IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP5005

ODIL: Ontology-Based Document Interrogation Language

Soraya Abad-Mota
abadmota@usb.ve, Universidad Simon Bolivar, Departamento Computacion, Aptdo 89.000, Baruta, Venezuela, Phone: 58-212-906-3266, Fax: 58-212-906-3121

Paul A. Helman
helman@cs.unm.edu, University of New Mexico, Computer Science Department, 301G Farris Engineering Center, Alburquerque, NM, 87131, USA, Phone: 1-505-277-2967, Fax: 1-515-277-6927

1 INTRODUCTION

The context for this research is an information processing scenario with data from different sources available on the web. We are mostly concerned with textual documents in any of their three possible forms: structured (pieces of text explicitly tagged), semi-structured (pieces of text with some implicit structure, which may be given by lexicographic clues contained in the text), and free (neither semi-structured nor structured). It is important to note that our main interest is in text sources and not semi-structured data, as defined in the semantic web literature ([1], [2], and [3]). Our architecture is applicable to semistructured data, like the ones found in html or xml sources, but our experiments are on semi-structured text, as defined above.

The content of the textual documents is to be made available to many users, through a simple interface language based on the relevant concepts in the domain of the documents. This way, the users will not be required to know about complex data structures or representations of the data contained in the documents.

Given the requirements stated above, the main goal of our project is to design a system which will extract the relevant information from the textual sources so that "semantic queries" involving some knowledge about the universe referred to in the documents can be answered.

In a recent paper ([4]) the *Document Interrogation Architecture (DIA)* was presented. The purpose of *DIA* is to extract information from a set of documents, to structure and store that information in a relational database, and to allow the interrogation of the documents using a simple semantic language which is processed against the relational database. The database contains the instances of the concepts referred to in the documents; an ontology is created with these concepts and can be extended using special language constructs.

The novelty of *DIA's* approach is that the database is not the source of the truth; instead, the documents hold the real content for the answers; these answers are anticipated by extracting relevant data from the documents and structuring them in the form of a relational database (RDB).

The set of documents is divided into types of documents, for each of which an Extended Entity-Relationship (EER) conceptual schema of the contents of the document type is built (see [5] for a description of the Extended ER model we use). This approach is not compatible with the traditional "closed world assumption (CWA)" from deductive databases and artificial intelligence, it needs a more appropriate assumption, for example the "local closed world information (LCW)" defined in [6], but this issue is beyond the scope of this article.

The simple semantic language of *DIA* is *ODIL (Ontology-based Document Interrogation Language*), it has an SQL-like syntax, but the targets of a query written in this language are concepts from the ontology, whereas the targets of an SQL query are relational tables. The subject of this paper is a brief description of ODIL emphasizing its interaction with the information extraction components of the architecture.

The paper is organized as follows. In section 2 we describe the database, the ontology and the high-level activities of DIA. The language is specified in section 3. Section 4 highlights the interaction between ODIL and the information extraction processes of DIA. Section 5 contains the conclusions of this work.

2 ONTOLOGY AND PHASES OF THE ARCHITECTURE

We cite from [7] a general accepted definition of a computer ontology: "agreement about a shared, formal, explicit and partial account of a conceptualization". Furthermore, Meersman et al. add: "an ontology contains the vocabulary (terms or labels) and the definition of the concepts and their relationships for a given domain."

Meersman et al. ([7]) contrast data modelling with ontology engineering and highlight several differences between data models and ontologies, in our work we try to combine both worlds. An ontology is more flexible and general as the basis of a query language, but a database and a query processor which evaluates queries against it is easier to implement.

Our approach is to define a query language based on an ontology, map this ontology to a relational database, and evaluate the queries against the database using the mapping. To apply this approach to DIA, a global EER schema for all the document types is translated into the schema of the RDB where the data extracted from the documents will be stored. This global schema is mapped into the ontology. The user poses queries to the documents based on the concepts defined in the ontology, she may also define new concepts; for the latter a mapping to the EER schema must be defined. An ontology-based query is translated into one or more RDB-based queries in order to obtain an answer from the data in the RDB. This translation is supported by the mappings from the ontology to the EER schema and from this schema to the RDB.

