SEMIAUTOMATIC GENERATION
OF RESILIENT DATA-EXTRACTION ONTOLOGIES

by

Yihong Ding

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

June 20, 2003

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Yihong Ding

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	David W. Embley, Chair
Date	Deryle W. Lonsdale
Date	Charles D. Knutson

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Yihong Ding in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

____
Date

____
David W. Embley
Chair, Graduate Committee

Accepted for the Department

____
David W. Embley
Graduate Coordinator

Accepted for the College

____
G. Rex Bryce
Associate Dean
College of Physical and Mathematical Sciences

ABSTRACT

SEMIAUTOMATIC GENERATION

OF RESILIENT DATA-EXTRACTION ONTOLOGIES

Yihong Ding

Department of Computer Science

Master of Science

The amount of data available on the World Wide Web has been growing explosively during the past few years. People want to obtain user-specified information from this fast-growing body of knowledge similar to the way they can query a database for user-specified information. BYU-Ontos is a system that addresses this problem [15, 11] and achieves good experimental results. A main difficulty for BYU-Ontos, however, is that currently people create the data-extraction ontologies manually, which is tedious, error-prone, and requires a high degree of computer-science knowledge. This thesis proposes a mechanism to semiautomatically generate data-extraction ontologies based on available knowledge resources and sample input documents. We have experimented with both the μK (Mikrokosmos) ontology [40] and SALT [38] as primary knowledge sources. Our method shows how to select object sets that should appear in a generated extraction ontology among a huge set of concepts; how to retrieve relationship sets from knowledge sources with complicated organizations; and how to discover constraints on the relationship sets among the selected object sets. Our results show that the method is effective, correctly generating on average more than 70% of the needed object sets, relationship sets, and constraints.

ACKNOWLEDGMENTS

I thank God who gave me the intelligence to do this thesis work.

I am grateful to my mentor, Dr. David W. Embley, who is not only an advisor, but also a kind elder. From him, I learned not only knowledge in the computer-science field, but also many valuable principles to be a better person. I cannot begin to enumerate the lessons I have learned from him. I would recommend him to anyone who wants to find a good advisor.

I appreciate the support of my committee. Particularly I want to thank Dr. Deryle W. Lonsdale, who has been very helpful in providing the knowledge sources for this project, shaping many of the ideas that comprise this research work, and embellishing my English. We have spent many hours together working over ideas and finding ways to improve our methods.

I am also grateful to Dr. Charles D. Knutson for his time and effort on my behalf.

I thank my parents who raised me and supported me. I deeply thank my wife, Li for her patience, her love, and her suggestions in my study. My work could not have been done without her persistent and unquestioning support. My lovely daughter, Vania, brings me so much joy.

The National Science Foundation, grant number IIS-0083127 provided partial financial support for this research work.

Finally, I would like to thank all the people in the Data-Extraction Research Group. I would not have been able to accomplish this work without their contributions.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

The amount of data available on the World Wide Web has been growing explosively during the past few years. As the Web has grown, people have become more and more interested in obtaining user-specified information from this fast-growing body of knowledge.

Traditionally, two approaches have been taken to enable users to query the Web like a database. One is to enhance traditional query languages to make them "Web aware" so that data in Web pages can be queried directly. For example, in 1997, Mendelzon, Mihaila, and Milo [41] presented a query language, WebSQL, that takes advantage of multiple index servers without requiring users to know about them, and that integrates textual retrieval with structure and topology-based queries. Later, in 1998, Arocena and Mendelzon [2] presented the WebOQL system that supports a general class of data restructuring operations in the context of the Web. WebOQL synthesizes ideas from query languages for the Web, for semi-structured data and for Web site restructuring. As another example, FLORID [24] is a prototype implementation of the deductive and object-oriented formalism F-logic [27]. Working as a Web query engine, FLORID provides a formalism for manipulating semi-structured data in a Web context.

The more popular approach, however, is to either actually or virtually extract the information contained within Web pages by wrappers [12, 22, 32]. A wrapper can extract content from information sources into a self-describing representation or can convert data and queries from one model to another. In the Web environment, a wrapper can convert data implicitly stored in an HTML document into data explicitly stored in a data structure for further processing. Several research teams have proposed ways to extract data using several methods, including hard-coded wrappers by declarative languages [2, 9, 21], natural language processing (NLP) [20, 43, 45, 47, 48], HTML structure analysis [10, 37, 46], inductive learning-based wrappers [6, 7, 26, 29, 43], wrappers created by example [1, 31], and regular-expression wrappers generated automatically [10].

Though all researchers report good results, the two main difficulties of traditional wrappers — resiliency and scalability — still remain. Resiliency means that a wrapper continues to function correctly even when the layout of a page changes and when new pages within the extraction ontology's domain come into existence. Scalability means a wrapper can extend its capability by adding new functions without changing the ones that already exist. For all the wrappers listed above, when a new site comes into existence or an old site is updated, we need to write a new wrapper manually or at least re-apply the wrapper generation procedure to get a new wrapper. Although some researchers claim that the generation process is easy [31], supervisors must set up and restart the generation process. Moreover, most of the previously mentioned wrappers are based on regular-pattern recognition. Thus, when a Web site changes, the change affects not only the part modified, but also the whole regular-expression pattern. Both resiliency and scalability problems make wrapper maintenance difficult [8, 30, 34].

An alternative approach, which does not have these resiliency and scalability difficulties, is the ontology-based data-extraction approach proposed by Embley et al. at Brigham Young University [11, 15]. A data-extraction ontology is a conceptual-model instance that describes a real-world application in a narrow, data-rich domain of interest. It works like a wrapper to extract structured data from unstructured or semi-structured documents. Its key feature is an extraction ontology, which is defined by a conceptual model of the application domain instead of the structure of a target Web site. This feature provides data-extraction ontologies with the capabilities of being both resilient and scalable. Because the recognizers and the schema of the concepts in data-extraction ontologies define the lexical appearance of constant objects and the relationship among objects based on conceptual modeling of the application domain instead of for some special characteristics of Web sites, people do not need to rewrite the recognizers as long as the application domain does not change. Hence the data-extraction-ontology approach solves the problem of resiliency. When people want to add a new object into the data-extraction ontology, they can just add a new concept instance recognizer and new schema components to link the new concept with the existing concepts without modifying any existing concept recognizers or schemas. Thus the data-extraction ontology approach also solves the problem of scalability.

Although the data-extraction-ontology approach has proved successful for many applications [11], one critical difficulty remains: a user must create the extraction ontology manually, which is both time-consuming and error-prone. Furthermore, data-extraction ontology writing requires a high degree of knowledge in both database theory and Perl regular-expression syntax [44]. In an effort to solve these problems, this research focuses on enabling semi-automatic (and potentially fully automatic) creation of data-extraction ontologies.

Generating data-extraction ontologies is like generating wrappers. Data-extraction ontologies, however, do not depend on the structure of the target site. Hence we cannot use the same generation techniques to create data-extraction ontologies. The motif of our approach to data-extraction-ontology generation is to simulate a typical ontology-creating scenario possibly happening inside human brains. A person creating a data-extraction ontology may first look at some sample documents of interest. After looking through these documents, a person may abstract some concepts that best match the information of interest. Then based on what is already known in the brain, the person may build a conceptual model that links the abstracted concepts together. Finally, the person generates the conceptual model in the format of a data-extraction ontology. Through this analysis, we hypothesize that a machine can simulate this human process behavior by substituting the human brain with an existing knowledge base. A machine can take sample documents as inputs. Through parsing these documents and communicating with the knowledge base, the machine can figure out the information of interest from the documents and match them with the concepts inside the knowledge base similar to what a human does in the brain. Presumably the knowledge base contains meaningful relationship among the concepts, which the system can recognize as relations among the selected concepts of interest. Finally, the system can automatically transfer the information it gathers to the format of a data-extraction ontology. Thus, in general, we can expect to build an automated ontology-generation system that can match the knowledge explicitly represented in knowledge base with the information implicitly included in the sample documents and model the gathered information as a data-extraction ontology.

Over the past several years, people have developed linguistic ontologies to represent semantic categories that are used as main organizational backbones, suitable to impose structure on large lexical repositories. For example, Maedche and Staab [39] present a Text-To-Onto ontology learning environment to discover generalized associations that occur between items in a set of transactions and to describe them at an appropriate level of abstraction. Their approach, however, is different because their focus is on NLP and on finding taxonomic conceptual *isa* relations. Although they also tried to discover other conceptual relationships, they found that their approach was too weak for automatic discovery of non-taxonomic relations even after applying complicated NLP analysis. Furthermore, their method is not applicable for our approach because we focus on Web documents, which are mostly semi-structured or structured, rather than free text. NLP usually is not applicable for semi-structured and structured text because many of the sentences are incomplete and do not follow regular grammatical patterns.

Kumar et al. [28] proposed another approach to semi-automatically construct taxonomies over the Web. They too, however, only consider taxonomic relations, and their ontology work is for constructing the relationships among different Web sites, which is more for site classification rather than data collection.

Our work links the two different fields: (1) standardization of lexical resources like linguistic ontologies, and (2) data extraction using data-extraction ontologies. By exploiting the knowledge provided by linguistic ontologies, we want to create data-extraction ontologies.

Chapter 2 describes the structure of data-extraction ontologies. Chapters 3 and 4 respectively discuss the preparation of knowledge sources and training documents. Chapter 5 discusses the details about the ontology generation process. Chapter 6 gives experimental results, and Chapter 7 presents conclusions and some possible future work.

Chapter 2

Data-Extraction Ontologies

As the core of the BYU-Ontos project [11], a data-extraction ontology works like a general wrapper to extract data from Web pages with different formats and structures but in the same topic domain. Setting the background for our subsequent discussion about our ontology-generation processes, this chapter introduces the components of data-extraction ontologies.

2.1 OSM Model Instances

OSM [18, 14] is an object-oriented model for system analysis, specification, design, implementation, and evolution. The structural components of OSM include object sets and relationship sets. An object, either lexical or nonlexical,[1] has unique identity. Object sets support the abstraction of generalization and specialization because an object set can be a superset or subset of another object set. Relationship sets support general n-ary relationships among objects, whole/part aggregations, and set/member associations. A relationship set also allows for the definition of cardinality constraints [35] among related object sets.

Figure 2.1 illustrates a graphical version of part of an OSM model instance for an obituary application domain. Each dashed box represents a set of lexical objects: `Age` and `Relationship` are examples. Each solid box represents a set of nonlexical objects: `Deceased Person` and `Viewing` are examples. Some object labels, like `Birth Date: Date`, have a colon. The part of the label to the left of a colon, or the entire

[1]Lexical objects are objects whose representations are considered to be indistinguishable from the object. All others objects are nonlexical.

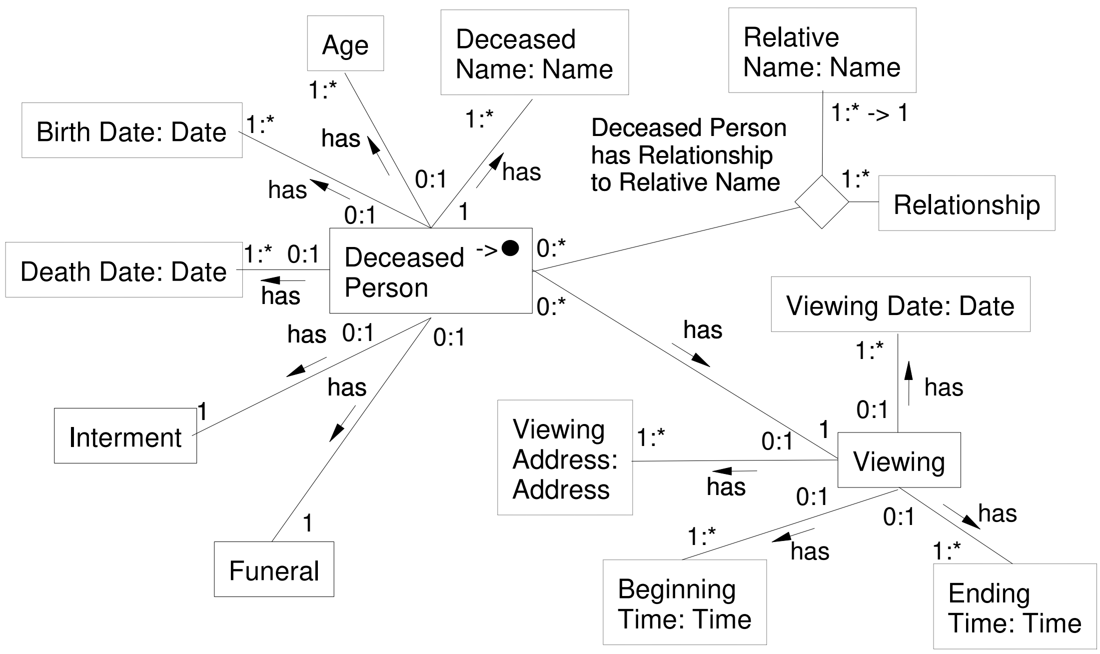

Figure 2.1: Sample OSM Model for Obituaries — Graphical Version

label without a colon, is the name of the object set. The part to the right of the colon denotes the object set that is a superset (generalization) of the object set named on the left of the colon. For example, `Birth Date: Date` represents an object set named `Birth Date` whose generalization object set is `Date`. When the colon and its right part are missing, the default generalization of the object set is `String`. For example, the generalization object set of `Age` is `String`.

The edges that connect object sets represent relationship sets, which are mostly binary but can be n-ary where $n > 2$. A binary relationship may have a label with a reading-direction arrow. For example, `Deceased Person has Age` is the name of the relationship set labelled `has` between `Deceased Person` and `Age`. A n-ary relationship set has a label that includes all related object sets. For example, `Deceased Person has Relationship to Relative Name` names the ternary relationship set among the three object sets, `Deceased Person`, `Relationship`, and `Relative Name`.

Each edge also has participation constraints for each connection. For example, the `0:1` beside `Deceased Person` in the relationship set `Deceased Person has Age` means that each `Deceased Person` participates either 0 times or 1 time in a relationship and thus has at most 1 `Age`. The participation constraint of `Age` for this relationship set is `1:*`, which means that each `Age` participates at least once in a relationship and may participate many times (denoted by *). Thus each `Age` is for 1 or more `Deceased Person`.

Besides these denotations widely used in many database studies, there are some special ones used just for extraction ontologies. A special object set in Figure 2.1, `Deceased Person`, has an arrow pointing to large dot within its box. This arrow notation specializes the model instance for a single record. For a single record, there is only one `Deceased Person`. An extraction ontology always has one and only one nonlexical object set in an OSM instance graph that is specialized to be an object for a single record. We call this object the primary object set. Often the primary object serves as the primary key for a record in the whole OSM model instance. Once specialized to be a single object, other constraints may be specialized by implication. In a single record, all `0:1` participation constraints on binary relationship sets become 1 — `Age`, for example, if present, can only be the age for the deceased person. When a specialization for the primary object set does not imply all needed constraints for a single record, we add additional specialization constraints. For example, the participation constraint for object set `Relative Name: Name` is `1:* -> 1`. This signifies that although each `Relative Name` in the object set participates one or more times in general for each single primary object, `Relative Name` participates only once. Note that we do not specialize the participation constraint for `Relationship`. Indeed, there can be many brothers, sisters, and other relatives.

2.2 Textual Data-Extraction Ontology Structure

Humans can easily understand graphical representations. Machines, however, more readily process text rather than graphical representations. Thus a data-extraction ontology has a textual version of an OSM model instance for an application domain [36]. Figure 2.2 shows part of the data-extraction ontology for the application described in Figure 2.1. The first line describes the fact that `Deceased Person` is the primary object of interest in an individual obituary. Lines 2 to 13 list the textual version of the object sets, the relationship sets and the participation constraints in Figure 2.1.

Lines 16 to 19 in Figure 2.2 illustrate the generalization/specialization between the object sets. Each object set in the list of object sets to the left of the colon is a specialization of the generalization object set on the right of the colon. Note that we need not declare the default generalization `String` because it is the default — indeed it is also the generalization for `Name`, `Date`, `Time`, and `Address`.

In order to extract data of interest from source documents, data-extraction ontologies use data frames [13] to recognize values belonging to specific object set. A data frame defines the lexical representation of objects within a lexical object set. Lines 20 to 28 in Figure 2.2 show some examples of data frames. A data frame may often contain multiple regular expressions, which are written in Perl style. Lines 21 to 23 show some regular expressions to extract dates. Also we can define lexicons, Line 24, to help to identify a list of instances.

A data frame may also include some appropriate keywords (see Line 27) that are likely to appear within the context where these objects are mentioned in the document.

```
Deceased Person [-> object];
Deceased Person [1] has Deceased Name [1:* -> 1];
Deceased Person [0:1] has Age [1:*];
Deceased Person [0:1] has Death Date [1:*];
Deceased Person [0:1] has Birth Date [1:*];
Deceased Person [0:*] has Viewing [1];
Deceased Person [0:1] has Funeral [1];
Deceased Person [0:1] has Interment [1];
Deceased Person [0:*] has Relationship [1:*] to Relative Name [1:*->1];
Viewing [0:1] has Viewing Date [1:*];
Viewing [0:1] has Viewing Address [1:*];
Viewing [0:1] has Beginning Time [1:*];
Viewing [0:1] has Ending Time [1:*];
...
...
Deceased Name, Relative Name :  Name;
Birth Date, Death Date, Viewing Date :  Date;
Beginning Time, Ending Time :  Time;
Viewing Address :  Address;
Date matches [16]
    constant { extract Month, ``\.?\s*(1\d|2\d|30|31|\d),?\s*(1[8|9]\d\d)''; },
             { extract Month, ``\.?\s*(1\d|2\d|30|31|\d)''; },
             { extract ``\d+''; context ``in\s*(1[8|9]\d\d)\b''; };
    lexicon { Month case insensitive; filename ``month.dict''; };
end;
Birth Date matches [16]
    keyword ``\bborn\b'';
end;
Viewing matches [0]
    keyword ``\bviewing\b'', ``\bmay\s+call\b'', ``\bvisitations?\b'';
end;
...
...
```

Figure 2.2: Sample Data-Extraction Ontology for Obituaries — Textual Version

Nonlexical object sets may also use keywords to limit their selection. Lines 29 to 31 shows an example of a nonlexical object set with keywords.

