Semiautomatic Generation of Data-Extraction Ontologies from Relational Databases

Orlando Miguel Vivan\textsuperscript{1,2}, Carlos Alberto Heuser\textsuperscript{1}

\textsuperscript{1}Universidade Federal do Rio Grande do Sul
Instituto de Informática
Caixa Postal 15064 – CEP 91501-970
Porto Alegre – RS - Brasil
e-mail: \{orlando, heuser\}@inf.ufrgs.br

\textsuperscript{2}Pontifícia Universidade Católica do Paraná
Centro de Ciências Jurídicas e Empresariais
Campus Londrina
e-mail: \{orlandov@rla01.pucpr.br\}

Abstract

Data extraction is the process used to gather and structure information in documents (e.g. Web pages). One approach to data extraction is the so-called ontology based data extraction. In this approach, an ontology is used as a guide to the parser that extracts data from the source documents. In this context, an ontology is a conceptual schema enriched with information needed to identify data items in the sources. The process of creation of an ontology is not a trivial task and may require the analysis of a big number of document instances. However, in many extraction applications, the information that is being extracted may already be modeled in a relational database. In this case, the relational database schema can be used as a starting-point to the construction of a data extraction ontology. Analysis of data instances stored in the database may help to generate the information used to parse data items in document sources. This paper presents a method for the semiautomatic creation of a data extraction ontology. This process is based on reverse engineering of the relational database schema combined with the analysis of data instances.

Keywords: Data Extraction, Ontology, Ontology construction, Grammatical inference.

1. Introduction

The use of the Web as a database is an important research subject [7]. One of the themes of investigation is the problem of extracting pieces of data from semi-structured documents like Web pages [9, 11, 12]. Among the several approaches to data extraction, one is the ontology based or semantic data extraction approach [5, 15].

In the approach developed by the Data Extraction Group (DEG) at Brigham Young University, an ontology is a conceptual schema enriched with so-called data frames. A data frame contains regular expressions or lexicon values that can be used to represent data format or data content. The ontology is used to guide the data extraction process. Ontology based data extraction has been applied successfully to the extraction of sets of records from data rich documents [3, 5]. The DEG approach is adequate for documents that are narrow in ontological breadth, which means that a document can be described with a relatively small ontology. Furthermore, the document must contain multiple records, a
sequence of chunks of information about the main entity in an ontology. The DEG group does not expect this approach to work well for unstructured documents that do not follow these rules [3].

In the DEG approach, an ontology is specified using OSM – Object-Oriented System Model [4]. In OSM, an ontology consists of lexical object sets (attributes in ER approaches) and non-lexical object sets (entities in ER approaches) as well as relationships among them. This model is enriched by two concepts, data frame and lexicon. A data frame is a regular expression that describes how a lexical object may be represented in documents. A lexicon is a list of possible values for a lexical object. The DEG parser receives as input such an enriched ontology as well as a document that contains information about several instances of a non-lexical concept that appears in the ontology. The output is the set of records with the data values of the lexical concepts that are found in the document. The DEG approach is reported as having been tested successfully on at least two case studies (obituaries and car ads). In both cases, the recall ratio and precision ratio were among 90% and 98%, respectively [3, 5, 6].

The construction of an ontology for data extraction is not a trivial task and involves the analysis of a large number of document instances. In one of the case studies mentioned above, 128 document instances were analyzed. Thus, an automatic or semiautomatic process of generation of an ontology would be helpful.

In many cases, organizations that need to extract data from documents already have relational databases that model the extracted data and may even contain instances of the data to be extracted. For example, a business organization that has a database containing information about the data it sells may be interested in extracting data from Web pages from its competitors. Another example is an organization that needs to extract legacy data contained in documents in order to include the extracted data in its database. In application domains like this, where a database already exists, a semiautomatic process for the construction of a data extraction ontology may be helpful.

The purpose of this paper is to present an approach to semi-automatically generate a DEG data extraction ontology from an existing relational database. This approach uses the knowledge contained in the database schema, as well as the knowledge contained in its data instances.

The paper is organized as follows. Section 2 briefly presents the DEG extractor exemplified with a case study. Section 3 presents the proposed approach. In this section, two topics are discussed: a) reverse engineering from the relational schema to an OSM model and b) generation of regular expressions from data instances. Section 4 presents a case study. Section 5 contains the concluding remarks.