The query processing in *ODIL* requires two fundamental mappings: from the EER schema to the ontology (used to load the initial ontology) and from the ontology to the EER schema which is needed when the user defines new concepts in the ontology, to allow queries on these new concepts. These mappings are fully specified, but are not included in this paper for space reasons.

To generate the initial ontology we need to map each concept in the global conceptual schema into concepts in the ontology. Before loading the ontology, the relational database is filled with data by a broad information extraction procedure. The goal of the initial ontology load is to populate the ontology with all the concepts present in the relational database, which were obtained from the original documents. In a way, the concepts in the ontology are like views over the global EER schema.

In essence, the fundamental issues around *DIA's* purpose are:

- The creation of an integrated conceptual schema of the contents of all the documents. Each document type has a schema and one document type may have many documents as instances of that type. Each document belongs to only one type.
- The automatic extraction of data from the documents and their structuring in a relational database.

- The mapping of the relational database to an ontology and viceversa, to allow queries based on the ontology.
- The focused information extraction from the documents after determining that there was not enough data in the RDB to answer some query.

These issues are dealt with in the phases of the architecture described next.

I. Data Preparation: This phase is performed manually and consists of the document type schema creation and the global schema integration activities.

II. Data Extraction: In this phase the actual extraction of data from documents is performed, involving the following steps:

1. Broad Information Extraction (BIE). The BIE component is a model-based information extractor, it has some similarities with the DEG approach proposed in [8]. This component does multislot extraction with as little human intervention as possible.

2. Database Population (DBP). The output of the BIE module is data extracted from the documents. Once the extracted data elements are related to the EER schema elements (in the BIE step), the mapping from the EER model to the relational model (necessary for loading the data) is fairly simple.

3. Ontology Generation (OG). The RDB loaded in the previous step is the basis for building the initial ontology. This process needs a mapping from the EER model concepts to the ontology concepts. In order for this to be correct, the relational database model that is implemented must correspond to the ER model resulting from the integration of the document type conceptual schemas. After the initial ontology is generated, the user of the query language extends its contents by using the appropriate constructs of the language.

III. Document Interrogation: The ODIL language has two sets of constructs, one for defining new concepts that will extend the ontology and can later be used to ask queries, and another set for asking the query. The processes performed in this phase are:

1. Knowledge Processing.
   a) Knowledge acquisition: involves the analysis of the query Q to separate the defined concepts from the concepts used to interrogate. The defined concepts are used to augment the ontology and a mapping is established between the newly defined concepts and the existing ones.
   b) Query Translation: maps the concepts used in the interrogation part of Q to concepts in the RDB, to transform the original semantic query into one or more subqueries over the RDB.
2. Approximate Query Processing. The processing of queries in ODIL should be flexible to provide approximate answers in the absence of accurate data (see section 4). This processing requires a measure of goodness of the answer or a measure of accuracy of the data in the RDB. With such measure the system can reason about the existing data and provide an approximate answer. We are planning on developing our measure based on the works of Levy [9] and Lambrecht et al. [10].
3. Focused Information Extraction (FIE). After a user has received an approximate answer to a query Q, the system may use Q to return to the documents, looking for additional data to extract. Hopefully, the additional information extracted will help return a more accurate answer to Q.

3 THE DOCUMENT INTERROGATION LANGUAGE

This section presents the two kinds of clauses of ODIL.

3.1 Concept definition Clauses

A new concept is defined by specifying the following aspects:

- A name for the concept.
- A definition based on one of the abstractions available, namely: primitive concept, aggregation, generalization, specialization, and enumeration. We define the details of this definition below.
- The source of the concept indicates where the concept was originally defined; if it was defined by using the special concept definition clauses, the value assigned to source is ontology, otherwise, the value of source is RDB.
- A set of instances that belong to this new concept. When defining a concept by enumeration the instances already exist in the relational database. In this case, we only need to refer to the name, the row id or the object id of these instances.
- A set of explicit constraints, not covered by the inherent constraints of the ontology, that must hold for the concept.
- A list of synonyms for the name of the concept. This list must be disjoint with respect to all other names and synonyms in the ontology.
- A descriptive phrase in natural language to describe the new concept. The phrase will be useful when doing focused information extraction, as a hint to find relevant data in the documents.

