The power of content matching can be further highlighted using the case of terms `Dropoff Date` in Alamo and `Return Date` in Avis. These two terms have associated value sets $\{(Select), 1, 2, \ldots, 31\}$ and $\{(Day), 1, 2, \ldots, 31\}$ respectively, and thus their match effectiveness is $\frac{31}{32} = 97\%$, and hence are identified by our method as semantically identical concepts.

Thesaurus Matching: Finally, terms and labels, not matched before, are matched using an ever expanding thesaurus. The thesaurus is constructed from user interactions with the adaptation module. Mismatched terms are presented to the user for manual matching. Every manual match identified by the user is accepted as a synonym and optionally a label is assigned. Each such manual match expands and enriches the thesaurus. This thesaurus is consulted in future matching to improve recall and precision.

4 System Architecture

The modular architecture of the ontology creation and query system is shown in Figure 5. There are three main modules in this system – (i) user interaction module that

[4] Usually a 50% match is considered a good measure, and hence we have used this threshold in all the steps described in this section. However, the user can adjust this threshold through the user interface, if desired.

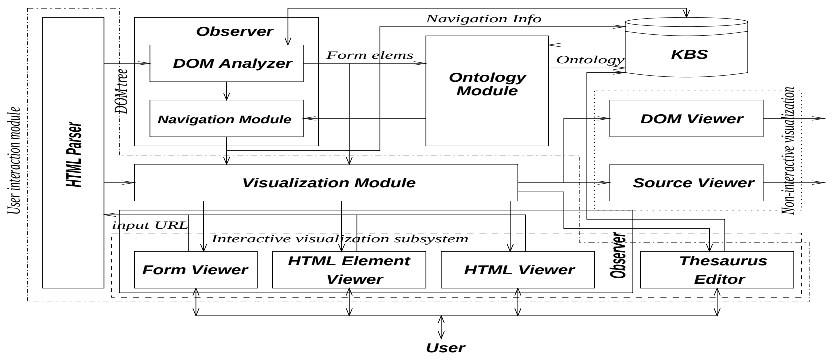


Fig. 5. The system architecture.

includes an HTML Parser and an interactive visualization module, (ii) an observer module, consisting of the DOM analyzer, Navigation Module, Form Viewer, HTML Element Viewer, and the HTML Viewer, and (iii) an Ontology Module.

The user interacts with the system through the user interaction module. She accesses a set of Web sites of interest through this module and provides feedback to the system. Each visited site is parsed by the HTML Parser with the help of an HTML/XML parser library to produce a DOM tree that represents the page. During this process, the input page is filtered to remove identifiable "noise" such as formatting tags (*e.g.*, <font>) and scripting tags (*e.g.*, <script>) before it is passed on to the DOM Analyzer. The DOM Analyzer identifies the HTML elements for the Ontology Module. Examples of HTML elements are <a>, <form>, <input>, and <select> elements, and <meta> and <frame> tags. The DOM Analyzer also captures the labels for each element in a form and forwards the information to the Navigation Module, to navigate and query the Web sites when an actual query is submitted by the user in a query session following the ontology extraction.

The observer module gathers information from the user interaction sessions. The information is forwarded to the Ontology Module for creating a target ontology or adapting a candidate ontology to a target ontology. The Ontology Module applies the methodology presented in Section 3 to perform its functions. The Navigation Module stores site-related information for future use. Its role in query processing is to recreate the memorized user navigation pattern and apply site specific terminologies in queries and thus it acts like a wrapper generator.

The Visualization Module acts as a link between the system and the user. It presents a Web browser-like functionality through which the user can interact with the Web sites. The Form and HTML element viewers are used in conjunction with the HTML viewer for this interaction. These interactions cycle through the HTML Parser, the observer and the Visualization Module before they become visible through these viewers again. The DOM Viewer and the Source Viewer are non-interactive in nature and are used for referencing purposes. The system also maintains a user editable thesaurus for effective ontology creation. Once the ontology is created, the thesaurus is automatically updated with new information.