2. The DEG Extractor

This section presents the DEG extractor [3,5] through an example of a data extraction application. Data is to be extracted from contracts between a Telecom Company and its suppliers. A contract (Figure 1) contains the following information: contract number, administrative process number, supplier name, address, and subscription date (see highlighted text in Figure 1). We consider a set of contracts with the structure of the example in Figure 1. Such a set of documents fulfills the requirements of the DEG extractor that are: a) to be described with a relatively small ontology and b) to contain multiple records about the main entity (in our case a contract) in the ontology. Such contracts belong to Sercomtel S.A. Telecomunicações, a state owned Telecom Company in Londrina, Brazil, and they are available at an official newspaper published in the Web (http://www.londrina.pr.gov.br/jornaloficial/index.php3). Contracts are stored in Word format and transformed into HTML in order to input the data extraction process.
CONTRATO Nº 14.810

CONTRATO DE PRESTAÇÃO DE SERVIÇOS, QUE ENTRE FAZEM A SERCOMTEL S.A. – TELECOMUNICAÇÕES E O. A. SCOTTON & CIA LTDA - ME.

Pelo presente instrumento de contrato, vinculado ao Convite n.º 003/2002, constante do Processo Administrativo n.º 019/2002, de um lado como CONTRATANTE, a SERCOMTEL S.A. - TELECOMUNICAÇÕES, sociedade anônima de economia mista, inscrita no CNPJ/MF sob o n.º 01.371.416/0001-89, com sede nesta cidade, na Rua Professor João Cândido n.º 555, neste ato representada por seu Presidente, FRANCISCO ROBERTO PEREIRA e por seu Diretor Administrativo-Financeiro e de Relações com Investidores, WALTER MASSAO IKEDA, doravante denominada simplesmente SERCOMTEL e, de outro lado O. A. SCOTTON & CIA LTDA - ME, pessoa jurídica de direito privado, inscrita no CNPJ/MF sob o n.º 80.062.920/0001-73, estabelecida na Rua Piauí n.º 554, na cidade de Londrina, Estado do Paraná, neste ato representada por seu Sócio-Gerente ODAIR ANTONIO SCOTTON, doravante denominada simplesmente CONTRATADA, ajustam e celebram o presente contrato, dentro do recurso orçamentário ORES ASI-662/01, havido pela conta n.º 313.34.3 – Produção de Cópias, em consonância com o disposto na Lei nº 8.666/93 e demais cláusulas e condições a seguir estipuladas:

Londrina, aos 07 de março de 2002.

Figure 1 – Example of an unstructured document

Figure 2 shows the graphical representation of the data extraction ontology for the contracts application. In the graphical representation, dotted rectangles represent lexical object sets (contract number, administrative process number, supplier name, subscription date) and solid rectangles represent nonlexical object sets (contract). Lines connecting rectangles represent relationship sets. Participation constraints represent the minimum and the maximum number of times an object in the set participates in the relationship.

![Ontology Graphical Representation](image)

Figure 2 – Ontology Graphical Representation

Figure 3 shows the same ontology in textual form (Object-oriented System Model Language (OSM-L)) already enriched with data frames. Inside a data frame, regular expressions or lexicon values can be used to represent data format or data content. These regular expressions are written in Perl-5 syntax. A data frame may also declare constants patterns and keyword patterns.
Figure 3 – Ontology Textual Representation

For example, in Figure 3, the term:

Contrnum matches [8]
  constant { extract "[0-9][5]/[0-9][2]";,
  { extract "[0-9][2],[0-9][3]";);
  keyword "CONTRATO\'sN";
end;

is the data frame for the nonlexical object Contrnum (the contract number). This data frame specifies that a contract number has maximum length eight, may be represented as a string of five digits, followed by a slash and followed by two digits or may be represented by two digits, followed by a dot and followed by three digits. The keyword term specifies that a contract number appear in a document preceded by the string “CONTRATO N”.

Once the ontology has been defined for a domain of interest, the data-extraction process parses the ontology and generates the extraction rules, which will be used as input for the data document extractor. The output is a data record table as shown in Figure 4.

Figure 4 – Data Record Table
3. Ontology Creation

The creation of the ontology is a manual process. Even though it is performed only once, it is a tedious and error prone task. In order to minimize this manual process, this paper presents a semiautomatic process to generate the ontology. This process is divided in two parts: the reverse engineering from the relational database schema to the OSM model, and the generation of regular expressions from the database data instances. The process is semiautomatic because user decisions about alternatives of ontology generation must be taken.

The first step is the reverse engineering process. This process adopts the classical reverse engineering rules defined in literature [1] for the entity-relationship approach and adapts them to the OSM approach. Only those OSM concepts that are used in the data extraction process are considered (e.g. inheritance is not handled). During this process the user defines the tables and columns that correspond to OSM objects of interest for data extraction. A prototype of a reverse engineering tool was built for Oracle databases using the Pl/sql language [17]. Figure 5 shows an example table in the database (left) that stores data about contracts and its equivalent OSM model (right) after the reverse engineering process.