The definition of the new concept using the abstractions is done with the following constructs:

1. Primitive concept. This abstraction is used when defining a new concept by specifying its name. This abstraction is useful to define properties of other concepts. For example, the concepts name, population, abbreviation, and geographical location may be defined as primitive concepts.
2. Aggregation. Defines a new concept A as an aggregate of some existing concepts; the notation is: $A = (C1, C2, \ldots, Cn)$, where the Ci's are concepts, either primitive or non-primitive, already defined in the ontology. For example, we may define the concept country as an aggregation of name, population, abbreviation of the country's name, and geographical location; according to our notation we write this definition as follows: country = (name, population, abbreviation, geographical location), in the ontology. Two other examples are, Institution(Iname, IType) and belongs(institution, country).
3. Generalization. This abstraction defines a general concept, associated with some other concepts that are specialized versions of it. The notation for this abstraction is: $G \supseteq S1 \cup S2 \ldots \cup Sn$, where G is the new concept being created with this abstraction. Conversely, by using specialization we may define a new concept as a specialized version of an existing general concept. Given the above definition of Institution, we may now define memberOfIstec $\subseteq$ Institution.
4. Enumeration. A concept is defined by enumeration when explicitly listing all its instances. The notation for defining a concept C as an enumeration of instances is: $C = \{I1, I2, \ldots, In\}$, where the Ij are instances of a concept existent in the relational database or in the ontology. After we have defined region as an aggregation of country, we may define AndeanRegion as an instance of region: region = {AndeanRegion} and later, as an enumeration of five countries:

AndeanRegion = {Bolivia, Colombia, Ecuador, Peru, Venezuela}. These two definitions of Andean Region are consistent.

It is important to note that the enumeration construct allows the definition of a concept by extension, whereas all the other abstractions define a concept by intension. The same concept that is defined by extension, with an enumeration, must be defined by intension with one of the other abstractions, in order to have a complete definition of it, and to be able to do the mappings between the ontology and the RDB properly. The enumeration abstraction is useful because it ties the concepts in the ontology with actual data in the RDB. The other abstractions are useful to describe the nature of the concept being defined.

The only instances contained in the ontology are those used in definitions by enumeration. An instance itself is not really present in the ontology; only a reference to the actual data in the RDB is what is stored in the ontology.

3.2 Inquiry Clauses

There are two main forms of a query in this language given by the following syntax:

List | **Count** < target concept > **instances**
show (property, option)*
such that < condition >

where

< target concept > is a concept that exists in the ontology. This concept should be defined by aggregation, or generalization, or specialization.

The **show** and the **such that** clauses are optional.

< property > is a primitive concept which must be a property of the target concept; for each property the query processor will find (or compute) its value and provide it with the answer; in the future it might be useful to allow a function applied to the property.

< option > the possible values are: for each and for all; defines if the property is to be computed for each instance in the output or if it will compute a single value for all the instances.

* means that there may be several pairs (property, option) associated with the same target concept.

< condition > is an expression that qualifies the instances of the target concept that are selected to be in the answer to the query.

The user may build a compound condition using a combination of the conditions specified above and linking them with boolean operators. For example, a possible query is:

List institution **instances**
such that c ∈ Country AND c.Cname = 'Venezuela'
AND i ∈ institution AND belongs(i,c)
AND m ∈ memberOfIstec AND m IsA i

4 INTERACTION WITH INFORMATION EXTRACTION

In the non-traditional approach to query processing that we plan to do with ODIL, we assume that there might not be enough information in the database to provide a precise answer to a query. For example, the total number of Computer Science students in Venezuela, cannot be answered precisely unless we have absolute numbers for all Venezuelan institutions. But if we have these data for two Venezuelan institutions, we have a portion of the required answer. So instead of giving an empty answer, we could reply: "the number of Computer Science students in Venezuela is greater than or equal to the number of Computer Science students in these two Venezuelan institutions for which the database has data". This answer is an approximate answer to the original query. It bounds the value of the precise answer and in this sense it is an approximation. In order to provide these kinds of approximations, the query processor must reason about the queries using the ontology, the knowledge about the domain, and metadata describing the database and the documents. Having this mechanism available allows us to go back to the original textual documents to try to find more data to improve the approximate answers.