5 Evaluation and Conclusion

In this work we have presented a tool to assist users in dynamic information seeking, using machine-generated ontologies. Our experiments show promise in creating ontologies by extracting concepts and structure from Web pages and matching two somewhat different ontologies using tools from classical IR. Before we conclude, we would like to present some results obtained in our study that substantiate our methodology.

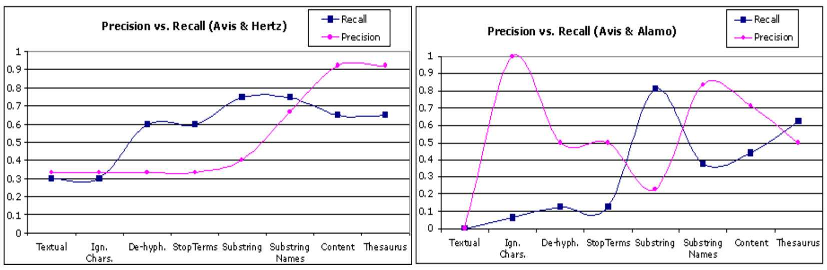


Fig. 6. Precision versus Recall for Avis as target ontology with candidates Hertz (left) and Alamo (right).

In our experiment, we have taken Avis as the target site for creating the car rental ontology. Figure 6 shows two instances in which we adapt Avis with Hertz (left) and Avis with Alamo (right). The X-axis corresponds to the various steps as described in Section 3. The results in Figure 6 are intuitive and expected, and validates our hypotheses. In the case of Avis and Hertz (left), high initial recall accounts for pairing of terms that potentially contains semantically incorrect matches. Once content and thesaurus matching is applied, these matches are rejected giving rise to diminished recall values hand-in-hand with increasing precision values. Avis and Alamo has a complementary situation where we have high initial precision and low recall that was corrected at the end of the process. This scenario is a result of a very low, but correct (100%), term matches during the initial stages in the merging process. Both metrics have demonstrated good results (more than 50% in both cases with a precision of more than 90% in the case of Hertz).

The metric E, discussed in section 2.2, can be used to demonstrate the gradual improvement of the ontology creation as we iterate through the various steps. Figure 7(left) shows the combined E values for the entire process during ontology creation for Avis-Hertz (left) and Avis-Alamo (right) presented in Figure 6. The plots suggest that the combined measure E gradually approaches zero and thus increases the accuracy of the ontology. Finally, Figure 7(right) shows that an improved thesaurus results in a diminished E value and thus increases the accuracy of the ontology. We have used a b value of 0.5 in all our current experiments. The choice of the parameter $b = 0.5$ in the expression for E indicates the fact that the user is twice as interested in precision than in recall.

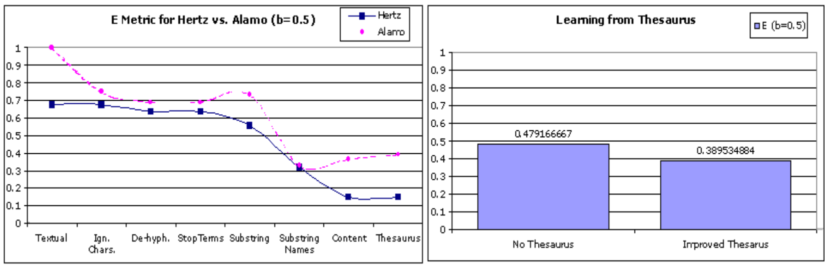


Fig. 7. Combined measure E for the graphs in Figure 6 (left). Effect of a thesaurus (right).

The paper is a step in providing theoretical, as well as practical, foundation for automatic ontology generation. It is our belief that such a process would allow the creation of flexible tools to manage metadata, either as an aid to a designer or as an independent system ("smart agent") for time critical missions. Also, an automatic reconciliation process would allow data management systems to use data even though it is originated from different ontologies.

Acknowledgments

We would like to thank Nicholas Belkin for his assistance.