<table>
<thead>
<tr>
<th>Table contract</th>
<th>CONTRACT [--&gt; object]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contrnum</td>
<td>CONTRACT [1] has CONTRNUM [1]; CONTRNUM matches [8] end;</td>
</tr>
<tr>
<td>Admproc</td>
<td>CONTRACT [0:1] has ADMPROC [1:*]; ADMPROC matches [8] end;</td>
</tr>
<tr>
<td>Supplier</td>
<td>CONTRACT [0:1] has SUPPLIER [1:*]; SUPPLIER matches [20] end;</td>
</tr>
<tr>
<td>Subsdate</td>
<td>CONTRACT [0:1] has SUBSDATE [1:*]; SUBSDATE matches [7] end;</td>
</tr>
</tbody>
</table>

Figure 5 – Relational table and the equivalent OSM textual model

After building the OSM conceptual schema, the second step is to define for each lexical object the regular expressions inside a data frame. The idea is to build the data frame as automatically as possible through the analysis of column types in the database schema combined with analysis of data instances in the database.

The process of generating a regular expression from a set of strings has been widely studied in the area of natural language processing (NLP) and is known as grammatical inference [2, 8, 13, 14]. The objective of grammatical inference is to find algorithms capable of obtaining a grammar for a language from a set of strings of this language [8]. A more formal definition states that the objective of grammatical inference is that, given a set of examples of a language L(G), the process of grammatical inference must infer (find) the grammar G that gave rise to that set [13].

In literature, several inference algorithms were proposed. Some of these algorithms infer a grammar from a set of positive and negative examples [2, 14]. Positive examples are strings that appear in the language defined by the grammar, whereas negative examples are strings that do not appear in this language. In our approach, the input to the grammar inference process is the set of data instances in the database. Therefore, an algorithm that infers the grammar from positive examples only is needed.

Another feature of the inference grammar algorithms is that they construct a grammar that recognizes exactly the set of strings given as examples. In the case of data
extraction, this will not always be true. In the relationship between the database and the
documents to be processed by the data extractor two situations may appear:

- The set of values of a lexical object that may appear in the documents is a subset of the
  values that appear in a database column. An example could be a table of zip codes. The
documents contain only zip codes that already exist in an address table in a database.

- The documents may contain values for lexical objects that are not stored in the
  database. This is the case, when the documents contain data that after the extraction
  process will be included in that database, for example, when legacy documents are
  being processed to be included that database.

Thus, in our case, at least two types of grammar inference algorithms are needed.
One that generates a regular expression that defines exactly the set of data instances that
may appear in the document and another that generates regular expressions that define a
super set of the data instances that may appear in the document.

The algorithm of the first type we are using is based on a well-known algorithm for
building digital trees [16]. This algorithm is called here digi-tree algorithm. A digital tree
is a standard data structure for representing sets of strings over a given alphabet [10]. As an
example of this structure, consider some names of states from USA: California, Colorado,
Indiana, Iowa, Florida, Texas and Utah. It is possible to group together all the words that
start with C, and then do the same with I, F and proceed in the same way with each initial
letter from the set of words. Proceeding recursively with the second letter, and so on, the
set of words is transformed in a digital tree as shown in figure 6.

![Digital Tree from a Set of Words](image)

**Figure 6 – Digital tree from a set of words**

The regular expression generated by the digital tree algorithm increases with the
number of data instances to be recognized. The data extractor may impose implementation
limits to the size of the generated regular expression. For the case of very large regular
expressions the DEG data extractor accepts a file that contains all data instances (called
lexicon). Thus, the ontology creation tool offers the alternative of generating a lexicon for a
table column.

The other type of algorithm that is used applies to the case in which the documents
to be processed may contain values for lexical objects that are not stored in the database. In
this case, the instances in the database are used to build a very simple regular expression
that specifies the type of character that may appear at every position of the input string.
This algorithm is called here position-based algorithm. This regular expression will then
be utilized by the user as a basis for the construction of the regular expression needed to recognize the data instances that actually appear in the documents.

As an example, consider the strings (California, Colorado, Indiana, Iowa, Florida, Texas, Utah). In this case, the generated regular expression would be:

\[CIFTU\{aonlet\}\{ldwoxa\}\{frais\}\{[oand]\}\{[rda]\}\{[no]\}\{ia\}\].

If the user wants to apply this regular expression to a data item that corresponds to the set of all states in USA, he must change the generated regular expression accordingly to the set of values that are to be recognized.