When a query requires approximate processing, after the approximation has been provided, the query is analyzed against the information extraction component. During this analysis the system decides if there is more information in the original documents which could have improved the answer to the query. If this is the case, additional information is extracted to populate the database. In other words, we want the system to learn from the queries it cannot answer precisely, in order to provide better answers to future queries. This additional process of extraction is what we called focused information extraction in section 2.

There is also an issue of levels of abstraction related to the interaction between information extraction and query processing. For example, we might decide how to aggregate the information extracted from the documents based on the queries we encounter. If we get mostly queries about Computer Engineering and Computer Science combined, we might decide to aggregate the individual data about these two majors for all institutions. Therefore, when we do information extraction from the documents, there is a decision to be made regarding the level of aggregation of the data to populate the global database. On the other hand, if we do have aggregate information for one institution, for example, the total number of undergraduate students, we can use statistical techniques (see [11] for an overview of these techniques) to try to guess the number of undergraduate students in Computer Science for that institution.

5 CONCLUSIONS

The work presented in this paper is part of the development of the *DIA* Architecture, which was introduced in [4] and allows high-level queries about the contents of a set of documents. The queries are processed against a relational database loaded with data automatically extracted from the documents.

The current paper contains the specification of the interrogation language of the architecture, *ODIL*, which is a high-level semantic language for interrogating documents. *ODIL* combines the abstraction level of an ontology with the advantages of query processing against a relational database. The key for doing this is the definition of appropriate mappings between the conceptual model of the contents of the documents and an ontology, that defines the relevant concepts and their relationships. The underlying goal of *DIA* is to build a system which requires as little human intervention as possible, and that can extract information from textual documents from any domain. The only required manual step is the construction of an integrated data model of the document types that will be interrogated.

ODIL is a very simple language, with a syntax similar to SQL, which operates on concepts existing in an ontology rather than over relations of a RDB. The most relevant aspect of the language is not its syntax or its expressive power; what is most attractive of this work is the use of this simple language and the other components of *DIA* to explore the interaction between information extraction and query processing. The architecture provides a mechanism to extract data automatically from textual documents and to build an ontology with them. The ontology is the basis for querying the documents and the reasoning about the answers to a query triggers focused information extraction which is guided by the query.

REFERENCES

[1] Daniela Florescu, Alon Levy, and Alberto Mendelzon, "Database techniques for the world-wide-web: A survey," SIGMOD Record, vol. 27, no. 3, pp. 59-74, September1998.

[2] Nabil R. Adam, Vijayalakshmi Atluri, and Igg Adiwijaya, "SI in digital libraries," Communications of the ACM, vol. 43, no. 6, pp. 64-72, June 2000.

[3] Dieter Fensel, James Hendler, Henry Lieberman, and Wolfang Wahlster, Eds., "Spinning the Semantic Web", The MIT Press, 2003.

[4] Soraya Abad-Mota and Paul A. Helman, "Dia: A document interrogation architecture," in Proceedings of the Text Mining Workshop in conjunction with the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-02), 2002, pp. 35-45.

[5] Ramesz Elmasri and Shamkant Navathe, "Fundamentals of Database Systems", AddisonWesley, third edition, 1999.

[6] K. Golden, O. Etzioni, and D. Weld, "Omnipotence without omniscience: Efficient sensor management for software agents," in Software Agents. Papers from the 1994 Spring Symposium (Technical Report SS-94-03).

[7] Peter Spyns, Robert Meersman, and Mustafa Jarrar, "Data modeling versus ontology engineering," SIGMOD Record, vol. 31, no. 4, pp. 12-17, Dec. 2002.

[8] D Embley, D Campbell, Y Jiang, S Liddle, D Lonsdale, Y Ng, and R Smith, "Conceptual-model-based data extraction from multiple-record web pages," Data and Knowledge Engineering, vol. 31, no. 3, pp. 227-251, November 1999.

[9] Alon Levy, "Obtaining complete answers from incomplete databases," in Proceedings of the 22nd VLDB Conference, 1996.

[10] Eric Lambrecht and Subbarao Kambhapti, "Planning for information gathering: A tutorial survey," Tech. Rep. 96-017, Arizona State University, May 1997.

[11] Soraya Abad-Mota, "Approximate query processing with summary tables in statistical databases," in Proceedings of the 3rd International Conference on Extending Database Technology. Lecture Notes in Computer Science No. 580. 1992, pp. 49-515, SpringerVerlag.

Related Content