In order to generate a data-extraction ontology, we need to generate these object sets, relationship sets, constraints, and data frames combine them together. We present the details of the extraction-ontology generation process in Chapters 3 through 5.

Chapter 3

KNOWLEDGE SOURCES

Before generating a data-extraction ontology, we must first obtain sufficient knowledge sources. Neither we nor the end-users, however, want to build the knowledge sources by hand. Creating knowledge sources is time-consuming and is even harder than writing a data-extraction ontology manually. Instead, we want to adopt existing knowledge sources for our application directly without modification. Simple adaptation of existing knowledge sources is also acceptable. Fortunately, there are many knowledge sources available that we can either directly adopt or easily adapt.

The structure of the rest of this chapter is as follows. In Section 3.1, we define the requirements for a primary knowledge source and illustrate these requirements by describing a primary knowledge source we use in the current project. Section 3.2 discusses a unique knowledge source for this project — the data-frame library. It presents examples of data frames in the data-frame library and how the data frame library is organized. Section 3.3 describes other knowledge sources used in this project, including WordNet and additional lexicons. Finally, Section 3.4 explains how we pre-process all the knowledge sources to build an integrated knowledge base.

3.1 Primary Knowledge Source

The primary knowledge source serves as a virtual world of all known concepts. The set of object sets (or concepts[1]) in each generated data-extraction ontology is a subset of the concepts in the primary knowledge source. Thus, the primary knowledge source

[1]We use the term "concept" and "object set" interchangeably. We do so because "object set" is the standard term from extraction ontologies, and "concept" is the standard term from the knowledge sources we use.

must be general. It should not be defined specially and only for some application domain. A restricted primary knowledge source would limit the application areas that a generated data-extraction ontology could handle.

Besides concepts, the primary knowledge source must also contain meaningful relationships, which may include relationship sets among the concepts, generalizations, and specializations. Cardinality constraints and sets of instance values are also useful, and we prefer primary knowledge sources that support these features.

Moreover, we also prefer that the format of the primary knowledge sources is XML (eXtensible Markup Language) [50]. XML appears to be the best choice for our knowledge storage format because schemas, relations, constraints, and records of an OSM-based data-extraction ontology can map to and from XML easily. Furthermore, we increasingly find that people use XML for data exchange on the Web [50, 51]. Thus, it is becoming easier to find knowledge of interest in XML [4, 5], and easier to find tools to transfer knowledge to and from XML [3].

In summary, the primary knowledge source in general must be:

1. available (directly adoptable or easily adaptable);
2. general in coverage;
3. rich in meaningful relationships; and
4. encoded in (or easily converted to) XML.

The Mikrokosmos (μK) ontology [40], which was developed at New Mexico State University jointly with the US Department of Defense, satisfies these requirements. It contains over 5000 concepts, and is described as "a broad-coverage ontology." Each concept connects to an average of 14 other concepts through relationships represented in its slots. Since the μK ontology is also represented in an XML format, it fulfills all four requirements above, and we use it as a primary knowledge source in this project. Moreover, we adopt the DTD of the XML for the μK ontology as our standard input XML DTD.

Figure 3.1 shows the top-level inheritance hierarchy, which serves as the backbone of the μK ontology. All the concepts inherit from the concept ALL. OBJECTs, EVENTs, and

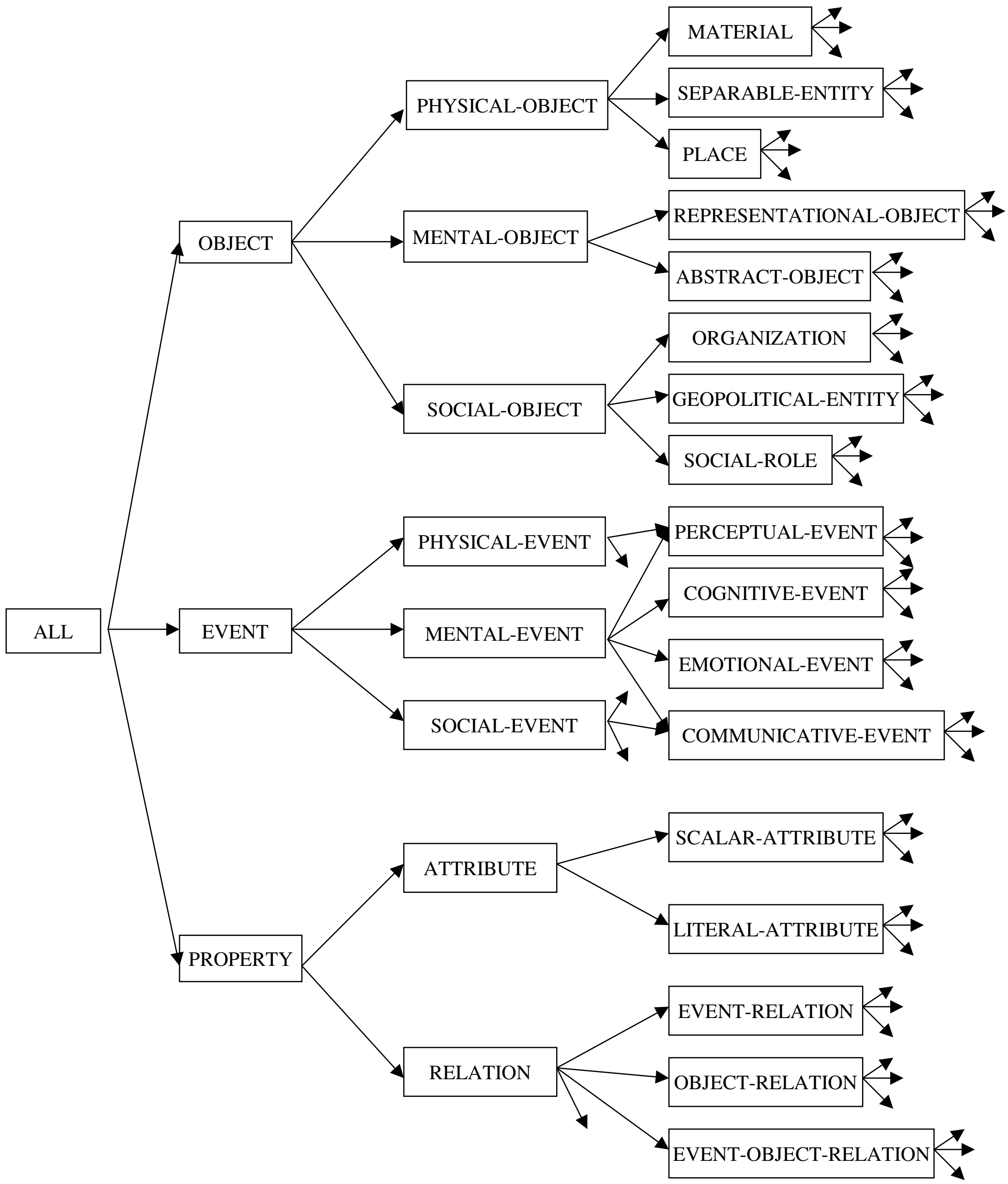

Figure 3.1: Top-Level Hierarchy of Inheritance in the μK Ontology (adopted from [40])

```
<RECORD>
    <concept>AGE</concept>
    <slot>DEFINITION</slot>
    <facet>VALUE</facet>
    <filler>How long something has existed</filler>
    <uid>0</uid>
</RECORD>
<RECORD>
    <concept>AGE</concept>
    <slot>IS-A</slot>
    <facet>VALUE</facet>
    <filler>SCALAR-OBJECT-ATTRIBUTE</filler>
    <uid>0</uid>
</RECORD>
<RECORD>
    <concept>AGE</concept>
    <slot>MEASURED-IN</slot>
    <facet>SEM</facet>
    <filler>TEMPORAL-UNIT</filler>
    <uid>4</uid>
</RECORD>
<RECORD>
    <concept>AGE</concept>
    <slot>RANGE</slot>
    <facet>SEM</facet>
    <filler>(&lt;&gt; 0 1410065408)</filler>
    <uid>0</uid>
</RECORD>
```

Figure 3.2: Example Records of the μK Ontology in XML Form

PROPERTYs constitute the concepts in the ontology. Figure 3.1 also shows the names of some of the concepts and the taxonomic relations among them.

Most of the concepts in the ontology have elaborate internal structures. Figure 3.2 shows an example of the entries for a typical node named AGE in the XML of the μK ontology. Each record between an opening and closing pair of tags, <RECORD> and </RECORD>, defines five attributes. The first attribute, <concept>, defines the name of the concept in the record. The <filler> attribute, which is the fourth one, can reference another concept or can be a literal or numerical constant. The <slot> attribute, which is the second one, defines a relationship from the <concept> attribute to the <filler> attribute. These three are the most important attributes; they reveal the fact that the structure of the μK ontology is a directed graph where both the <concept> and the <filler> represent two different nodes and the <slot> represents a directed edge from the <concept> node to the <filler> node. The <facet> attribute represents the type of the <filler>, where VALUE declares a constant or a super- or sub-class concept (the first and the second records in Figure 3.2), and SEM means that the current node either points to another concept, which is not a super- or sub-class concept, or may present some semantic meaning (the third and the fourth records in Figure 3.2). We do not use the last attribute, <uid>, in our current ontology generation process.

With this background it is possible to understand Figure 3.2. The first record in this figure presents the definition of the concept AGE, which is "How long something has existed". Following the same pattern, the second record expresses the fact that AGE inherits from the concept SCALAR-OBJECT-ATTRIBUTE, which is defined in another record. The third record shows that AGE can be measured by the concept TEMPORAL-UNIT, another μK concept that defines a set of units including YEAR, MONTH, DAY, MINUTE, and SECOND. The last record defines a range (or semantic class) of values for AGE, between 0 and 1410065408.

Because the μK ontology is represented as a directed graph, it is clear that we can convert any other knowledge sources that may be presented as a graph, either directly or indirectly, to the μK style by mapping the edges to slots and the nodes to concepts and fillers. This direct mapping is an additional benefit we get from using μK as our primary knowledge source.

Even with these characteristics, the μK ontology itself is not sufficient for extraction-ontology generation. The whole Mikrokosmos project includes not only the μK ontology, but also lexicons and related NLP capabilities. One role of the lexicons in the μK ontology is to provide concept names for representing word meanings for natural languages (including non-English natural languages). Thus, the basic μK ontology itself does not contain extensive lexical content, but rather has an associated lexicon base that supports the matching of the lexical content in input documents. Since the original μK lexicons were not

```
(1) Deposit matches [5] case insensitive
(2)     constant { extract ``[1-9]([0-9]{0,2}|,?[0-9]{3})'';
(3)                context ``\$\s*[1-9]([0-9]{0,2}|,?[0-9]{3})\b'';
(4)                substitute ``,''->``''; };
(5)     keyword ``\bdeposit\b'', ``\bdep\b'';
(6) end;
```

Figure 3.3: Data Frame for Deposit

available for this work, we have created our own separate lexicon base, which we discuss later in Section 3.3. Additionally, we cannot describe some concepts, like AGE in Figure 3.2 as a list of lexicon items because we cannot easily enumerate their values. Since the μK ontology itself provides no mechanism to handle this type of occurrence, we introduce a data-frame library to play this role, as described in the next section.

3.2 Data-Frame Library

One of the significant characteristics of our data-extraction ontology is that we use Perl regular expressions [44] to recognize values that are not finitely or easily enumerable, such as age, year, and price. These regular expressions not only overcome the difficulties of denumerability for the target concepts, but also provide some additional capability in restricting the domain of the target concept.

Data frames defined in the data-frame library are the same as in normal data-extraction ontologies as discussed in Chapter 2. Figure 3.3 shows an example of a data frame for recognizing a deposit value whose value is not greater than \$9999 in an apartment-rental application. The first line defines a concept named `Deposit`. The concept allows at most a 5-character-long string for storage. For this data frame, the extraction patterns and keywords match with the text in a manner insensitive to case, which is also the default. We may specify case-sensitive matches in other data frames. Lines 2-4 define a set of regular-expression patterns that recognize deposit strings. Line 2 defines an extraction rule to extract the target string; Line 3 defines a context restriction; Line 4 defines a string substitution condition. More precisely, for this extraction pattern, the extraction rule extracts a string, which consists of a dollar sign, some optional white space, up to five digits with

```
(1) Name  matches [80] case sensitive
(2)     constant { extract First, "\s+", Last; },
(3)     ...
(4)     lexicon {
(5)               First case insensitive;
(6)               filename "first.dict";
(7)            },{
(8)               Last case insensitive;
(9)               filename "last.dict";
(10)           };
(11) end;
```

Figure 3.4: Data Frame for Name

an optional comma preceding the last three digits. Since the substitute phrase removes the comma and the dollar sign is a context indicator, the extract phrase extracts only the digits that represents the deposit value. Line 5 defines some keywords that typically appear in the context of an apartment-rental deposit and may help identify it. The last line specifies the end of this data frame. When any regular expression of a data frame recognizes a string, the retrieved string is an instance of the concept.

Besides the regular expressions, a data frame may also contain lexicon dictionaries as in Figure 2.2. Figure 3.4 shows another example. A data frame entitled `Name` extracts people's names. `First` and `Last` in the second line refer respectively to two dictionary files: `first.dict` (Lines 5-6) and `last.dict` (Lines 8-9). The extraction pattern in Line 2 defines a rule to extract a name string composed of three parts, a first name followed by one or more white spaces followed by a last name. Each of the occurrences of the first name and last name must be lexical items from the dictionaries named `first.dict` and `last.dict`.

Within the data-frame library, we may also define data frames through an inheritance hierarchy. For instance, in a real-estate application a phone number of an agent is often an attribute of interest. An agent phone number is essentially a phone number with the restriction that it is for a real-estate sales agent. Suppose we have defined a data frame for general phone numbers named `PhoneNr`. Figure 3.5 shows that we add a line "`Agent`

```
Agent Phone :  PhoneNr;
Agent Phone matches [14] case sensitive
    keyword ``\bAgent\b'';
end;
```

Figure 3.5: Example of a Subclass Concept Recognizer in the Data-Frame Library

`Phone :  PhoneNr;`" to present the inheritance relationship. Accordingly, an `Agent Phone` is a `PhoneNr`. Moreover, we can add some keywords to distinguish an agent phone from other possible phone numbers. The presence of the keyword `Agent` helps us distinguish agent phone numbers from other phone numbers. We can also add regular expressions into subclass data frames to narrow the search space, if desired.

The structure of the data-frame library is flexible. People can freely extend the capabilities of the data-frame library either by adding new data frames or extending existing ones or describing new inheritance hierarchies without affecting any already existing data frames. We can also directly adopt any data frame into a data-extraction ontology without modification if the relevant concept is within the domain of interest.

3.3 Other Knowledge Sources

Besides the μK ontology and the data-frame library, we also use a synonym dictionary (WordNet), and lexicons to help construct new data-extraction ontologies. We mentioned in Section 3.2 how we use lexicons, but we did not say where we obtain them — a question we answer in this section.

A synonym dictionary serves a very important role in this project. As discussed in Section 3.1, our primary knowledge source may not contain enough bottom-level lexical knowledge for use in matching strings in real documents. We therefore introduced the data-frame library and the lexicons to overcome this deficiency. The synonym dictionary sits at the center of all the knowledge sources and helps link them together. The synonyms help us construct a unified knowledge base (Section 3.4) and serve in the ontology generation process (Chapter 5).

WordNet [42, 49] was developed by the Cognitive Science Laboratory at Princeton University and is a lexical database of the English language. WordNet defines a fairly exhaustive synonym lists for almost all the senses of every English word. This makes it a very good synonym dictionary for the current project. However, it is not suitable for the primary knowledge source because it lacks the relationship sets we need for generating data-extraction ontologies. Therefore, it violates the property of being "*rich in meaningful relationships*" and thus should not serve as the primary knowledge source.

As mentioned in Section 3.1, we use additional lexicons to complement the μK ontology. There are many places we can search to find useful lexicons. The rapidly growing World Wide Web is perhaps the most knowledge-rich resource we can use to gather this information. For example, if we search for "*country capital list*" on Google,[2] we can get a link to "*US State Department - Independent States of the World,*" which lists all the capital cities of the independent countries in the world. The hyponyms in WordNet may provide some lexical items as well. For example, a list of hyponyms of fish in WordNet gives a list of many of the fish names in English. In addition to lexicons listing full words or phrases, we may also have lists of abbreviations and acronyms.

3.4 Pre-processing Knowledge Sources

Before starting to generate a data-extraction ontology for any application, we pre-process the knowledge sources to reduce the burden of the generation process. In general, pre-processing semiautomatically maps lexicons and data frames to corresponding concepts defined in the primary knowledge source — the μK ontology. Many sets of the lexical items or data frames can automatically map to their related concept in the primary knowledge source by comparing the synonyms between the labels of the sets and the name of the concept itself. For example, a list of capital cities labelled *Capitals* can map to the concept named CAPITAL-CITY in the μK ontology. A data frame labelled *Population* can map to the concept named POPULATION in the μK ontology as well. But not all of the mappings can be done in this way. For instance, we are not able to map the data frame *PhoneNr* to any concept in the μK ontology. The reason is that the μK ontology does not instantiate a

[2]www.google.com

concept for phone numbers.[3] In such cases, we will manually link the unmatched lexicons or data frames to the proper locations in the μK ontology. Following the previous example, we create a new concept named PHONE-NUMBER as a subclass concept of the concept ANY-NUMBER, which is defined in the μK ontology. Then we create two links, both of whose slots are set as PHONE-NUMBER-OF, that connect from the newly created PHONE-NUMBER to the concepts HUMAN and ORGANIZATION, which are in the range of both the property concepts PHONE-NUMBER-OF and HAS-PHONE-NUMBER. Because the μK ontology is a directed graph, we also create two reverse links from HUMAN and ORGANIZATION slotted as HAS-PHONE-NUMBER to the newly created concept PHONE-NUMBER. Fortunately, this mapping process needs to be done only once.