For some types of columns no data instance analysis is needed. For columns of data types like temporal data (date, year, month) or currency a set of regular expressions from a library of predefined templates may be offered to the user. Some examples of such templates are depicted in table 1.

<table>
<thead>
<tr>
<th>Regular Expression Template</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;(0[1-9]</td>
<td>1[0-2])/0[1-9]</td>
</tr>
<tr>
<td>&quot;([d-ld]{1}[^89]d[10]</td>
<td>2[01]rd\d)&quot;</td>
</tr>
<tr>
<td>(([^-]</td>
<td>1-9)</td>
</tr>
<tr>
<td>\bpage\s*(no</td>
<td>.?s*)?d+\b</td>
</tr>
<tr>
<td>&quot;(jan</td>
<td>feb</td>
</tr>
</tbody>
</table>

The process described above, semi-automatically generates a data extraction ontology according to the DEG approach. This ontology reflects only the information that may be inferred from the database schema and data instances. However, in many cases additional knowledge about the data representation in the documents is needed to improve the generated ontology. For example, certain data items may appear in the text prefixed by specific keywords, or other items may appear with specific presentation masks, containing special symbols. An example is the contract number given in the previous section.

### 4. Case Study

In order to evaluate the process of ontology generation described in the previous section, a case study was performed. The application domain is the contracts database described in section 2. The Telecom Company has a database that holds information about existing contracts. The company further has a set of older contracts that are not stored in the database and that should be included in the database. In order to extract data from these legacy contracts and insert them in the database, the DEG data extractor was used. The ontology creation was performed following the approach described in previous section. This ontology was then applied on a training set of documents. On the basis of this experiment, the ontology was modified in order to achieve better extraction results. This section describes this experiment.

The first step was the construction of a first version of the ontology from the existing contracts table in the database. In this process we applied the rules described in the previous section. We chose the table and columns where data about contracts are stored and applied the rules of reverse engineering and generation of regular expression. The resulting ontology is shown in figure 7.
The following columns in the contracts table were analyzed: contract number (contrnum), administrative process number (admproc), supplier name (supplier) and subscription date (subsdate).

A contract number may be any number of five digits. Thus, for the contract number column (contrnum) no data instance analysis was performed. Instead, the regular expression was derived from the column type (NUMBER(5)). The generated data frame is

```
Contrnum matches [5]
    constant { extract "[0-9][5]"; }
end;
```

For the administrative process number (admproc) we have applied the position-based algorithm. The resulting analysis showed that three numeric digits followed by a slash and then followed by four numeric digits compose this column. The generated data frame is

```
Admproc matches [8]
    constant { extract "[0-9][3]/[0-9][4]"; }
end;
```

For the supplier name (supplier), that stores the name of the supplier, we have applied the position-based algorithm. The resulting analysis showed that this column may starts with any letter from the alphabet being followed by a sequence of other letters or a space. The generated data frame is

```
Supplier matches [20]
    constant { extract "[A-Z][A-Z]\s+\"s+\d\d\"; }
end;
```

For the subscription date (subsdate), we chose a regular expression from the library of templates to represent a date in Portuguese format, for example, “15 de abril de 2001”. The generated data frame is

```
Subsdate matches [40]
    constant { extract "\d{2}\s+de\s+\d{4}"; }
end;
```
In this example of application, the data instances appearing in the documents may contain data that is not already stored in the database. Thus, the option for generating a regular expression that exactly matches the set of instances in the database (digi-tree algorithm) could not be used. Instead, the algorithm that generates a first version of a simple regular expression for each column was used.

In order to execute the data extraction process, the contract ontology generated and the contracts HTML page were uploaded to the DEG’s demo page (www.deg.byu.edu). As expected, this first version of the data-extraction ontology generated did not extract correctly the required data. The reason for that is that the documents contain several input strings with the same format as the instances to be extracted. The generated regular expressions were too generic. So, the records extracted were wrong. The only data extracted correctly was the subscription date, which has a very specific format.

As a second step in our case study, we analyzed by hand five documents to be used as input in the process of data extraction and enhanced the ontology generated by the semiautomatic approach. Figure 8 shows the second version of the data extraction ontology that resulted from this step. The highlighted text indicates the changes made to the first version of the ontology.

```
Contract [-> object];
Contract [0:1] has Contrnum [1];
Contrnum matches [5]
    constant { extract "[0-9]{5}/[0-9]{2}"; }
    { extract "[0-9]{2}.[0-9]{3}"; }
    keyword "CONTRATO\sN";
end;
Contract [0:1] has Admproc [1];
Admproc matches [8]
    constant { extract "[0-9]{3}/[0-9]{4}"; }
    keyword "\bProcesso\s+Administrativo\b";
end;
Contract [0:1] has Supplier [1];
Supplier matches [20]
    constant { extract \"[A-Z] * \[A-Z]* \[A-Z]\"; }
    keyword \"empresa\b";
end;
Contract [0:1] has Subsdate [1: * ];
Subsdate matches [40]
    constant { extract \"d{2}\s+de\s+[a-z]*\s+de\s+\d{4}\"; }
end;
```

Figure 8 – Ontology corrections

In this version, the contract number has now a more precise regular expression. For contract number, administrative process and supplier keywords were defined, in order to delimit where this data items appear in the document.