References

1. *Concise Oxford Dictionary.* Oxford Univ. Press, 8 edition, 1991.
2. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J.L. Wiener. The LOREL query language for semistructured data. *International Journal on Digital Libraries*, 1(1), 1997.
3. J. Aitchison, A. Gilchrist, and D. Bawden. *Thesaurus construction and use: a practical manual.* Aslib, London, third edition, 1997.
4. Y. Arens, C.A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. In G. Wiederhold, editor, *Intelligent Integration of Information*, pages 11–42. Kluwer Academic Publishers, 1996.
5. A. Borgida. Knowledge representation, semantic data modelling: What's the difference? In *Proceedings of the 9th International Conference on Entity-Relationship Approach (ER'90)*, pages 1–2, Lausanne, Switzerland, 1990.
6. M. Bunge. *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World.* D. Reidel Publishing Co., Inc., New York, NY, 1977.
7. M. Bunge. *Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems.* D. Reidel Publishing Co., Inc., New York, NY, 1979.
8. M.J. Carey et al. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proceedings of the RIDE-DOM workshop*, pages 124–131, 1995.
9. S. Castano, V. De Antonellis, M.G. Fugini, and B. Pernici. Conceptual schema analysis: Techniques and applications. *ACM Transactions on Database Systems (TODS)*, 23(3):286–332, 1998.

10. C. Fox. Lexical analysis and stoplists. In W.B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 102–130. Prentice Hall, Englewood Cliffs, NJ 07632, 1992.
11. W.B. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures & Algorithms.* Prentice Hall, Englewood Cliffs, NJ 07632, 1992.
12. W. Francis and H. Kucera, editors. *Frequency Analysis of English Usage.* Houghton Mifflin, New York, 1982.
13. A. Gal. Semantic interoperability in information services: Experiencing with CoopWARE. *SIGMOD Record*, 28(1):68–75, 1999.
14. J. Kahng and D. McLeod. Dynamic classification ontologies for discovery in cooperative federated databases. In *Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)*, pages 26–35, Brussels, Belgium, June 1996.
15. T.D. Millstein, A.Y. Levy, and M. Friedman. Query containment for data integration systems. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Dallas, Texas, May 2000. ACM Press.
16. A. Moulton, S.E. Madnick, and M. Siegel. Context mediation on Wall Street. In *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS'98)*, pages 271–279, New York City, New York, August 1998. IEEE-CS Press.
17. S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. In L.M. Haas and A. Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 295–306. ACM Press, 1998.
18. A.M. Ouksel and C.F. Naiman. Coordinating context building in heterogeneous information systems. *Journal of Intelligent Information Systems (JIIS)*, 3(2):151–183, April 1994.
19. A.M. Ouksel and A.P. Sheth. Semantic interoperability in global information systems: A brief introduction to the research area and the special section. *SIGMOD Record*, 28(1):5–12, March 1999.
20. C.J. Van Rijsbergen, editor. *Information Retrieval.* Butterworths, London, 1979.
21. G. Salton and M. McGill. *Modern Information Retrieval.* McGraw-Hill, New York, 1983.
22. P.L. Schuyler, W.T. Hole, and M.S. Tuttle. The UMLS (Unified Medical Language System) metathesaurus: representing different views of biomedical concepts. *Bulletin of the Medical Library Association*, 81:217–222, 1993.
23. A.P. Sheth, S.K. Gala, and S.B. Navathe. On automatic reasoning for schema integration. *Intenational Journal on Intelligent Cooperative Information Systems (IJICIS)*, 2(1):23–50, June 1993.
24. P. Simon. *Parts: A Study in Ontology.* Clarendon Press, New York, NY, 1987.
25. H. Smith and K. Poulter. Share the ontology in XML-based trading architectures. *Communications of the ACM*, 42(3):110–111, 1999.
26. D. Soergel. *Organizing information : principles of data base and retrieval systems.* Academic Press, Orlando, FA, 1985.
27. A. Varzi. On the boundary between mereology and topology. In R. Casati, B. Smith, and G. White, editors, *Philosophy and the Cognitive Sciences.* Hoelder-Pichler-Tempsky, Vienna, Austria, 1994.
28. B.C. Vickery. *Faceted classification schemes.* Graduate School of Library Service, Rutgers, the State University, New Brunswick, N.J., 1966.