Figure 3.6 summarizes the organization of all the knowledge sources. The μK on-

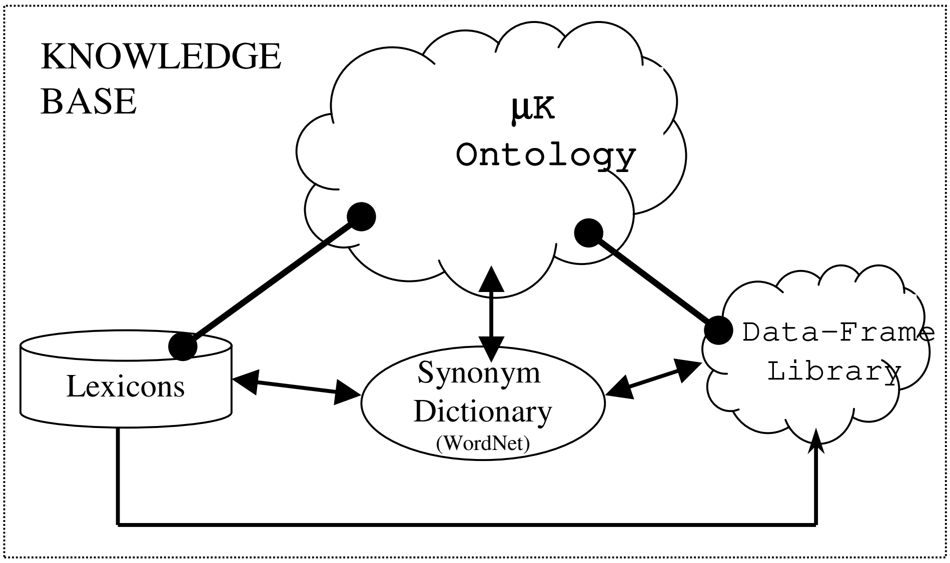

Figure 3.6: Knowledge Base Mapping Structure

tology serves as the primary conceptual structure. Since it does not contain many lexicons or recognizers itself, we establish mappings between μK and the lexicons and between μK

[3] The μK ontology does implicitly include knowledge about phone numbers because it defines two property concepts about phone number as HAS-PHONE-NUMBER or PHONE-NUMBER-OF, but phone number itself is not in the ontology.

and the data-frame library. WordNet sits at the center and serves as a synonym dictionary. For some instances, lexicons are also part of some data frames. To make use of the lexicons, we merge them into data frames and use them as explained in our discussion of Figure 3.4.

Chapter 4

APPLICATION TRAINING DOCUMENTS

The ontology-generation process needs two inputs: (1) the knowledge sources described in Chapter 3, and (2) a set of application training documents. This chapter discusses how we select and pre-process the training documents.

4.1 Assumptions for Training Documents

Training documents can be in any format and any natural language as long as the combined knowledge base (Chapter 3) contains information related to the documents. However, this project will deal only with documents written in English. Furthermore, we will only handle documents in HTML or plain text. Currently, HTML and plain-text pages are the two most popular types of pages on the World Wide Web.

By their basic assumptions, data-extraction ontologies are for data-rich documents in a narrow domain of interest [15]. Therefore, we also require that training documents be data-rich and narrow in topic breadth. If the training documents are not data-rich, we may face the difficulty of not being able to find data and thus not being able to generate an extraction ontology that covers the data of interest. If the domains related to the training documents are not narrow in topic breadth, the generated data-extraction ontology may face the problem of trying to link highly unrelated concepts.

Each training document represents a subpart of the domain of interest. The reason that it is a subpart instead of the whole domain is because there is no restriction on documents that each of them must include values of all features of the domain of interest. For example, if the domain is about car advertisements, there are some ads that do not contain mileage numbers though mileage is an important feature people like to know. But in general, the union of multiple training documents must represent the whole domain of interest. For example, as long as there is one advertisement in the collection in which a mileage number appears, we get mileage as an object set of interest. Note, however, that a training set composed of unsupervised, randomly selected training documents is not guaranteed to represent the whole domain of interest. A supervised selection process can fix this problem with the tradeoff of requiring more human effort.

4.2 Document Pre-processing

People write HTML pages in a wide variety of styles. Most often, one page presents one topic. There may be one or multiple records in one document where each record presents a parallel but different meaning with respect to other records. For example, each page in the CIA Factbook Web site[1] presents facts about one country, while an automobile advertisement in a newspaper usually contains multiple records independent of each other with each record describing one car for sale. However, some pages may only contain part, instead of all, of the information of interest. In these cases users must follow links to obtain the missing information. For example, in most of the shopping Web sites, stores list various goods available for customers to buy, but people must click on one of the listed items to view its details. Furthermore, even in the pages that include all of the information of interest, there may exist some distracting messages, like advertisements posted by sponsors.

In order to reduce the burden, and also increase the performance, of the ontology generation process, we convert the documents into a list of standard singleton records. Therefore, we decompose a page that contains multiple records into a list of individual records. We also retrieve any useful linked information to build a complete record. We also remove any extraneous advertisement to obtain a list of cleaned records. We do some of these processes automatically and others manually; automating this process is not the focus of this project. (See [17] and [19] for descriptions about how to automate this document preprocessing work.)

[1] http://www.cia.gov/cia/publications/factbook/index.html

Figure 4.1 shows the sequence of the pre-processing stages for the training documents. First we clean the HTML body by removing all irrelevant parts like advertisements

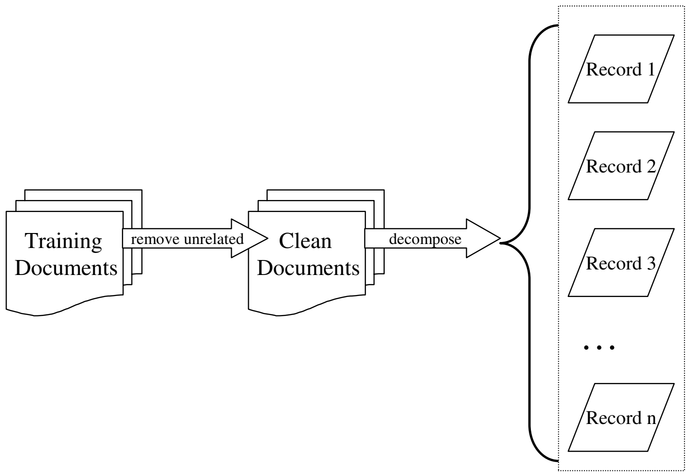

Figure 4.1: Pre-processing of Training Documents

and index frames. If there is missing linked information, we bring it into the appropriate documents. We call these documents "clean documents". We next decompose the clean documents into multiple, singleton records. Finally, we pass each record to the data-extraction ontology-generation process. In the following paragraphs, we will use examples to show how the different pre-processing stages work.

Figure 4.2 shows an example of an apartment-rental advertisement page from the *LA Weekly*. This example shows only 4 advertisements. Observe that there are sponsors' advertisements on the top of the page and two side frames on the left and right sides of the page. We thus remove the advertisements and side frames to get a clean document for this training page. Figure 4.3 shows the clean document, which contains only the desired records about different apartments for rent.

Figure 4.2: Original Training Document for the Apartment-Rental Advertisement

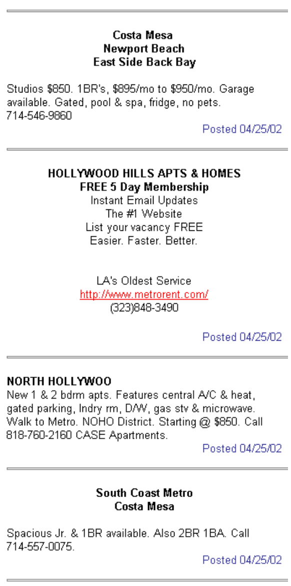

Figure 4.3: Cleaned Training Document for the Apartment-Rental Advertisement

Record 1

```
Costa Mesa Newport Beach East Side Back Bay
Studios $850.  1BR's, $895/mo to $950/mo.  Garage available.
Gated, pool & spa, fridge, no pets.  714-546-9860 Posted 04/25/02
```

Record 2

```
HOLLYWOOD HILLS APTS & HOMES
FREE 5 Day Membership Instant Email Updates
The #1 Website List your vacancy FREE Easier.  Faster.  Better.
LA's Oldest Service http://www.metrorent.com/
(323)848-3490 Posted 04/25/02
```

Record 3

```
NORTH HOLLYWOO
New 1 & 2 bdrm apts.
Features central A/C & heat, gated parking,
lndry rm, D/W, gas stv & microwave.
Walk to Metro.  NOHO District.
Starting  $850.  Call 818-760-2160 CASE Apartments.
Posted 04/25/02
```

Record 4

```
South Coast Metro Costa Mesa
Spacious Jr.  & 1BR available.  Also 2BR 1BA.
Call 714-557-0075.  Posted 04/25/02
```

Figure 4.4: Decomposed Individual Training Records for Apartment-Rental Advertisements

In addition, we can see that these records are data-rich and that the application domain focuses only on apartment rental, which is a narrow domain fulfilling our assumptions. The clean document in Figure 4.3 contains four individual records. We thus decompose it into four individual training records as Figure 4.4 shows.

Chapter 5

EXTRACTION-ONTOLOGY GENERATION

After gathering meaningful knowledge sources and training documents, we may start to generate a data-extraction ontology with respect to a domain of interest represented in the training documents. This chapter explains how we generate a data-extraction ontology from this collected information.

The structure of the chapter is as follows. Section 5.1 presents the whole picture of the project and addresses the problems of generating an extraction ontology. Then starting from Section 5.2, we give the details of a four-stage generation process. Section 5.2 explains the concept-selection process, which includes the heuristics used to recognize the candidate output concepts and the methods of conflict resolution. Section 5.3 explains the ways of retrieving relationship sets among the selected concepts from the integrated knowledge base. Section 5.4 presents the heuristics to determine the constraints among the selected concepts. Finally, Section 5.5 discusses refinements to the output ontology.

5.1 Architecture

Before explaining the details, we first describe the main idea of the ontology-generation process. Figure 5.1 shows the high-level architecture of the ontology-generation project. The project has two main parts: (1) ontology generation and (2) result evaluation. After pre-processing, we have constructed an integrated knowledge base (Chapter 3) and have collected a set of decomposed, clean records for training (Chapter 4). The ontology-generation process then applies the concept-selection, relationship-set-selection, and constraint-discovery processes in sequence to generate the data-extraction ontology automatically. Note that the training documents may also be part of the test documents.

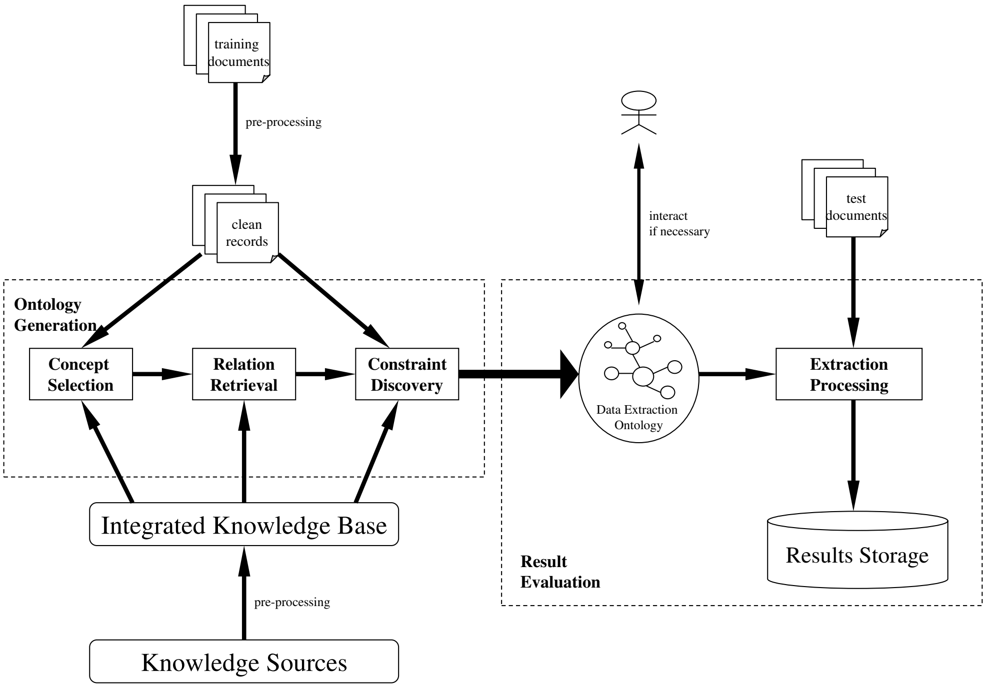

Figure 5.1: The Architecture of Data-Extraction Ontology Generation

After generating ontologies, users may evaluate the performance of the generated ontology by comparing the data extracted from the test documents by the generated data-extraction ontology with the data contained in the test documents. In addition, users may interrupt at any time to update the generated data-extraction ontology until the final result fulfills their requirements.

5.2 Concept Selection

The first step of the ontology-generation process is to select potential concepts that may appear in the output ontology based on the training records. As explained in Chapter 3, the number of concepts defined in the input knowledge sources is huge. Most of the time, we are interested in only tens out of thousands of concepts. Thus, we need to find the subset of these concepts that are of interest for the user-specific application.

5.2.1 Basic Selection Strategies

To select correct concepts from the pool of all object sets in the integrated knowledge base, the system needs to recognize which ones appear in the training records. In order to achieve this, the system matches the strings in training records using recognizers of concepts. A match indicates a potential object set that may appear in the output ontology. Note that a match here means not only an exact match of two strings, but also a match of synonyms of two strings. We use WordNet as our synonym dictionary.

We use three basic concept-selection strategies to identify occurrences of concepts:

S1 compare a string with the name of a concept;

S2 compare a string with values belonging to a concept; and

S3 apply data-frame recognizers to recognize a string.

The following examples show how the three strategies work.

In Figure 5.2, **S1** matches the string *Capital* in the training page with the concepts

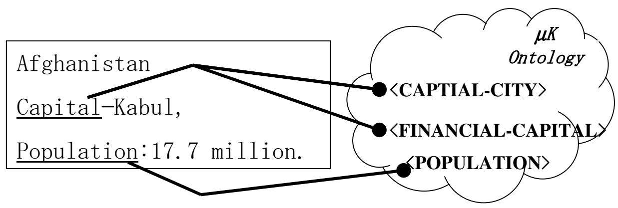

Figure 5.2: Concept Matching Examples by Selecting Names from the μK Ontology

CAPITAL-CITY and FINANCIAL-CAPITAL in the μK Ontology. Similarly, the string *Population* matches with the μK concept POPULATION. Thus, the three object sets, CAPITAL-CITY, FINANCIAL-CAPITAL and POPULATION, are potential candidates for selected object sets.

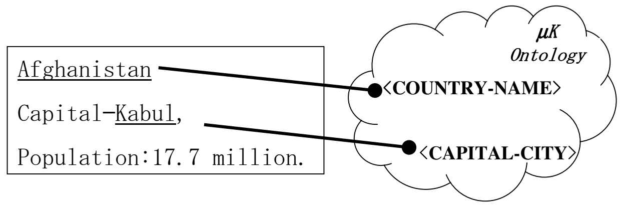

Figure 5.3: Concept Matching Examples by Selecting Values from the μK Ontology

Figure 5.3 shows the kind of matches **S2** provides. *Afghanistan* is an instance of a name of a country and *Kabul* is an instance of a capital city. Hence, we select the object sets COUNTRY-NAME and CAPITAL-CITY as potential candidates.

Figure 5.4 illustrates **S3**, which selects concepts based on regular-expression recognizers in the data-frame library. Two data frame patterns match the "17.7 million" in

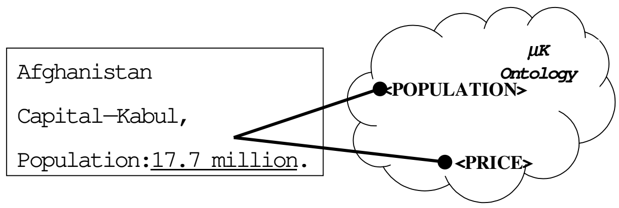

Figure 5.4: Data-Frame Matching Example

the training record. One pattern associates with the concept Population, and the other associates with Price. Note that the knowledge sources preprocessing has already integrated the concepts in the data-frame library with the concepts in the μK Ontology. We, thus, add the associated two μK concepts into the potential candidate list of object sets.

5.2.2 Conflict Resolution

Clearly, matches occasionally entail incorrect conceptual interpretations. For example, in Figure 5.2, **S1** selects both FINANCIAL-CAPITAL and CAPITAL-CITY, but only CAPITAL-CITY is the correct selection since we are clearly in the context of countries, not money. The conflict resolution step attempts to resolve these incorrect matches.

Two natural realities cause the conflicts: (1) synonymy (different words or phrases with the same meaning) and (2) polysemy (one word or phrase with different meanings). For example, the word *capital* means both "capital city" and "financial capital" (polysemy); and both *capital* and *money* mean "financial capital" (synonymy).

To resolve conflicts, we have studied three heuristics, which are (1) *same-string only one meaning*, (2) *favor longer over shorter*, and (3) *context decides meaning*. We discuss these three heuristics in the following paragraphs.

Same-string–only one meaning. One string in a training record may match an instance of more than one concept. Such a string, however, can have only one meaning for each occurrence. Therefore, only one concept can be the correct selection. Practically, if the heuristics **S1**, **S2**, and **S3** confidently identify a concept to be a correct selection, any other concept that conflicts with this concept by the same string is considered to be an error. Actually, we use the weaker condition that if in one training record, both **S1** and **S2**, or both **S1** and **S3** select a concept, we accept it as a correct selection.[1] Any other concept that recognizes the same string then is considered to be an error, and the conflict-resolution procedure removes it from the list of candidate concepts. Figure 5.5 illustrates this heuristic. Although both FINANCIAL-CAPITAL and CAPITAL-CITY match "Capital" in the training record by **S1**, we know we cannot have both. Since CAPITAL-CITY also matches "Capital" by **S2**, we mark CAPITAL-CITY as a correct selection. Figure 5.6 shows another example. Although **S3** suggests both the concept `Population` and Price for the string "17.7 million", the system accepts the concept `Population` as the correct selection based on the additional suggestion from **S1** because of the string "Population".

[1]Theoretically, exceptional cases may exist. However, we have not encountered any exceptions during our experiments.