With this second version of the data extraction ontology, the results reached were considerably better. The result of data extraction from five contracts is shown in figure 7.

```
<table>
<thead>
<tr>
<th>Contract</th>
<th>Admproc</th>
<th>Supplier</th>
<th>Subsdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>05/2001</td>
<td>EXCLIM PROPAGANDA</td>
<td>20 de agosto de 2001</td>
</tr>
<tr>
<td>102</td>
<td>04/2001</td>
<td>CONTRATO DO</td>
<td>04 de fevereiro de 2001</td>
</tr>
<tr>
<td>103</td>
<td>10/2001</td>
<td>IMPRESSORA</td>
<td>10 de fevereiro de 2001</td>
</tr>
<tr>
<td>104</td>
<td>16/2001</td>
<td>COM SISTEMAS</td>
<td>10 de fevereiro de 2002</td>
</tr>
<tr>
<td>105</td>
<td>16/2002</td>
<td>AST</td>
<td>10 de fevereiro de 2002</td>
</tr>
</tbody>
</table>
```

Figure 7 – Sample of extracted records
Finally the same experiment was performed with 50 contracts using the ontology of Figure 8. The results obtained are presented in table 2.

Table 2 – Results from Contracts data extraction

<table>
<thead>
<tr>
<th></th>
<th>Number of occurrences in the document (N)</th>
<th>Number of occurrences extracted correctly + partially correct (C)</th>
<th>Number of occurrences extracted incorrectly (I)</th>
<th>Recall Ratio (C / N)</th>
<th>Precision Ratio (C / (C + I))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract number</td>
<td>50</td>
<td>50</td>
<td>0</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Adm.Process number</td>
<td>50</td>
<td>35</td>
<td>15</td>
<td>0.70</td>
<td>0.70</td>
</tr>
<tr>
<td>Supplier name</td>
<td>50</td>
<td>5 + 25</td>
<td>20</td>
<td>0.60</td>
<td>0.60</td>
</tr>
<tr>
<td>Subscription date</td>
<td>50</td>
<td>40</td>
<td>0</td>
<td>0.80</td>
<td>1.00</td>
</tr>
<tr>
<td>Total</td>
<td>200</td>
<td>155</td>
<td>35</td>
<td>0.77</td>
<td>0.81</td>
</tr>
</tbody>
</table>

The results obtained with the use of the second version of the data extraction ontology are much better. The biggest problem is the supplier name. The low recall and precision ratio is not so much a consequence of bad design of the data extraction ontology, but much more a problem of the data extraction approach itself. The documents used as a test bench are written in natural language containing few delimiters to guide the extraction process. The DEG extractor was not designed for this kind of documents [3,5].

5. Conclusion and future work

In this paper we have presented a method to the semiautomatic creation of a data extraction ontology from a relational database schema and from the database data instances. The data extraction ontology produced follows the DEG approach [3,5] to extract data from unstructured documents. Instead of creating the ontology manually we have proposed a method to build the ontology as automatically as possible. In this process, the user intervenes defining which database tables and columns will be reverse engineered to the data extraction ontology, as well as defining which kind of rule will be used to construct the regular expressions that represent the data instances of each column.

This process has been applied in a case study of a company that has legacy documents. Such documents contain data that has to be extracted and included into the database. The existing database was the starting-point for the process of ontology creation. From this database a first data extraction ontology was constructed. This data extraction was tested on a set of five documents and further enriched manually. The resulting ontology was then applied to a set of 50 documents. The extraction process was rather successful, with average recall and precision ratios of 77% and 81% respectively. The documents we have used are texts in natural language and contain few keywords or markups to guide the extraction process. Probably the results could be better if more structured text, like HTML pages, were used as input.

This is an ongoing work. The algorithms to infer the regular expressions are in a first version and they are being enhanced in order to generate more precise regular expressions. After finishing the algorithms that generate the regular expressions, the approach will be tested on a set of documents, that include marks to guide the data extraction process (e.g. HTML pages) and that are more adequate to the DEG approach.
References