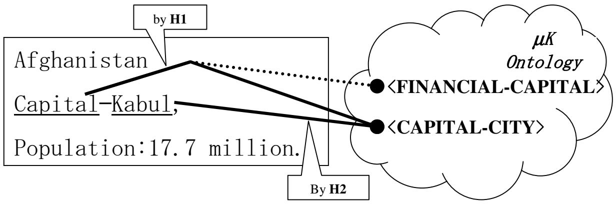

Figure 5.5: Same-string–only One Meaning, **S1** and **S2**

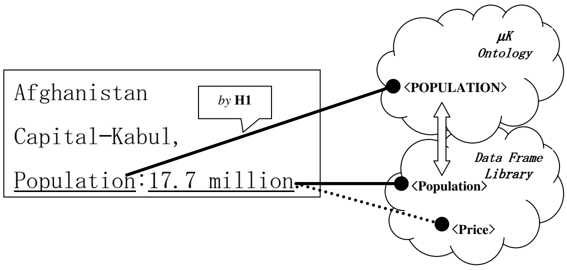

Figure 5.6: Same-string–only One Meaning, **S1** and **S3**

Favor longer over shorter. If concept A recognizes string A' and concept B recognizes string B', where A' is a proper substring of B', the selection process chooses concept B over A because the meaning of B' overrides the meaning of A'. In Figure 5.7, **S2** rec-

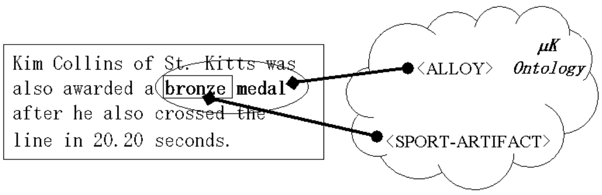

Figure 5.7: Favor Longer Over Shorter

ognize the word "bronze" as an instance of ALLOY, while **S2** matches the whole string, "bronze medal" as an instance of SPORT-ARTIFACT. Because "bronze" is a proper substring of "bronze medal", the system chooses the concept SPORT-ARTIFACT instead of the concept ALLOY.

Context decides meaning. Sometimes, we can use the context around a target string to identify one of multiple matching concepts for a string. Figure 5.8 shows an example. In

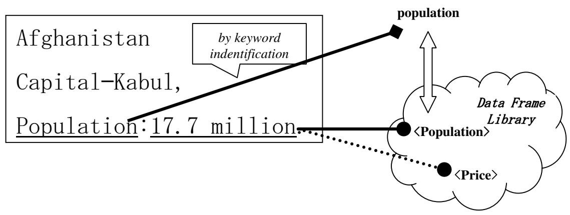

Figure 5.8: Context Decides Meaning

order to decide between the potential candidates Population and Price, we use an associated keyword defined in the data-frame library (Section 3.2) as a context keyword. Since we find the keyword "Population" close to the occurrence of a data value for the concept Population and do not find any context keywords of the concept Price, such as "price", "cost", "GNP", "amount", "$", or "dollars", the concept `Population` has a higher probability of being the correct match.

5.3 Relationship-Set Selection

To generate the relationship sets among the selected concepts, we need to find them in the knowledge base. As we have seen in Chapter 3, the primary knowledge source, which is the μK Ontology in this project, stores information about relationship sets among object sets. This section discusses how we find them.

5.3.1 Theoretical Resolution Difficulty

The structure of the μK Ontology is a directed graph where nodes are concepts. If we can determine the relationship sets among the concepts identified to be of potential interest for our application, we can form a subgraph of the μK Ontology representing the application of interest. Theoretically, a solution is to find all possible paths among the concepts, but this solution is too expensive to execute because to find all paths is an NP-complete problem. As a straightforward proof, we point out that to find the longest path in a graph is a well-known NP-complete problem[2] [25]. If finding all paths could be solved in polynomial time, by sorting the results in polynomial time, finding the longest path could also be solved in polynomial time — a contradiction. We therefore seek a heuristic solution to find an appropriate subgraph among the identified concepts.

5.3.2 Heuristic Resolution

Considering our problem, we can reason that it is almost never necessary to find all the paths. Multiple paths present different relationships between concepts, but for an extraction ontology, we typically only want one. Usually, the shortest path represents the most desired relationship between two concepts. Dijkstra's algorithm is a well-known way to find the shortest path between any two nodes, and it has polynomial time complexity. In addition, if the shortest path between two concepts is too long, most likely the relationship between them either is very weak, or happens rarely, and is likely not to be of interest. Thus, we set a threshold distance to omit these occurrences.

Figure 5.9 gives the Find Binary algorithm to find binary relationship sets. It is a breadth-first search algorithm to generate binary relationships between concepts selected by the concept-selection procedure. We denote those selected concepts as SNODE in the algorithm (Figure 5.9). Because the algorithm follows a first-meet-first-retrieve strategy, it retrieves the shortest path between the two SNODEs.

In the Find Binary algorithm, we first initialize NL, a list of nodes containing the traversed nodes, as an empty list. For each SNODE, we find all the retrievable relationship sets that are the paths with a distance between this SNODE and the other SNODE less than the threshold distance (ThresholdDistance in Figure 5.9). One difficulty of this breadth-first-search approach is the inheritance hierarchy defined in the μK Ontology. All concepts, except ALL, are inherited from superclass concepts.[3] Therefore, if concept A has a relation with concept B, all the subclass concepts that inherit from concept A have relations with

[2]There is even a song singing about this using the tune "The Longest Time." See Appendix A.
[3]Multiple inheritance is permitted in the μK Ontology.

```
PROCEDURE FIND-RELATIONSHIP(Graph)
    initialize the NodeList (NL) as an empty list
    FOR each node marked selected (SNODE)
        set the StartNode as this SNODE
        FIND-SUPERCLASS-NODES(SNODE)
        set the distance of each returned node to 0
        add this SNODE and the returned nodes to NL
        set CurrentDistance to 0
        WHILE CurrentDistance less than ThresholdDistance AND more nodes in NL
            FOR each node in NL with CurrentDistance
                FOR each LinkedNode of this node
                    IF the LinkedNode is an SNODE
                        set this LinkedNode as the EndNode
                        generate a binary relationship set between the StartNode and the EndNode
                    ELSE IF the Node represents literal or numerical constant
                        indicate an end of a branch
                    ELSE
                        set the distance of the LinkedNode (DisL) to be CurrentDistance plus 1
                        IF the distance less than ThresholdDistance
                            FIND-SUBCLASS-NODES(LinkedNode)
                            FOR each returned node
                                set the distance of this node to DisL
                                IF this node NOT in NL
                                    add this node into NL
            advance CurrentDistance by 1
    RETURN all the relationship sets retrieved

SUBPROCEDURE FIND-SUPERCLASS-NODES(GraphNode)
    RETURN List of GraphNodes that are superclass-nodes of the input GraphNode
    (Note that the returned nodes do not include the first two levels of objects — ALL,
    OBJECT, EVENT, and PROPERTY — because they are very general category definitions
    that do not establish useful relationships for the applications.)

SUBPROCEDURE FIND-SUBCLASS-NODES(GraphNode)
    RETURN List of GraphNodes that are subclass-nodes of the input GraphNode
```

Figure 5.9: Find Binary Algorithm

concept B, and vice-versa. Line 5 of Figure 5.9 obtains all the superclass concepts of the starting node. The starting node (StartNode) and all these nodes are inserted into NL as a queue.

The rest of the algorithm is as follows. Lines 9 to 25 constitute the loop to retrieve the relationship sets. Every node in NL has an attribute recording the distance between that node and the StartNode. All the superclass nodes of the StartNode have distance 0. All the nodes that link directly with the nodes of distance 0 have distance 1, and so on. The WHILE-loop starting at Line 9 terminates when either the CurrentDistance is larger than the ThresholdDistance or all the nodes in NL have been exhausted. Three cases can happen when the algorithm evaluates each LinkedNode in Line 11. If the LinkedNode is another node that has been selected by the concept-selection process, we generate a binary relationship set between this node and the StartNode and terminate this branch (Lines 12 to 14). Lines 15 to 16 show that the branch terminates when the LinkedNode is a literal or numerical constant, which must be a leaf node in the graph. For all other cases, the system sets the distance of these nodes as 1 plus the distance of current node. Again, if the current node links to the LinkedNode, the current node links to all the subclass nodes of the LinkedNode with the same distance. Thus, the system adds the LinkedNode and the subclass nodes of the LinkedNode to NL. Finally, the system advances CurrentDistance by 1 and continues the WHILE-loop starting at Line 9 until the loop terminates.

5.3.3 Largest Subgraph

After concept selection, conflict resolution, and binary-relationship creation, we will have generated one or more connected subgraphs. Disconnected subgraphs likely represent different application domains. Since we focus only on narrow domain applications and assume training documents represent this domain, we expect only one subgraph. Since we also assume that the training documents are data-rich, we can assume that the largest subgraph of concepts is the subgraph of interest. This subgraph becomes the conceptual model instance for the extraction ontology we generate.

We use a standard algorithm to find the largest subgraph, which is the subgraph containing the largest number of concepts. Theoretically there could be two largest subgraphs with the same size. This situation should rarely happen; indeed, usually there should be one large subgraph and either no other subgraphs or maybe a few other small subgraphs, often having only one node and no edges.

After we have selected the largest subgraph, we remove all concepts not belonging to that subgraph from the list of selected concepts. At the same time, we remove all retrieved binary relationship sets connecting eliminated concepts.

5.3.4 Primary Concept

To build correct schemas for data-extraction ontologies, we need to find out which one of the concepts is the primary concept, or the primary object set. Often the primary object set is at the center, having connections to most of the other object sets in the OSM graph. Thus, the primary object set often has the highest degree of connectivity among all the object sets in the OSM graph. As a heuristic, we therefore find or create a nonlexical concept with the highest degree of connectivity in the subgraph obtained as explained in Section 5.3.3. The node with the highest degree of connectivity can be: (1) a nonlexical node, (2) a lexical node, or (3) an intermediate node. We discuss these three possibilities separately.

First, if the node with the highest degree of connectivity is a selected nonlexical object set, the system directly takes it as the primary object set.

Next, if the node with the highest degree of connectivity is a selected lexical object set, the system creates a new nonlexical object set to be the primary object set. The new nonlexical object set takes the place of the former lexical object set and thus become the node with the highest degree. The system connects the original lexical object set to the new nonlexical object set through a new 1-1 relationship set. The new nonlexical object set thus becomes the root of the OSM graph. Figure 5.10 shows the idea of creating such a mapping. The object sets with solid borders are nonlexical object sets; others are lexical object sets. In the left side of this figure, the lexical object set D has the highest degree of connectivity. To make the primary object set, the right side of Figure 5.10 shows that the system creates a new nonlexical object set and lets it take the place of the original lexical

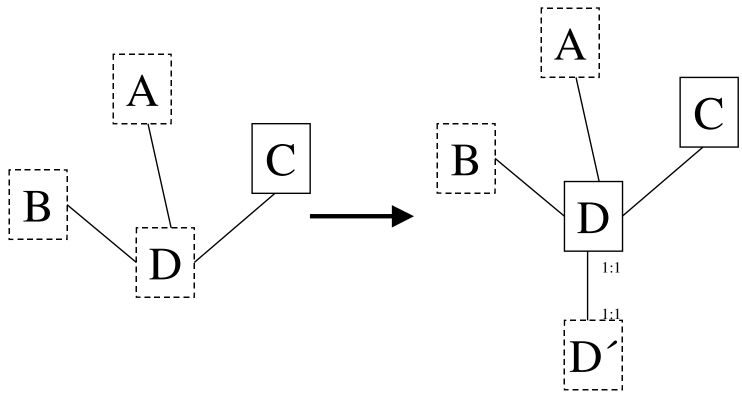

Figure 5.10: Primary Object-Set Creation Based on Lexical Object Set

object set D. The system then renames the original D to be D' and makes it have a one-to-one relationship with the newly created lexical object set D. Finally, the system makes the new nonlexical object set D the primary object set.

In our last case, if the node with the highest degree of connectivity is neither a selected lexical object set nor a selected nonlexical object set, it must be an intermediate node between some selected object sets. The system then directly marks this node as a selected node and makes it the primary object set. The system always makes this new intermediate object set nonlexical even if it actually is a lexical concept in the knowledge base.

5.3.5 Multiplicity Problem

Until now, all the retrieved relationship sets have been binary. This is because all the relationship sets in our primary knowledge source, the μK ontology, are binary. Sometimes, however, n-ary relationship sets are more representative for relationships among more than two object sets. Figure 5.11 illustrates an appropriate resolution for generating an n-ary relationship set from multiple unselected binary relationship sets. For any set of concepts whose size is greater than two, if an intermediate, unselected concept links with selected concepts, we may establish an n-ary relationship set among all these concepts.

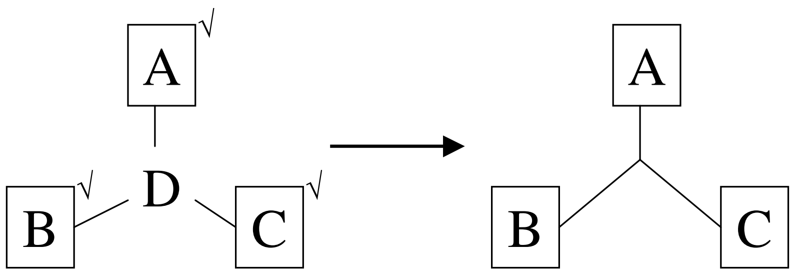

Figure 5.11: Transformation to a Ternary Relationship Set

Note that if there is no intermediate node between any two nodes, the distance between those two nodes is 1, which means the knowledge base defines the two object sets as being directly related to each other. Furthermore, since we have assumed the relationship sets are meaningful in the knowledge base, we can reasonably say that the binary relationship sets with distance 1 are important, so that we must preserve them in the generated data-extraction ontology.

The resolution in Figure 5.11 may generate many duplicate n-ary relationship sets and may establish many n-ary relationship sets that are subsets of some other n-ary relationship sets. In order to avoid the duplicates and subsets, we propose the following procedure to produce the final relationship sets.

(1) We retrieve all the binary relationship sets by the Find Binary Algorithm in Figure 5.9.

(2) We finalize all the binary relationship sets that directly connect two selected object sets and make them part of the generated data-extraction ontology.

(3) If Step 2 has not included all the selected concepts, the system needs to generate n-ary relationship sets. (Note that here $n \geq 2$.) In order to do so, the system first sorts all the intermediate nodes in the subgraph by the degree of connectivity.

(4) The system then selects an intermediate node N with the highest degree of connectivity that links with at least one of concept nodes not included in the finalized relationship sets. The system then finalizes an n-ary relationship set including all the selected concept nodes linking with N. The system repeats Step (4) until all the selected concepts are included in at least one of the finalized relationship sets.

5.4 Constraint Discovery

Schemas for data-extraction-ontologies require participation constraints. A participation constraint consists of two parts: a minimum number and a maximum number, which respectively specify the minimum and the maximum times an object in an object set participates in a relationship of an associated relationship set.

A straightforward solution is to retrieve participation constraints from the knowledge base if it contains the information about participation constraints. Unfortunately, the μK Ontology defines no participation constraints, nor do any of our other knowledge sources.

An alternate solution is to gather the information about participation constraints from the training records. In order to do so, the system takes each finalized relationship set discussed in the previous section and computes the number of times each object in that relationship set. Based on the following participation-constraint-generation rules, the system discovers the participation constraints for different object sets within their respective relationship sets. To simplify the discussion, we first show how to obtain participation constraints for binary relationship sets. Later, we show how to extend the idea for n-ary relationship sets.

When the primary concept P is one of the object sets in a relationship set, we treat this situation special because there is one and only one occurrence of the primary concept in any one training record. We use A to denote the object set that relates with P to constitute a binary relationship set. Since any instance[4] of A must appear one or more times in at least one record, the system sets the minimum participation constraint of concept A in the relationship set as 1. Thus, [1:?] is the participation constraint for the object set A in the relationship set, where "?" denotes "not yet known but some other rule will decide it." Since a training record may or may not contain an instance of A, it is not necessary that each instance of the primary concept participates with an instance of concept A. This may happen even more frequently for Web documents because they are not necessarily well-structured, and the authors of Web documents may omit some features. Hence, it is safe and correct to set the minimum participation number for the primary concept as 0 instead

[4] An instance here can either be a lexical appearance or a nonlexical appearance.

of checking whether the other concept A actually occurs at all training records. Thus, the system always sets the participation constraint for the primary concept P as [0:?].

For other binary relationship sets that do not contain the primary concept, we count the minimum number of instances of each object set occurring in all the training records to set the minimum participation number of each concept. Suppose a binary relationship set contains two object sets, A and B. If the objects of concept A appear at least once in any record and the objects of concept B appear in some but not all of the records, then for each object of B, there is at least one corresponding object of A; while for each object of A, there may or may not be a corresponding object of B. The system thus sets the minimum participation number for concept A to 0 and for concept B to 1. Thus, the participation constraint for such a binary relationship set between A and B is [0:?] for A and [1:?] for B.

In order to generate the maximum participation number, the system counts the maximum number of instances that appears for each concept over all the training records. As before, we assume there is a binary relationship set that contains the two object sets, A and B. There are three cases in this situation.

Case 1. For any one record, the maximum number of instances of concept A is less than the maximum number of instances of concept B. This means the pairs (a1, b1), and (a1, b2) exist, where a1 is an instances of A and b1, and b2 are instances of B. At least one instance of concept A relates to more than one instance of concept B. The system thus sets the participation constraint for A to be [?:*], where "*" represents an arbitrary, unknown maximum number. If at the same time, there is another record that the maximum number of instances of concept B is less than the maximum number of instances of concept A, the system also sets the participation constraint for B to be [?:*].

Case 2. For each record, the maximum number of instances of concept A is less than the maximum number of instances of concept B. In this case, the system needs to examine every pair of instances to decide the maximum participation constraint of B. The reason is that even though in general the total number of instances of A is less than the total number of instances of B, there may be a case where an instance of B relates to more than one instance of A. If this is the case, the system must set the participation constraint for B

to be [?:*] as well as for A. Otherwise, the system sets the participation constraint for B to be [?:1], which means each instance of B uniquely refers to an instance of A.

Case 3. For each record, the maximum number of instances of concept A equals the maximum number of instances of concept B. The system sets the participation constraint for both A and B to be [?:1]. Note that theoretically we may need the same examination as for Case 2 to check each pair of occurrences. However, such an examination is a time consuming process. When there are dozens of training records and the case holds for all them, the probability of mis-generation is very low.

We may apply the similar participation constraint generation rules to compute the constraints for n-ary relationship sets, where $n > 2$. The only difference is that the computation now bases its number of set instances on a set of object sets that includes all the object sets in the n-ary relationship set except A instead of on the number of instances for one other object set B. Therefore, as the relationship set grows larger, computation time reaches about $O(n!)$ time complexity and $O(n^2)$ space complexity to find the participation constraints.

In addition to the computation overhead, there are three other reasons of why evaluation rules for n-ary relationship sets may not be best. First, the n-ary relationship sets happen comparatively rarely with our current knowledge base. The percentage of generated n-ary relationship sets for our experiments was less than 4% of all of the generated relationship sets. Second, the desired concept nodes are usually located close together in the μK Ontology. In our experiments, when we did generate an n-ary relationship set, it included over 80% of all the selected concept nodes. Since the computation overhead is $O(n!)$ time complexity, the generation of the n-ary relationship set was costly. The last reason is that based on our experiments, even when we added the computation overhead for cardinality constraints on n-ary relationship sets into the system, we did not achieve higher accuracy. Since we were unable to improve system performance, we decided not to expend additional effort on this problem.

Therefore, the system applies a simple rule to decide the participation constraints for n-ary relationship sets at this time. If the primary concept P appears in an n-ary relationship set, the system sets [0:*] as the participation constraint for P and [1:*] for all other object sets. Otherwise, all the object sets in the n-ary relationship set get the participation constraint [0:*].

Figure 5.12 shows an example of discovering a 1-to-1 participation constraint using training records. Suppose that we have two training records as this figure shows. For each

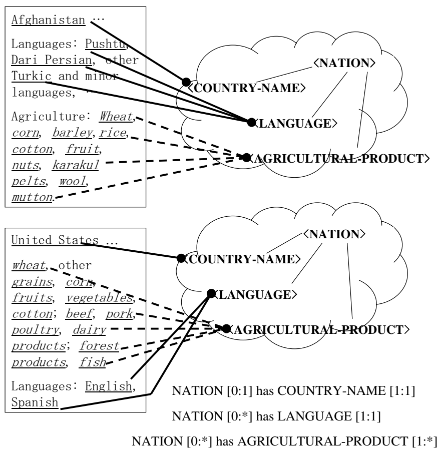

Figure 5.12: Participation-Constraint Generation Example

record, we recognize three concepts: `COUNTRY-NAME`, `LANGUAGE` and `AGRICULTURAL-PRODUCT`. By the primary concept resolution process we determine that `NATION` is the primary concept of this application. We retrieve three binary relationship sets: `NATION has COUNTRY-NAME`, `NATION has LANGUAGE`, and `NATION has AGRICULTURALPRODUCT`. Since `NATION` is the primary concept, the minimum participation constraint for `NATION` for each relationship set is 0, and each minimum participation constraint of the object sets in each of the three relationship sets is 1. By construction, the primary concept appears only once in each record. The concept `COUNTRY-NAME` also appears only once in each record, and for a different instance of `COUNTRY-NAME` appearing in each record. Thus, the system sets the maximum participation constraint for both concepts to 1. Hence, the system generates `NATION[0:1] has COUNTRY-NAME[1:1]`.

For the relationship set, `NATION has AGRICULTURAL-PRODUCT`, the system generates the maximum participation constraint for `NATION` to be * because each record contains more than one instance of `AGRICULTURAL-PRODUCT`. However, we can find at least one instance "wheat" in both the records. That is, the same instance of `AGRICULTU-RAL-PRODUCT` can be found for different instances of `NATION`. The system therefor also generates the maximum participation constraint for `AGRICULTURAL-PRODUCT` to be *. Hence, the system generates `NATION[0:*] has AGRICULTURAL-PRODUCT[1:*]`.

For the relationship set `NATION has LANGUAGE`, Record 1 has three instances of `LANGUAGE` and Record 2 has two instances of `LANGUAGE` while none of these instances is the same. Thus, each `NATION` can have more than one `LANGUAGE`, and each `LANGUAGE` belongs to only one `NATION`. Hence, the system generates `NATION[0:*] has LANGUAGE[1:1]`.

Note that this last example is not totally correct. People know that more than one nation can use the same language. For example, both the United States and the United Kingdom use English as their native language. Too small of a training set may cause significant bias because of the lack of sufficient examples. Thus, we need a large enough collection of examples to generate correct participation constraints. However, even a huge unsupervised selected training base may not totally solve this problem. For example, we know that mileage itself cannot uniquely identify a used car. But practically it is not common for two cars to have the same mileage. Supervised selection of training documents can overcome this problem, but the tradeoff requires more human effort.

5.5 Data-Extraction Ontology Refinements

After finding the object and relationship sets of interest and deriving participation constraints, it is straightforward to generate the initial data-extraction ontology. The matched attributes within the subgraph and their connecting relationships define the structure for the extraction ontology; their respective regular expressions become the regular expressions for the data frames in the extraction ontology; and the generated participation constraints become the participation constraints in the ontology. This, however, may not be the final version of the data-extraction ontology that fulfills an end-user's requirements. Extraction-ontology producers may want to update the generated data-extraction ontology.

Designers can modify the data-extraction ontology using any normal text editor since the output is a plain ASCII file without any encoding. Modifications require that the designer knows the syntax of the data-extraction ontology. Basically, end-users may refine the generated ontology in four ways; they can: (1) remove some unexpected object sets or relationship sets from the ontology, (2) rename object sets and relationship sets, (3) modify constraints, and (4) add object sets, relationship sets, or constraints. Note that our goal is to avoid the last modification as much as possible. To remove or to modify a generated item is much easier than to add a new one to the ontology. There are already some tools available to help build or update an existing ontology [23], but integrating and modifying these tools to fit our needs for this project are beyond the scope of this thesis.

Chapter 6

EXPERIMENTAL ANALYSIS AND RESULTS

After generating a data-extraction ontology through the ontology-generation process, we want to evaluate how well the generation process performed and how good the results are. This chapter discusses the experimental results and the measures we use to evaluate these results.

6.1 Threshold Distance

As discussed in the Find Binary Algorithm (Figure 5.9), the threshold distance is an important characteristic for the ontology-generation process. It significantly affects the running time of the process and the accuracy of the generated data-extraction ontology. We thus need to find a suitable threshold distance to balance the running time and the performance of the ontology-generation system.

From the μK Ontology, we have generated data-extraction ontologies for three different application domains: automobile sales, nation information, and apartment rentals. Figure 6.1 shows the running time with respect to different distance thresholds for these three applications. We can see that the running time for generating data-extraction ontologies increases significantly when the distance threshold is less than 3, and thereafter starts to maintain a steady state. Moreover, for these three applications, Figure 6.2 shows that when the distance threshold is 4, the ontology-generation process has generated over 99% of what could have been generated from the knowledge base. These two highly correlated figures suggest that the best distance threshold for balancing the running time and the system performance is 4. Hence we evaluate all the following performance comparisons based on this distance threshold.

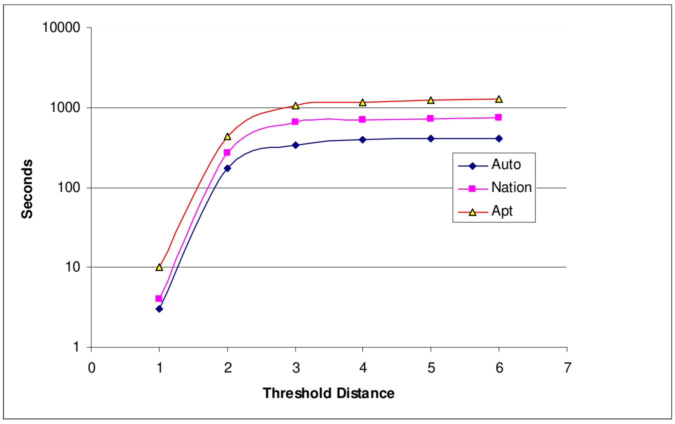

Figure 6.1: Running Time with Distance Threshold up to Six

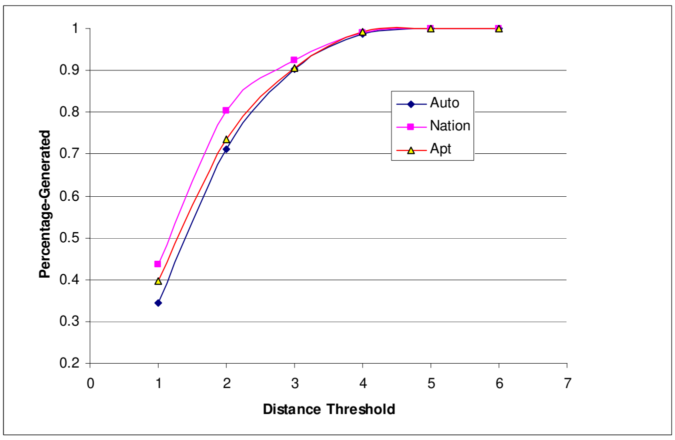

Figure 6.2: Percentage Generated for Different Distance Thresholds up to Six

Record 1: 1992 FORD F150, XLT Flair-side Ext.Cab, Excellent condition. Loaded. All power, Towing pkg, Auto trans, Captain seats, Sunroof, bedliner, dual tanks, AC, CC, tilt, Am/FM Cassette, wired for CD changer. , $9,000.00, (801) 261 -1936 Location: Murray, UT

Record 2: 1992 CHEVROLET CAVALIER RS, Gray interior. Good condition. Pwr Windows & Door Locks. Cruise, Tilt. Email: ixlr8colo@aol.com or call listed phone number. color: Black mileage: 66835, $4,399.00, (970) 226-1625 Location: Ft. Collins, CO

Figure 6.3: Sample Training Records for Automobile Sales Domain

The rest of this chapter is as follows. First, we consider a comparatively simple and previously well-studied application domain, *automobile sales*, to see how the generation-results work compared with a well-tested, expert-generated, data-extraction ontology. At the same time, we discuss the problems with this example in detail. Then we present more experimental results for the other application domains. Finally, we discuss one more experimental result using SALT[1], in addition to the μK as our primary knowledge source.

6.2 Experimental Results on the Automobile Sales Application

As an experiment to test how well our data-extraction-ontology generation works, we first apply our system on a previously well-studied, comparatively small application domain – automobile sales [11, 15, 16]. In order to do so, we must find some sample training documents, which are common on today's Web sites. As discussed in Chapter 3, we should first pre-process the training records to get the cleaned training records. Figure 6.3 shows two training records, both from our DEG demo sample Web pages [11].

There are two types of performance measurements in this project: (1) performance of the extraction-ontology generation and (2) performance of generated ontology for extracting data. Performance of extraction-ontology generation evaluates how well the system generates a data-extraction ontology, while performance of the generated ontology evaluates how well a generated extraction-ontology extracts information from documents. Since the goal of this thesis focuses on how to generate an ontology, we are more interested in the first performance measurement.

[1]Standards-based Access service to multilingual Lexicons and Terminologies (SALT), funded under EU Fifth Framework IST/HLT 3.4.1, addresses issues of international cooperation, standards for coding and interchange of linguistic data, and the combining of technologies. Refer to www.ttt.org/salt/index.html for details.

For convenience, we define several terms often used in following discussion. An item in a data-extraction ontology is either a concept, or a relationship, or a participation constraint. There are two basic measures, POG (Precision of Ontology Generation) and ROG (Recall of Ontology Generation), for evaluating the data-extraction ontology generation process. Informally, POG is the percentage of "correctly" generated items. ROG is the percentage of "correctly" generated items out of the all possible items in the knowledge sources that should have been generated with respect to the training records.

Before discussing the details of the performance metrics of the precision and recall measures, we need to point out that the evaluation of ROG is sometimes impossible because the knowledge base can be huge, having too many concepts and relationships among these concepts to evaluate manually. We therefore use a $PROG$ (Pseudo-Recall of Ontology Generation) value instead of a "real" ROG value to evaluate the recall of the data-extraction ontology-generation process. To evaluate $PROG$, we require a human expert to create a data-extraction ontology separately based on the same training records the system uses. We assume that the expert-created ontology is the best we can achieve, and we evaluate $PROG$ as the percentage of "correctly" generated items as compared with the number of expert-created items.

Therefore, the formal performance metric of POG and $PROG$ are

$$POG = \frac{number\ of\ items\ correctly\ generated}{number\ of\ items\ generated}$$

$$PROG = \frac{number\ of\ items\ correctly\ generated}{number\ of\ items\ in\ the\ expert-created\ ontology}$$

There is one more issue that affects the correctness of evaluating ontology-generation recall values. Because the system cannot generate any concept or any relationship set that is not inside the knowledge base, our recall evaluation should not count the ones that may be important from an expert's point of view but actually are impossible to generate from the μK ontology because the knowledge base does not include them. We thus define another factor, $EPROG$ (Effective-$PROG$), as follows:

$$EPROG = \frac{number\ of\ items\ correctly\ generated}{number\ of\ items\ in\ the\ expert-created\ ontology\ that\ could\ be\ generated}$$

Note that both $PROG$ and $EPROG$ have their own physical meanings. While $EPROG$ values present the real recall values of the ontology-generation process for the specified domain, the difference between $PROG$ and $EPROG$ indicates how well the knowledge base defines the specified domain, or how suitable the program is for the specified domain given current knowledge sources.

Since the system applies three different sub-processes to obtain object sets, relationship sets, and constraints, we measure three POG, $PROG$, and $EPROG$ triples to evaluate how well the three sub-processes work separately. We also give an overall performance measure that combines the three sub-measures together by weighting the three different sub-processes equally and averaging the summations.

The automobile sales application domain has been previously well-studied by the BYU Data Extraction Research Group [11, 15, 16]. Although the structure of this application domain is comparatively simple (see Figure 6.4), it is good for testing the ontology-generation procedure and illustrating problems.

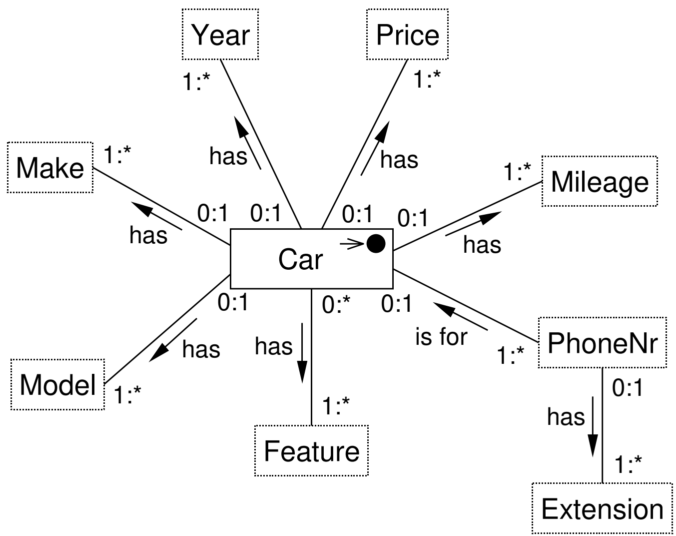

Figure 6.4: OSM Diagram for the Expert-Created Automobile Sales Application

Figure 6.5 and Figure 6.6 show the schema parts of the textual versions of the expert-created and system-generated automobile data-extraction ontology respectively.

```
(01) Car [-> object];
(02) Car [0:*] has Feature [1:*];
(03) Car [0:1] has Make [1:*];
(04) Car [0:1] has Mileage [1:*];
(05) Car [0:1] has Model [1:*];
(06) Car [0:1] has Price [1:*];
(07) Car [0:1] has Year [1:*];
(08) PhoneNr [1:*] is for Car [0:1];
(09) PhoneNr [0:1] has Extension [1:*];
```

Figure 6.5: Expert-Created Data-Extraction Ontology for the Automobile Sales Application — Schemas

```
(01) Automobile [-> object];
(02) Automobile [0:1] has Mileage [1:1];
(03) Automobile [0:1] IsA.ARTIFACT.CostOfProduction Price [1:1];
(04) Automobile [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Furniture [1:1];
(05) ArtifactPart [0:1] IsA.ARTIFACT.CostOfProduction Price [1:1];
(06) ArtifactPart [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Furniture [0:1];
(07) AudioMediaArtifact [0:1] IsA.ARTIFACT.CostOfProduction Price [1:1];
(08) AudioMediaArtifact [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Furniture [0:1];
(09) CommunicationDevice [0:1] IsA.ARTIFACT.CostOfProduction Price [1:1];
(10) CommunicationDevice [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Furniture [0:1];
(11) ControlEvent [0:1] IsA.SOCIALEVENT.Location.PLACE.Subclasses Furniture [0:1];
(12) Price [1:1] IsA.SCALARATTRIBUTE.MeasuredIn.MEASURINGUNIT.Subclasses Year [0:*];
(13) TravelEvent [0:1] IsA.CHANGELOCATION.Destination.PHYSICALOBJECT.Subclasses ArtifactPart [0:1];
(14) TravelEvent [0:1] IsA.CHANGELOCATION.Destination.PHYSICALOBJECT.Subclasses Furniture [0:1];
(15) TravelEvent [0:1] IsA.CHANGELOCATION.Destination.PHYSICALOBJECT.Subclasses
     CommunicationDevice [0:1];
(16) TravelEvent [0:1] IsA.CHANGELOCATION.Destination.PHYSICALOBJECT.Subclasses
     AudioMediaArtifact [0:1];
(17) TravelEvent [0:1] IsA.CHANGELOCATION.Destination.PHYSICALOBJECT.Subclasses Automobile [0:1];
(18) TravelEvent [0:1] IsA.CHANGELOCATION.Destination.PHYSICALOBJECT.Subclasses Truck [0:1];
(19) Truck [0:1] IsA.ARTIFACT.CostOfProduction Price [1:1];
(20) Automobile [0:1] relatesTo PhoneNr [1:*] relatesTo ArtifactPart [1:*] relatesTo Mileage [1:*]
    relatesTo Truck [1:*] relatesTo AudioMediaArtifact [1:*] relatesTo CommunicationDevice [1:*]
    relatesTo ControlEvent [1:*] relatesTo TravelEvent [1:*];
```

Figure 6.6: Automatically Generated Data-Extraction Ontology for the Automobile Sales Application — Schemas

Comparing these two figures, we see that the primary object set `Automobile` in Figure 6.6 is the same as the primary object set `Car` in Figure 6.5. As a baseline, we first examine the POG and $PROG$ results for fully automatic ontology generation. Table 6.1 shows the POG and $PROG$ values for this generation process. Observe that for both POG

Comparing Items	POG	$PROG$
Object Sets	0.417	0.556
Relationship Sets	0.105	0.250
Participation Constraints	0.044	0.125
Overall	0.189	0.310

Table 6.1: POG and $PROG$ for Automobile Ontology-Generation

and $PROG$, the values for object sets are greater than those for relationship sets, and the values for relationship sets are greater than those for participation constraints. This is reasonable because miss-generated object sets lead to more miss-generated relationship sets that contain these object sets. Similarly, miss-generated relationship sets lead to more miss-generated participation constraints that relate to these relationship sets. Thus, in order to improve the overall performance, the most important factor is to improve the performance of object-set generation.

We, therefore, first study the experimental results of object sets. There are a total of 12 selected concepts in Figure 6.6. Five out of these twelve object sets, which are `Automobile`, `Mileage`, `Price`, `Year`, `PhoneNr`, match object sets in Figure 6.5. The POG value is thus 5 over 12, which is 0.417, as Table 6.1 shows. Figure 6.5 shows a total of 9 expert-specified object sets. Therefore, the $PROG$ value is 5 over 9, which is 0.556.

As mentioned before, we aim to achieve higher recall results with only acceptable precision values because it is much easier to let ontology builders reject and delete something wrong in a generated ontology than to let them write something totally new into an ontology. Thus, we are more interested in the four missing object sets, `Feature`, `Make`, `Model`, `Extension`, that the expert-created ontology includes but the automatically generated data-extraction ontology does not include, than in the incorrectly generated object sets, like `TravelEvent` in Figure 6.6. We can categorize these four object sets into two types: (1) object sets missing in the knowledge base, and (2) object sets in the knowledge base for which there are no data instances in the training records.

If the knowledge base does not contain any information about a concept, there is no way the system can generate an object set with respect to the concept. For example, the μK Ontology does not include information about car features, automobile makes and models. Thus, the object sets, `Feature`, `Make`, and `Model`, do not appear in Figure 6.6. This type of problem is unavoidable unless someone expands the knowledge base to include the missing knowledge. Further, as stated earlier, this is the reason we should apply $EPROG$ — to show the real recall of the ontology-generation process.

Table 6.2 shows the POG and $EPROG$ results of this automobile sales domain. Note that the POG values are the same as in Table 6.1. Since the μK Ontology does not

Comparing Items	POG	$EPROG$
Object Sets	0.417	0.833
Relationship Sets	0.105	0.400
Participation Constraints	0.044	0.200
Overall	0.189	0.478

Table 6.2: POG and $EPROG$ for Automobile Ontology-Generation

include the three object sets: `Feature`, `Make`, and `Model`, we do not count them when computing $EPROG$ values. Thus, the denominator for computing the $EPROG$ value of object sets is now 6 instead of the 9 object sets in Figure 6.5. Hence, the $EPROG$ result for object sets is 5 over 6, which is 0.833, instead of 0.556 as Table 6.1 shows. We can see that the $EPROG$ measures really represent the capability of our ontology-generation program.

There is still one missing object set, which is `Extension`, after we apply our $EPROG$ measurement. This is because of the second type of missing object sets as mentioned earlier — our training records do not have any instances of an extension of a phone. One way to avoid this type of error is to add more training records. For example, we used 10 randomly selected training records to generate the automobile data-extraction ontology in Figure 6.6. If we had increased this number to 20, there might have been a record containing a phone number that has an extension. Another possible resolution is to use human-supervised training sets instead of randomly selected sets. The ontology builder should know which records are better and more representative. We should point out, however, that if an object set does not appear in any of 10 randomly selected training records, it may not be an important object set for the application domain. This implies that if the object set is missing, there may not be severe problem even if the ontology builder does not fix this error in the final data-extraction ontology. Indeed, `Extension` is such an object set in this example. Extensions appear rarely in car ads.

There are two conditions that a desirable generated relationship set must fulfill. First, it must include only correctly generated object sets; and second, the relationship sets must be meaningful for the target domain. In Figure 6.6, only 2 out of a total of 19 generated relationship sets satisfy these two conditions. The POG result for relationship sets then is 2 over 19, which is 0.105. Since there are eight total relationship sets in the expert-created ontology, the $PROG$ value for relationship sets is 2 over 8, which is 0.25. At the same time, the system cannot generate three out of a total of eight relationship sets, which are Lines 2, 3 and 5 in Figure 6.5, because at least one of the related object sets is not inside the knowledge base. Thus, the $EPROG$ value is 2 over 5, which is 0.4. Another relationship set, Line 9 in Figure 6.5, however, is also impossible to generated by the system because although the knowledge base does have `Extension`, the object set did not appear in the list of selected object sets. Therefore, it is not the error of the relationship-retrieval process for missing such a relationship set. Because of these dependencies, we define a further measure, $DEPROG$ (Dependent $EPROG$), just for evaluating the recall of relationship sets and participation constraints, where the later results depend on earlier results. The $DEPROG$ value of relationship sets is thus 2 over 4, which is 0.5. Table 6.3

shows the POG and $DEPROG$ results for the automobile sales domain. Note that there is no dependency information for object sets, its $DEPROG$ value is the same as its $EPROG$ value.

Comparing Items	POG	$DEPROG$
Object Sets	0.417	0.833
Relationship Sets	0.105	0.500
Participation Constraints	0.044	0.500
Overall	0.189	0.611

Table 6.3: POG and $DEPROG$ for Automobile Ontology-Generation

Most of the miss-generated relationship sets arise because they contain at least one incorrectly selected object set. However, Line 12 in Figure 6.6 shows an example of a relationship that is still incorrect even though both the object sets, which are `Price` and `Year`, are object sets of interest. This relationship represents the variation of prices versus different years. Although this may be an important relationship in some other domain, people usually do not think about automobile sales in this way. Fortunately, however, such an "incorrect" relationship set at least gives us an object set of interest — `Year`. The ontology builder can figure out a way to correctly link the object set to other correctly selected object sets, i.e., to link `Automobile` with `Year`.

Another special type of error comes from n-ary relationship sets. We generate large n-ary relationship because we do not know how to properly decompose them and because it is easier for ontology builders to decompose large n-ary relationship sets than to create something totally new. Line 20 in Figure 6.6 shows an example of an n-ary relationship set consisting of nine object sets. We expect a relation between the concepts `Automobile` and `PhoneNr`. However, there is no direct connection between these two concepts in our knowledge base. In addition, the concept `PhoneNr` does not directly connect to any one of the selected concept nodes. The system thus creates an n-ary relationship set to link the concept `PhoneNr` with other selected concepts. As discussed earlier in Section 5.4, the created n-ary relationship set includes many of the selected concepts as Figure 6.6

shows. Because it contains some miss-selected concept, like `TravelEvent`, this n-ary relationship set as a whole is an error. It is, however, still useful because end-users can easily retrieve the expected binary relationship set between the concepts `Automobile` and `PhoneNr` by deleting all other concepts of the n-ary relationship set.

A question that arises here is whether we can "straightforwardly" solve this problem by allowing the system to generate more binary relationship sets even when the distance between the two concepts is greater than one. This solution is actually not good because it leads to two bad side effects: (1) We may generate many more incorrect binary relationship sets, making it difficult to look through all of the generated relationship sets and reject the miss-generated ones. (2) It is easier to decompose a large n-ary relationship set than to write a new binary or n-ary relationship set.

Another question that arises with these large n-ary relationship sets is whether we can solve this type of error by improving our knowledge base to be exactly what we expect. This expectation could be reasonable or unreasonable. We can design knowledge sources to fulfill in any types of requirements. The problem is, however, whether it is practically possible or how hard it is, if it is possible, to find existing knowledge sources for a user's expectations. Phones, for example, are not generally associated with cars; indeed, the phone number expected in this example is not the phone number for the car, but rather the phone number of a person or dealer wishing to sell the car.

There appears to be no easy way to perfectly solve the problems with relationship sets. The one we have currently chosen is to let the system generate the binary relationship sets that are directly connected and large n-ary relationship sets and then allow extraction-ontology builders to discard unwanted binary relationship sets and decompose these large n-ary relationship sets.

Finally, we discuss mis-generated participation constraints. Note that only the participation constraints in a correctly generated relationship set can be correct. To compute the $EPROG$ result, we first recall that the system can only generate five out of the nine binary relationship sets. Therefore, only ten participation constraints within those five relationship sets can appear in the generated data-extraction ontology. Thus, the denominator of the $EPROG$ measure is 10. Both the participation constraint for `Mileage` and `Price` in their relationship sets with `Automobile` should be [1:*] instead of [1:1] and the system missed relationship sets with `Year`, `PhoneNr`, and `Extension`. Thus, the system only generated 2 correct participation constraints. The $EPROG$ result for participation constraint then is 2 over 10, which is 0.2. To be more realistic, we again evaluate a $DEPROG$ value of participation constraints based on correctly generated relationship sets. Since there are only two correct relationship sets, the $DEPROG$ value is 2 over 4, which is 0.5.

For all the miss-generated cases, participation-constraint errors occur because of the lack of training examples. For example, Line 2 in Figure 6.6 shows a miss-generated maximum participation constraint for object set `Mileage` in the relationship set with `Automobile`. Because all the values of mileage in the training records are different, the system assumes that the values in this object set uniquely identify the objects in the object set `Automobile` and thus sets the participation constraints as [1:1]. By common sense, we know, however, that [1:1] is not correct because different vehicles can have the same mileage. Theoretically, increasing the number of training records can solve this problem. Practically, however, this is not that easy because it is hard to find two used cars advertised for sale that actually have the same mileage value. If we gather the training records by unsupervised random selection, we would likely need a very large number of records to fix this error.

This observation leads naturally to the question: what is the best size for the number of training records? Increasing the size of the training space may lead to problems like increasing generation time, and more miss-generated object and relationship sets. The solution depends on how many total object sets are in the application and how often the instances of each object set appear at the training records, which is unpredictable across application domains. In addition, we note that the size of training space needed for supervised selection can obviously always be smaller than the size needed for unsupervised selection. An increase in the number of training records may generate participation constraints better. On the other hand, even if the generated participation constraints are incorrect, they are comparatively easy to fix because domain experts usually know their application well.

Based on our discussion and analysis as a guide, extraction-ontology builders can modify the automatically generated data-extraction ontology to create a more applicable

```
(01) Automobile [-> object];
(02) Automobile [0:1] has Mileage [1:1];
(03) Automobile [0:1] IsA.ARTIFACT.CostOfProduction Price [1:1];
(12) Price [1:1] IsA.SCALARATTRIBUTE.MeasuredIn.MEASURINGUNIT.Subclasses Year [0:*];
(20) Automobile [0:1] relatesTo PhoneNr [1:*] relatesTo ArtifactPart [1:*] relatesTo Mileage [1:*]
     relatesTo Truck [1:*] relatesTo AudioMediaArtifact [1:*] relatesTo CommunicationDevice [1:*]
     relatesTo ControlEvent [1:*] relatesTo TravelEvent [1:*];
```

Figure 6.7: Deleted Automatically Generated Data-Extraction Ontology for the Automobile Sales Application — Schemas

```
(01) Automobile [-> object];
(02) Automobile [0:1] has Mileage [1:1];
(03) Automobile [0:1] IsA.ARTIFACT.CostOfProduction Price [1:1];
(12) Automobile [1:1] IsA.SCALARATTRIBUTE.MeasuredIn.MEASURINGUNIT.Subclasses Year [0:*];
(20) Automobile [0:1] relatesTo PhoneNr [1:*];
```

Figure 6.8: Updated Automatically Generated Data-Extraction Ontology for the Automobile Sales Application — Schemas

extraction ontology for their specified domain. First, extraction-ontology builders need to remove all the "bad" relationship sets. Note that an incorrect relationship set may not necessarily be bad. Line 12 in Figure 6.6 is an example. We should not delete this one because it is the only relationship set that contains `Year`, which is a correct object set. Figure 6.7 shows the schema part of an automatically generated ontology after deleting all the "bad" relationship sets. At the same time, they should also remove all the concept recognizers of those object sets no longer in the data-extraction ontology.

After that, the ontology builder should adjust the remaining incorrect relationship sets by either substituting an incorrectly linked object set with a correct one (e.g. replace `Price` with `Automobile` in Line 12) or reducing a long n-ary relationship set to one or more small binary or n-ary relationship sets where each of them represent a relationship of interest(e.g. reduce Line 20 to a binary set containing only `Automobile` and `PhoneNr`). Figures 6.8 shows the schema part of the reduced, automatically generated ontology for our automobile sales application domain.

The next step is to modify a data-extraction ontology in three ways: (1) modify the names of object sets and relationship sets so that they can be more meaningful; (2) adjust the order of object sets in a relationship set so that people can more easily understand

```
Car [-> object];
Car [0:1] has Year [1:*];
Car [0:1] has Mileage [1:*];
Car [0:1] has Price [1:*];
PhoneNr [1:*] is for Car [0:1];
```

Figure 6.9: Modified Automatically Generated Data-Extraction Ontology for the Automobile Sales Application — Schemas

```
Car [-> object];
Car [0:1] has Year [1:*];
Car [0:1] has Mileage [1:*];
Car [0:1] has Price [1:*];
PhoneNr [1:*] is for Car [0:1];
PhoneNr [0:1] has Extension [1:*];
Car [0:*] has Feature [1:*];
Car [0:1] has Make [1:*];
Car [0:1] has Model [1:*];
```

Figure 6.10: Final Version of Automatically Generated Data-Extraction Ontology for the Automobile Sales Application — Schemas

them; and (3) fix incorrect participation constraints. Figure 6.9 shows the modified data-extraction ontology. Note that although some modifications are not not necessary, like the name change of `Automobile` to `Car`, or to switch the sequence of object sets, like we did for `PhoneNr` and `Car`, we make these changes on purpose, both to illustrate the possibility and to make the generated ontology correspond directly to the expert-created ontology.

As a last step, ontology builders can add anything they want to the generated ontology. Figure 6.10 shows that the ontology builder can add an extension object set to let the system extract the extension of phone numbers if applicable. Moreover, since ontology builders know about features, makes, and models of cars (concepts our μK Ontology knows nothing about), they can provide the object and relationship sets and the recognizers for these object sets and add them into the final version data-extraction ontology. Comparing Figure 6.10 with Figure 6.5, we can see that they are the same. The only difference between them is that the ontology builder only needs to write four out of nine lines by hand. Thus, more than half of the human's load has been shifted to the machine. Note that the percentage generated can be much higher if the knowledge base contains more concepts of interest.

6.3 More Experimental Results based on the μK Ontology

We have evaluated some other application domains besides automobile sales. Among them, some domains, such as nation information and apartment rentals, achieved good recall results. Some others, like cell-phone plans and obituaries, however, achieved poor results both on precision and recall. The reason for the difference is the knowledge base. If the integrated knowledge base defines the information well, including the concepts, the concept recognizers, and the defined relationships among all the concepts of interest, our ontology-generation process can generate a good initial ontology fulfilling most of the extraction-ontology builder's requirements. For example, because the μK Ontology defines general geographical and economic information well, the overall $PROG$ value of the generated data-extraction ontology for the nation domain is above 0.9.

Table 6.4 shows the POG, $PROG$, $EPROG$ and $DEPROG$ results of the other two application domains that are reasonably well-defined in our integrate knowledge base. Our system can generate a good initial data-extraction ontology when the knowledge base matches the specific requirements of users. Users may then obtain a satisfactory final data-extraction ontology by rejecting inapplicable concepts and making some minor adjustments. Appendix B shows the schema portion of the output data-extraction ontologies of these two additional applications. In addition, Appendix B also shows an updated extraction ontology obtained by rejecting extraneous relationship sets and reducing large n-ary

Application Domain	Comparing Items	POG	$PROG$	$EPROG$	$DEPROG$
Apt-Rental	Object Sets	0.297	0.643	0.900	0.900
	Relationship Sets	0.039	0.385	0.556	0.625
	Participation Constraints	0.028	0.308	0.444	0.500
	Overall	0.121	0.445	0.633	0.675
Nation	Object Sets	0.476	1.00	1.00	1.00
	Relationship Sets	0.133	1.00	1.00	1.00
	Participation Constraints	0.099	0.722	0.722	0.722
	Overall	0.236	0.907	0.907	0.907

Table 6.4: POG, $PROG$ and $EPROG$ Values

relationship sets. It also shows an expert-created version of data-extraction ontology for each of the two domains.

We can find some more interesting problems in these two applications. There is a relationship set in the expert-created ontology for the apartment-rental domain, which is `Apt has Location`. Although the knowledge base does have a corresponding concept, which is `Address`, our ontology-generation system matches the information in the training records with three component concepts of `Address`, which are `USAddressCity`, `USAddressState`, and `USAddressZipCode`. Since these three object sets can aggregate to be `Address` or `Location` as in the expert-created ontology, we decide that the object set `Location` should be treated as having been correctly generated. A similar but more interesting example of this problem is found in the domain of nation information in which we have a relationship set `Nation has AgriculturalProducts`. Again, although there is a concept named `AgriculturalProduct` in the μK Ontology, the system matches some of its subclass concepts, like `AnimalProduct`, `CropPlant`, `FruitFoodstuff`, etc. For this example, however, the system matches only a subset but not all of the subclass concepts of `AgriculturalProduct`. A human, however, should have enough knowledge to judge that `AgriculturalProduct` is the real object set of interest. Thus, we also treat `AgriculturalProduct` is a correctly generated object set.

Another type of problem arises because of some concepts, which are close in meaning. For example, in the apartment-rental domain, we have an object set named `Furnished`, which refers to whether the apartment is furnished or not. There is no such concept in the knowledge base which, however, has a related concept `FurniturePart` that indicates the presence of furniture. If an apartment has furniture, it is furnished; otherwise it is not. So, we face a situation in which the information exists at an abstract conceptual level, but not exactly in the form desired. Since we have what we want at an abstract conceptual level, we treat this as a correct selection.

When the integrated knowledge base does not contain the application domain, usually the system recognizes several instances where the recognized concepts compose a cluster that does not represent the actual domain of the training records. For example, when training some obituary records in our system, we obtain a cluster of concepts about sport events because the μK Ontology defines a fairly large sport event domain which has some overlap with information specific for obituaries. The system, for example, automatically decides that the names of athletes are person names (deceased-person names and relative names) and that the dates for some sport event match dates of birth, death, and funerals. It thus does not help people to write a data-extraction ontology for the obituary domain. These experiments indicate that the success of this ontology-generation system depends fundamentally on the quality of the knowledge sources for a desired application domain. This is consistent with our simulation of the human brains — a person who knows nothing about cell phones is not likely to create a good data-extraction ontology for a cell-phone domain.

6.4 Experiments with μK plus SALT

We have seen that our system can generate a good data-extraction ontology. The quality of the generated ontology, however, heavily depends on how well the knowledge base includes the domain of interest. Because no existing knowledge sources fit perfect on all kinds of data-extraction tasks, we expect our knowledge base be extensible so that the ontology builders could use our system to generate data-extraction ontology even if the default knowledge base does not contain their domain of interest. In this section, we describe an experiment about extending the default μK Ontology with another knowledge source.

In our experiment, we added a new knowledge source – SALT [38] – that contains both new concepts and new relationship sets. The SALT project contains a data model that addresses the problem of interchange among diverse collections of lexicon/termbase[2] data, including their ontological substructure.[3] In our experiments, we use the Eurodicautom terminology bank[4] as our termbase. The Eurodicautom terminology bank consists of over a million concept entries covering a wide range of topics. Each entry is multilingual in character, containing equivalents in any of several languages. Figure 6.11 shows part of the

[2] terminology bases
[3] In progress as ISO 16642 (forthcoming).
[4] See europa.eu.int.

entry for a sample term ("charcoal-kiln"); it shows translation equivalents for the term in various languages (Danish on Lines 3 and 4, Italian on Lines 9 and 10, Spanish on Lines 12 and 13, and Swedish on Lines 15 and 16). Crucially for this project, the line that begins

```
(01) {\%\%CM AG8 IC6}
(02) {\%\%DA}
(03) {\%\%VE trækulsmile}
(04) {\%\%RF A.Klougart}
(05) {\%\%EN}
(06) {\%\%VE charcoal-kiln[1];earth kiln[2]}
(07) {\%\%RF CILF,Dict.Agriculture,ACCT,1977[VE1,VE2]}
(08) {\%\%IT}
(09) {\%\%VE catasta di carbonella sottoposta a combustione incompleta per ottenere il carbone di
     legna}
(10) {\%\%RF Enciclopedia Scientifica-Tecnica Garzanti,Ed.Garzanti,1982}
(11) {\%\%ES}
(12) {\%\%VE horno carbonero}
(13) {\%\%RF CILF,Dict.Agriculture,ACCT,1977}
(14) {\%\%SV}
(15) {\%\%VE kolmila}
(16) {\%\%RF Györki,I.(m.fl.),Bonniers svenska ordbok.-1994}
(17) ...
```

Figure 6.11: Partial Entry from the Eurodicautom Termbase in Native Format

with %%CM lists two Lenoch codes [33] that associate pertinent subject areas with the term: AG8 (representing the subclass FORESTRY) and IC6 (representing EXPLOSIVES AND BLASTING MATERIALS).

The Eurodicautom is not in the XML format. In order to make use of this knowledge source, we first convert termbase entries via the SALT-developed TBX termbase exchange framework. Figure 6.12 shows a sample of the TBX termbase transformed from Figure 6.11. Note that each <langSet> pair defines an interpretation of the term specified in the language indicating by the attribute of <langSet>. Thus, the first <langSet> indicates a Danish interpretation and the second <langSet> represents an English interpretation. The <termGrp> for each <langSet> indicates the terms that can represent the same technical meaning in the specified language. Since we are only interested in data extraction

```
<termEntry id="eid-EDIC-BTB-DAG78-337">
        <admin type="originatingInstitution">BTB</admin>
        <admin type="projectSubset">DAG77</admin>
        <descrip type="reliabilityCode">6</descrip>
        <descrip target="cla-lenoch" type="classificationCode">AG8</descrip>
        <descrip target="cla-lenoch" type="classificationCode">IC6</descrip>
        <langSet lang="da">
          <ntig>
            <termGrp>
              <term>trækulsmile</term>
              <termNote type="termType">fullForm</termNote>
            </termGrp>
            …
          </ntig>
        </langSet>
        <langSet lang="en">
          <ntig>
            <termGrp>
              <term>charcoal-kiln</term>
              <termNote type="termType">fullForm</termNote>
            </termGrp>
            …
          </ntig>
          <ntig>
            <termGrp>
              <term>earth kiln</term>
              <termNote type="termType">fullForm</termNote>
            </termGrp>
            …
          </ntig>
        </langSet>
        …  …
</termEntry>
```

Figure 6.12: Sample Eurodicautom-TBX Encoding

in English, we have removed all <langSet>'s that are not English from the Eurodicautom-TBX encoding for further processing.

The TBX termbase itself cannot help us because it just defines some terms without any hierarchy or relationship information. As we have seen in our knowledge base, a concept can mean something only after it has a meaningful relationship with other concepts. Lenoch codes help us define the hierarchy of Eurodicautom terms. Figure 6.13 shows some examples of Lenoch codes. The numbers appended to codes define a hierarchical status.

ER	Earth Resources - Energy
ER1	general aspects of the subject field
ER4	energy sources
ER41	fossil energy
ER42	nuclear energy
ER43	alternative sources of energy (nt: primary sources of energy)
ER431	durable sources of energy
ER4311	wind energy
ER4312	tidal energy
ER4313	solar energy
ER7	raw materials
ER71	coal (sn: energy source; us: for winning and preparation, see: MI)
ER711	coal varieties

Figure 6.13: Sample Lenoch Codes

For example, `ER4`, which represents `energy sources`, is a subclass of `ER`, which is a general class of energy. Similarly, `ER43`, which represents `alternative sources of energy`, is a subclass of `ER4`. `ER431`, which represents `durable sources of energy`, is one type of alternative source of energy. Thus, `ER431` is a subclass of `ER43`. The rest of the definitions follows the same pattern.

After we have collected the terms of a TBX termbase and the hierarchy information from Lenoch codes, We convert both inputs and transfer them to the μK XML format as Figure 6.14 shows. Because Lenoch codes define a term hierarchy and Eurodicautom

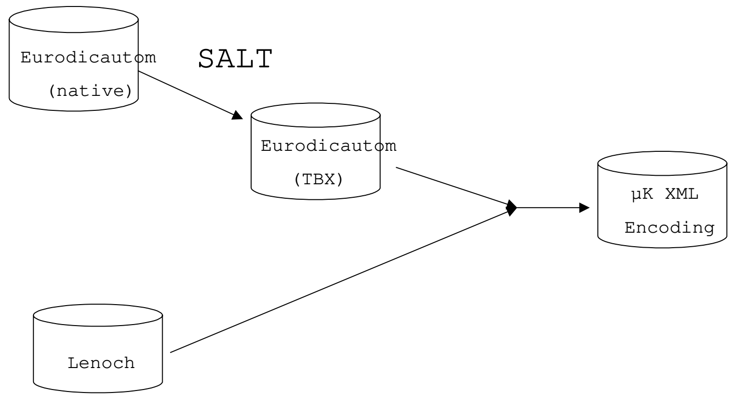

Figure 6.14: SALT Conversion Process

terminology bank defines the category information of technical terms, it is possible for us to take them together and arrange them in the μK XML format. Figure 6.15 shows samples of the μK XML encoding.

After we have transferred all the new knowledge sources into the μK XML format, we need to redo the knowledge-base preprocessing to integrate these new sources into the existing knowledge base. Typically, by using WordNet, the system can automatically match some concepts in new sources with existing concepts in the current knowledge base and suggest the mappings. A human expert, however, needs to examine especially the root-level concepts in the new knowledge source to make sure they are properly linked inside the knowledge base. Note that the new added knowledge source from SALT includes only hierarchical structure; it does not have relationships and thus does not add new relationship sets into the μK Ontology. Since some concepts in the new knowledge source match with existing concepts in the μK Ontology, however, we obtain all the relationship sets μK already has among matched concepts and among SALT concepts and μK concepts.

```
<RECORD>
  <CONCEPT>energy sources</CONCEPT>
  <SLOT>SUBCLASSES</SLOT>
  <FACET>VALUE</FACET>
  <FILLER>alternative sources of energy</FILLER>
  <UID><0></UID>
</RECORD>
<RECORD>
  <CONCEPT>alternative sources of energy</CONCEPT>
  <SLOT>SUBCLASSES</SLOT>
  <FACET>VALUE</FACET>
  <FILLER>durable sources of energy</FILLER>
  <UID><0></UID>
</RECORD>
<RECORD>
  <CONCEPT> durable sources of energy</CONCEPT>
  <SLOT>SUBCLASSES</SLOT>
  <FACET>VALUE</FACET>
  <FILLER>wind energy</FILLER>
  <UID><0></UID>
</RECORD>
….
```

Figure 6.15: Samples of μK-Type XML Encoding

We ran the system on various U.S. Department of Energy (DOE) abstracts[5] to test if we had successfully updated our knowledge base by properly integrating the new knowledge sources into the existing knowledge base. Figure 6.16 shows a sample abstract. Figure 6.17 shows a very small portion of the output ontology generated by the system using the updated knowledge base.

> The trend in supply and demand of fuel and the fuels for electric power generation, iron manufacturing and transportation were reviewed from the literature published in Japan and abroad in 1986. FY 1986 was a turning point in the supply and demand of energy and also a serious year for them because the world crude oil price dropped drastically and the exchange rate of yen rose rapidly since the end of 1985 in Japan as well. The fuel consumption for steam power generation in FY 1986 shows the negative growth for two successive years as much as 98.1%, or 65,730,000 kl in heavy oil equivalent, to that in the previous year. The total energy consumption in the iron and steel industry in 1986 was 586 trillion kcal (626 trillion kcal in the previous year). The total sales amount of fuel in 1986 was 184,040,000 kl showing a 1.5% increase from that in the previous year. The concept Best Mix was proposed as the ideal way in the energy industry. (21 figs, 2 tabs, 29 refs)

Figure 6.16: Sample DOE Abstracts Text

We should make a few remarks about the generated ontology. Among several dozens of relationship sets, some are spurious and several are correct. For example, a relationship is appropriately posited between the concept `CrudeOil` and the concept `Nation`. The DOE-abstract reader may be interested in which country has oil products. However, we did not try to build an expert-created comparative data-extraction ontology for DOE abstracts and compute the precision and recall values as we have done for previous

[5]The corpus is part of the ACL/DCI collection; see www.ldc.upenn.edu/Catalog/LDC93T1.html.

```
-- DOE Abstract Ontology

Alloy [0:*] MadeOf.SOLIDELEMENT.Subclasses MetallicSolidElement [0:*];
Alloy [0:*] IsA.METAL.StateOfMatter.SOLID.Subclasses CrudeOil [0:*];
Alloy [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Produce [0:*];
AmountAttribute [0:*] IsA.SCALARATTRIBUTE.MeasuredIn.MEASURINGUNIT
Consumption [0:*] IsA.FINANCIALEVENT.Agent Human [0:*];
ControlEvent [0:*] IsA.SOCIALEVENT.Agent Human [0:*];
ControlEvent [0:*] IsA.SOCIALEVENT.Location.PLACE.Subclasses Nation [0:*];
CountryName [0:*] NameOf Nation [0:*];
CountryName [0:*] IsA.REPRESENTATIONALOBJECT.OwnedBy Human [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.OwnedBy Human [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.GROW.Subclasses GrowAnimate [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Increase [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Combine [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Display [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Produce [0:*];
Custom [0:*] IsA.ABSTRACTOBJECT.ThemeOf.MENTALEVENT.Subclasses AddUp [0:*];
Display [0:*] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.Subclasses Gas [0:*];
Display [0:*] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.OwnedBy Human [0:*];
ForProfitCorporation [0:*] OwnedBy Human [0:*];
ForProfitCorporation [0:*] IsA.CORPORATION.HasNationality Nation [0:*];
Gas [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];
Gas [0:*] IsA.PHYSICALOBJECT.ThemeOf.GROW.Subclasses GrowAnimate [0:*];
LinseedOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Increase [0:*];
```

Figure 6.17: Sample Relations in the Generated Ontology for DOE Abstracts

studies. There are two reasons. First, unlike our previous studies, DOE abstracts cover a wide range of topics, which contradicts with our assumption of a narrow domain of interest. Second, as said at the beginning of this section, the goal of this additional experiment was to test whether the system can properly integrate new sources and generate meaningful relationships between the new added concepts and the original concepts. In our experiments, we see that the system can now extract some new relationship sets, which do not exist in only one of the knowledge sources. The relationship between `CrudeOil`, which is a new concept, and `Nation`, which is an old concept in the original μK Ontology, is an example.

As a final remark, we mention that the performance of generating a data-extraction ontology with added domains depends on how well the knowledge base and the new sources integrate. WordNet cannot solve everything. We may want to study some ways to automate this integration process and improve its performance to make the system more useful [52], though this is out of the scope of this thesis work.

Chapter 7

CONCLUSIONS AND FUTURE WORK

This chapter presents the conclusions and possible future explorations of this thesis work.

7.1 Conclusions

Previous studies show that the BYU-Ontos approach works successfully in many applications. However, users must create data-extraction ontologies manually, which tends to be the bottleneck of the approach. This is our first attempt to automate the ontology-generation process. We base our idea on the belief that machines can simulate a human's behavior of building data-extraction ontologies. Through a similar process, machines can match the knowledge implicitly represented in target documents with the knowledge explicitly stored in the knowledge base and retrieve the relations among the matched knowledge to create a data-extraction ontology for the application domain of interest.

This work shows how to create data-extraction ontologies by exploiting existing knowledge. We presented four requirements for a primary knowledge source: (1) available, (2) general in coverage, (3) rich in relationship, and (4) formatted in XML. Based on these four requirements, we selected the μK ontology as our primary knowledge source in our studies and adopted the μK XML format as the format of our knowledge base. We also described the data-frame library for handling concept recognizers and showed how we used WordNet as our synonym dictionary. Through either mapping the concepts in the μK Ontology with the corresponding lexicons or recognizers or adding into the μK Ontology some concept nodes that only appear in the data-frame library, we integrate all of these knowledge sources together as a unified knowledge base.

As the input of our ontology-generation program, we defined a standard way to specify the application domain of interest. We parse training documents, clean them by getting rid of the useless advertisements and frames, and divide them into multiple training records unless there is only one record in a document. Each training record represents some part of the domain of interest and the total domain of interest is the union of all these subparts.

After the knowledge base and selected training documents are ready, the system starts the generation process by locating the concepts that appear both in the knowledge base and in the training documents. Then the system retrieves the best cluster of concepts out of the knowledge base and discovers the participation constraints for the retrieved relationship sets from the training records to make the schema of the generated data-extraction ontology.

Based on an expert-created data-extraction ontology for the same application, the overall $DEPROG$ (Dependent Effective Pseudo-Recall of Ontology Generation) values average greater than 70% when the application domain is embedded in the knowledge base. When the primary knowledge source defines the application domain of interest well, as the μK Ontology does for the nation information domain, the $DEPROG$ values are over 90%.

Besides the μK ontology, we also tried to add a SALT termbase into our knowledge base to enlarge the coverage. The results on the DOE abstracts domain showed that we can generate data-extraction ontologies for more than just the μK ontology and thus we can broaden the domain of possible applications by introducing new knowledge sources.

This project has integrated many different issues. We applied data-extraction and string-retrieval techniques to select appropriate concepts. We also studied many different conflict resolution heuristics to solve conflictions among the selected concepts. We used graph-search techniques to create a subgraph within a large knowledge graph to represent the domain of interest and to retrieve relationship sets within the selected subgraph. We also tried to automatically generate n-ary relationship sets based on multiple binary relationships. Although the results did not yield the desired relationship sets, they did, however, yield useful superset relationship sets. We also presented heuristics to retrieve participation constraints from training records.

7.2 Future Work

Although we have accomplished our goal of semi-automatically generating data-extraction ontologies, much remains to be done.

As discussed in Chapter 6, although we use the μK ontology in this research work, the μK ontology does not work as well as we expected. For many application domains, the relationship sets defined in the μK ontology do not describe the schemas in real Web documents well. One reason why many of current ontologies do not work well for data-extraction task is because they are designed for linguistic purposes by NLP researchers. In particular, the relationships in these ontologies are designed for the purposes of natural language parsing instead of data extraction. In order to make this ontology-generation approach more applicable, we or someone should perhaps build a general-purpose knowledge source for data-extraction usage.

This research focuses on automating schema-generation of data-extraction ontologies. In addition to the existing concept recognizers, or data frames in the knowledge base, the system can output an executable data-extraction ontology that fulfills the ontology builder's interest. The data frames, however, are not trivial. For new applications, we cannot guarantee that all the concepts of interest have properly defined data frames. Thus, there are three problems for data frames: (1) whether we can correctly locate a concept using an existing data frame; (2) whether we can adjust an existing data frame to make it fit a specified situation; and (3) whether we can build a new data frame with a collection of information of interest. In this research work, we have studied and experimented with the first problem, which is still not perfectly solved as we have seen. These three problems are some other interesting future work worth exploring.

In conclusion, since this work is the first attempt at solving the problem of automating ontology generation for data extraction, we cannot expect all the problems to have been solved. Because more and more people are interested in gathering information from Web, and since the data-extraction-ontology approach is one good way to achieve this goal, more research in this field is warranted.

Bibliography

[1] B. Adelberg. NoDoSE—a tool for semi-automatically extracting structured and semistructured data from text documents. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 283–294, Seattle, Washington, June 1-4, 1998.

[2] G.O. Arocena and A.O. Mendelzon. WebOQL: Restructuring documents, databases and webs. In *Proceedings of the Fourteenth International Conference on Data Engineering (ICDE 1998)*, pages 24–33, Orlando, Florida, February 23-27, 1998. IEEE Computer Society Press.

[3] R. Bourret. Data transfer strategies: Transferring data between XML documents and relational databases. URL: http://www.rpbourret.com/xml/DataTransfer.htm, Checked July, 2001.

[4] R. Bourret. XML and databases. URL: http://www.rpbourret.com/xml/ XMLAndDatabases.htm#isxmladatabase, Checked February, 2002.

[5] R. Bourret. XML database products. URL: http://www.rpbourret.com/xml/ XMLDatabaseProds.htm#xmlanddatabases, Checked March, 2002.

[6] D. Buttler, L. Liu, and C. Pu. A fully automated object extraction system for the World Wide Web. In *Proceeding of the 21st International Conference on Distributed Computing (ICDCS-21)*, pages 361–370, Phoenix, Arizona, April 16-19, 2001. IEEE Computer Society Press.

[7] M.E. Califf and R.J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 487–493, Orlando, Florida, July 18-22, 1999.

[8] B. Chidlovskii, U. Borghoff, and P. Chevalier. Towards sophisticated wrapping of web-based information repositories. In *Proceedings of 5th International RIAO Conference on Computer-Assisted Information Searching on the Internet*, pages 123–135, Montreal, Canada, June 25-27, 1997.

[9] V. Crescenzi and G. Mecca. Grammars have exceptions. *Information Systems*, 23(8):539–565, December 1998.

[10] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In P.M.G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R.T. Snodgrass, editors, *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 109–118, Rome, Italy, September 11-14, 2001. Morgan Kaufmann 2001.

[11] Homepage for BYU Data Extraction Group. URL: http://www.deg.byu.edu.

[12] L. Eikvil. Information extraction from world wide web — a survey. Technical Report 945, Norweigan Computing Center, Oslo, Norway, July 1999.

[13] D.W. Embley. Programming with data frames for everyday data items. In *Proceedings of the 1980 National Computer Conference*, pages 301–305, Anaheim, California, May 19-22, 1980.

[14] D.W. Embley. *Object Database Development: Concepts and Principles*. Addison-Wesley, Reading, Massachusetts, 1998.

[15] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.

[16] D.W. Embley, D.M. Campbell, S.W. Liddle, and R.D. Smith. Ontology-based extraction and structuring of information from data-rich unstructured documents. In *Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM'98)*, pages 52–59, Washington D.C., November 3-7, 1998.

[17] D.W. Embley, Y.S. Jiang, and Y.-K. Ng. Record-boundary discovery in Web documents. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 467–478, Philadelphia, Pennsylvania, 31 May - 3 June, 1999.

[18] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object-oriented Systems Analysis: A Model-Driven Approach*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[19] D.W. Embley and L. Xu. Record location and reconfiguration in unstructured multiple-record Web documents. In *Proceedings of the Third International Workshop on the Web and Databases (WebDB2000)*, pages 123–128, Dallas, Texas, May 18-19, 2000.

[20] D. Freitag. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2/3):169–202, May 2000.

[21] J. Hammer, H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig, and V. Vassalos. Template-based wrappers in the TSIMMIS system. In *Proceedings of 1997 ACM SIGMOD International Conference on Management of Data*, pages 532–535, Tucson, Arizona, May 13-15, 1997.

[22] M. Helles. A survey and taxonomy of wrappers. Technical report, University of Copenhagen, 2000.

[23] K. Hewett. An integrated ontology development environment for data extraction. Master's thesis, Brigham Young University, Provo, Utah, April 2000.

[24] R. Himmeroder, G. Lausen, B. Ludascher, and C. Schlepphorst. On a declarative semantics for web queries. In *Proceeding of the 5th International Conference on Deductive and Object-Oriented Databases (DOOD'97)*, pages 386–398, Montreux, Switzerland, December 8-12, 1997.

[25] D.S. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, Boston, MA, 1997.

[26] C-N. Hsu and M-T. Dung. Generating finite-state transducers for semi-structured data extraction from the Web. *Information Systems*, 23(8):521–538, December 1998.

[27] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, July 1995.

[28] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. On semi-automated web taxonomy construction. In *In the Proceedings of the Fourth International Workshop on the Web and Databases (WebDB'2001)*, pages 91–96, Santa Barbara, California, May 24-25, 2001.

[29] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, April 2000.

[30] N. Kushmerick. Wrapper verification. *World Wide Web: Internet and Web Information Systems*, 3(2):79–94, 2000.

[31] A.H.F. Laender, B. Ribeiro-Neto, and A.S. Da Silva. Debye—data extraction by example. *Data and Knowledge Engineering*, 40(2):121–154, 2002.

[32] A.H.F. Laender, B.A. Ribeiro-Neto, A.S. da Silva, and J.S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31(2):84–93, June 2002.

[33] LENOCH Source. URL: http://www.admin.ch/ch/i/bk/termdat/d/termdat/class.htm.

[34] K. Lerman and S. Minton. Learning the common structure of data. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 609–614, Austin, Texas, July 30 - August 3, 2000. AAAI Press.

[35] S.W. Liddle, D.W. Embley, and S.N. Woodfield. Cardinality constraints in semantic data models. *Data & Knowledge Engineering*, 11(3):235–270, December 1993.

[36] S.W. Liddle, D.W. Embley, and S.N. Woodfield. Unifying Modeling and Programming Through an Active, Object-Oriented, Model-Equivalent Programming Language. In *Proceedings of the Fourteenth International Conference on Object-Oriented and Entity-Relationship Modeling (OOER'95), Lecture Notes in Computer Science,*

1021, pages 55–64, Gold Coast, Queensland, Australia, December 13-15, 1995. Springer Verlag.

[37] L. Liu, C. Pu, and W. Han. An XML-enabled wrapper construction system for Web information sources. In *Proceedings of the 16th International Conference on Data Engineering*, pages 611–621, San Diego, California, February 28–March 3, 2000.

[38] D. Lonsdale, Y. Ding, D.W. Embley, and A. Melby. Peppering knowledge sources with SALT: Boosting conceptual content for ontology generation. In *AAAI Workshop for Semantic Web Meets Language Resources, The Eighteenth National Conference on Artificial Intelligence*, pages 30–36, Edmonton, Alberta, Canada, July 28, 2002.

[39] A. Maedche and S. Staab. Mining ontologies from text. In *Proceedings of 12th International Conference of Knowledge Engineering and Knowledge Management (EKAW 2000)*, pages 189–202, Juan-les-Pins, France, October 2-6, 2000.

[40] K. Mahesh. Ontology development for machine translation: Ideology and methodology. Technical Report MCCS-96-292, Computer Research Laboratory, New Mexico State Univeristy, 1996.

[41] A.O. Mendelzon, G.A. Mihaila, and T. Milo. Querying the World Wide Web. *International Journal on Digital Libraries*, 1(1):54–67, April 1997.

[42] G.A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, November 1995.

[43] I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, March 2001.

[44] The Source for PERL. URL: http://www.perl.com/.

[45] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth national Conference on Artificial Intelligence (AAAI-99)*, pages 474–479, Orlando, Florida, July 18-22, 1999.

[46] A. Sahuguet and F. Azavant. Building intelligent web applications using lightweight wrappers. *Data and Knowledge Engineering*, 36(3):283–316, March 2001.

[47] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1–3):233–272, 1999.

[48] C.A. Thompson, M.E. Califf, and R.J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the Sixteenth International Machine Learning Conference (ICML'99)*, pages 406–414, Bled, Slovenia, June 27-30, 1999.

[49] Homepage for WordNet. URL: http://www.cogsci.princeton.edu/˜wn/.

[50] Extensible markup language (XML). URL: http://www.w3.org/XML/.

[51] Homepage for a listing of organizations producing industry-specific XML DTDs. URL: http://xml.org/xmlorg_registry/index.html.

[52] L. Xu and D.W. Embley. Discovering direct and indirect matches for schema elements. In *Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA 2003)*, pages 39–46, Kyoto, Japan, March 26-28 2003.

Appendix

Appendix A

The Longest Path

Woh, oh-oh-oh
Find the Longest Path
Woh oh-oh
Find the Longest Path

If you said P is NP tonight
There would still be papers left to write
I have a weakness
I'm addicted to completeness
And I keep searching for the longest Path

The algorithm I would like to see
Is of Polynomial Degree
Buts its elusive,
Nobody has found conclusive
Evidence that we can find the Longest Path

I have been hard
Working for so long
I swear its right,
But he marks it wrong

[0]Using the tune "The Longest Time"

Somehow I'll feel sorry when its done
GPA 2.1,
Is more than I hoped for

Garey, Johnson, Karp and other Men (and Women)
Try to make it Order n log n.
Am I a math fool
If I spend my life in Grad School
Forever following the Longest Path.

Woh oh-oh-oh
Find the longest path
Woh oh-oh-oh
Find the longest path

Appendix B

Experiment Results: Output Data-Extraction Ontologies (Schema Part)

This appendix includes the generated data-extraction ontologies for two application domains — apartment rentals and nation information. For each domain, we first present some sample training records, typically the ones we used to represent the domain of interest. Following the training records, this appendix presents three versions of the data-extraction ontologies for each domain: (1) the raw, automatically generated ontology; (2) the updated ontology from which the ontology builder has removed all the unwanted relationship sets;[1] and (3) the expert-created ontology used for comparison.

B.1 Apartment Rentals

B.1.1 Sample Training Records

Record 1 – from Provo Herald, checked 04/24/02

OREM condo 2 fmle prvt rms/ba, $325dep; $350mo, WD incl avail. 5/1 Emily 227-0445

Record 2 – from Provo Herald, checked 04/24/02

OREM Lakeridge Condo, furn, shrd rm/ba, DW, dryer, cbl, $250 + util BYU appr 224-1133

Record 3 – from L.A. Weekly, checked 04/24/02

Canoga Park House 3 bed, 1bath, formal dining room, W/D Hookups, Range,courtyard A/C, Fireplace, Hardwood Floors, yard, garage $1395 Get phone # address Free, No feeWe have others http://www.apartmenthunterz.com/ 310-276-4663 Posted 04/24/02

[1] Refer to Section 6.2 and Figure 6.7.

B.1.2 Version 1: Automatically Generated Data-Extraction Ontology

Apartment [-> object];

Apartment [0:1] IsA.ROOM.IsA.PLACE.LocationOf.SOCIALOBJECT.Subclasses AcademicOrganization [1:1];

Apartment [0:1] IsA.ROOM.Subclasses Bathroom [1:*];

Apartment [0:1] IsA.ROOM.Subclasses Bedroom [1:*];

Apartment [0:1] IsA.ROOM.IsA.PLACE.LocationOf.SOCIALOBJECT.Subclasses MilitaryOrganization [1:1];

Apartment [0:1] HasPhoneNumber PhoneNr [1:*];

Apartment [0:1] IsA.ROOM.TypeOf RoomType [1:*];

Apartment [0:1] IsA.ROOM.IsA.INTERIOR-BUILDING-PART.IsA.BUILDING-PART.Subclasses.ADDRESS.Subclasses US-AddressCity [1:1];

Apartment [0:1] IsA.ROOM.IsA.INTERIOR-BUILDING-PART.IsA.BUILDING-PART.Subclasses.ADDRESS.Subclasses US-AddressState [1:1];

Apartment [0:1] IsA.ROOM.IsA.INTERIOR-BUILDING-PART.IsA.BUILDING-PART.Subclasses.ADDRESS.Subclasses US-AddressZipCode [1:*];

AcademicOrganization [0:1] IsA.ORGANIZATION.AgentOf.WORKACTIVITY.Subclasses InsuranceServe [0:1];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses Building [0:1];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses ExteriorBuildingPart [0:1];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses Garage [0:*];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses Basement [0:1];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses FurniturePart [0:1];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses Room [0:*];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses CommunicationDevice [0:1];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses CleaningAppliance [0:1];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses EngineeredArtifact [0:1];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses AudioMediaArtifact [0:*];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses InsurancePolicy [0:*];

AcademicOrganization [0:1] IsA.ORGANIZATION.ObjectInvolved.FINANCIALOBJECT.Subclasses Investment [0:1];

AcademicOrganization [0:1] IsA.ORGANIZATION.HasPhoneNumber PhoneNr [1:*];

AcademicOrganization [0:1] IsA.SOCIALOBJECT.ThemeOf.SOCIALEVENT.Subclasses FinancialEvent [0:*];

AcademicOrganization [0:1] IsA.SOCIALOBJECT.ThemeOf.SOCIALEVENT.Subclasses Ceremony [0:1];

AcademicOrganization [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses LivingEvent [0:1];

AudioMediaArtifact [0:*] IsA.ARTIFACT.CostOfProduction Price [1:*];

AudioMediaArtifact [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Building [0:1];

AudioMediaArtifact [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses ExteriorBuildingPart [0:1];

AudioMediaArtifact [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Garage [0:*];

AudioMediaArtifact [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Basement [0:1];

AudioMediaArtifact [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses FurniturePart [0:1];

AudioMediaArtifact [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Room [0:*];

Basement [0:1] IsA.BUILDINGPART.PartOf Building [0:1];

Basement [0:1] IsA.PLACE.LocationOf.SOCIALOBJECT.Subclasses MilitaryOrganization [0:1];

Building [0:1] HasParts Room [0:*];

Building [0:1] HasParts.BUILDINGPART.Subclasses ExteriorBuildingPart [0:1];

Building [0:1] HasParts.BUILDINGPART.Subclasses Garage [0:*];

Building [0:1] HasParts.BUILDINGPART.Subclasses FurniturePart [0:1];

Building [0:1] IsA.ARTIFACT.CostOfProduction Price [1:*];

Ceremony [0:1] IsA.NONWORKACTIVITY.Location.PLACE.Subclasses Building [0:1];

Ceremony [0:1] IsA.NONWORKACTIVITY.Location.PLACE.Subclasses ExteriorBuildingPart [0:1];

Ceremony [0:1] IsA.NONWORKACTIVITY.Location.PLACE.Subclasses Garage [0:*];

Ceremony [0:1] IsA.NONWORKACTIVITY.Location.PLACE.Subclasses Basement [0:1];

Ceremony [0:1] IsA.NONWORKACTIVITY.Location.PLACE.Subclasses FurniturePart [0:1];

Ceremony [0:1] IsA.NONWORKACTIVITY.Location.PLACE.Subclasses Room [0:*];

Ceremony [0:1] IsA.SOCIALEVENT.Theme.SOCIALOBJECT.Subclasses MilitaryOrganization [0:1];

CleaningAppliance [0:1] IsA.ARTIFACT.CostOfProduction Price [1:*];

CleaningAppliance [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses ExteriorBuildingPart [0:1];

CleaningAppliance [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Garage [0:*];

CleaningAppliance [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses FurniturePart [0:1];

CleaningAppliance [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Room [0:*];

CommunicationDevice [0:1] IsA.ARTIFACT.CostOfProduction Price [1:*];

CommunicationDevice [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses ExteriorBuildingPart [0:1];

CommunicationDevice [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Garage [0:*];

CommunicationDevice [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses FurniturePart [0:1];

CommunicationDevice [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Room [0:*];

EngineeredArtifact [0:1] IsA.ARTIFACT.CostOfProduction Price [1:*];

EngineeredArtifact [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Building [0:1];

EngineeredArtifact [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses ExteriorBuildingPart [0:1];

EngineeredArtifact [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Garage [0:*];

EngineeredArtifact [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses FurniturePart [0:1];

EngineeredArtifact [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Room [0:*];

ExteriorBuildingPart [0:1] IsA.PLACE.LocationOf.SOCIALOBJECT.Subclasses MilitaryOrganization [0:1];

FaxNr [0:1] PhoneNumberOf.ORGANIZATION.Subclasses MilitaryOrganization [0:1];

FaxNr [0:1] PhoneNumberOf.ORGANIZATION.Subclasses AcademicOrganization [0:1];

FurniturePart [0:1] IsA.PLACE.LocationOf.SOCIALOBJECT.Subclasses MilitaryOrganization [0:1];

Garage [0:*] IsA.PLACE.LocationOf.SOCIALOBJECT.Subclasses MilitaryOrganization [0:1];

InsurancePolicy [0:*] ProducedBy.ORGANIZATION.Subclasses MilitaryOrganization [0:1];

InsurancePolicy [0:*] IsA.ARTIFACT.CostOfProduction Price [1:*];

InsurancePolicy [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses ExteriorBuildingPart [0:1];

InsurancePolicy [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Garage [0:*];

InsurancePolicy [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Basement [0:1];

InsurancePolicy [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses FurniturePart [0:1];

InsurancePolicy [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Room [0:*];

InsuranceServe [0:1] IsA.PROFESSIONALSERVICEEVENT.Location.PLACE.Subclasses Building [0:1];

InsuranceServe [0:1] IsA.PROFESSIONALSERVICEEVENT.Location.PLACE.Subclasses ExteriorBuildingPart [0:1];

InsuranceServe [0:1] IsA.PROFESSIONALSERVICEEVENT.Location.PLACE.Subclasses Garage [0:*];

InsuranceServe [0:1] IsA.PROFESSIONALSERVICEEVENT.Location.PLACE.Subclasses Basement [0:1];

InsuranceServe [0:1] IsA.PROFESSIONALSERVICEEVENT.Location.PLACE.Subclasses FurniturePart [0:1];

InsuranceServe [0:1] IsA.PROFESSIONALSERVICEEVENT.Location.PLACE.Subclasses Room [0:*];

InsuranceServe [0:1] IsA.WORKACTIVITY.Agent.ORGANIZATION.Subclasses MilitaryOrganization [0:1];

Investment [0:1] IsA.ASSET.OwnedBy.ORGANIZATION.Subclasses MilitaryOrganization [0:1];

LivingEvent [0:1] IsA.PHYSICALEVENT.Location.PLACE.Subclasses Building [0:1];

LivingEvent [0:1] IsA.PHYSICALEVENT.Location.PLACE.Subclasses ExteriorBuildingPart [0:1];

LivingEvent [0:1] IsA.PHYSICALEVENT.Location.PLACE.Subclasses Garage [0:*];

LivingEvent [0:1] IsA.PHYSICALEVENT.Location.PLACE.Subclasses Basement [0:1];

LivingEvent [0:1] IsA.PHYSICALEVENT.Location.PLACE.Subclasses FurniturePart [0:1];

LivingEvent [0:1] IsA.PHYSICALEVENT.Location.PLACE.Subclasses Room [0:*];

LivingEvent [0:1] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.Subclasses MilitaryOrganization [0:1];

LivingEvent [0:1] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.Subclasses CommunicationDevice [0:1];

LivingEvent [0:1] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.Subclasses CleaningAppliance [0:1];

LivingEvent [0:1] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.Subclasses EngineeredArtifact [0:1];

LivingEvent [0:1] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.Subclasses AudioMediaArtifact [0:*];

LivingEvent [0:1] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.Subclasses InsurancePolicy [0:*];

MilitaryOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses Building [0:1];

MilitaryOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses Room [0:*];

MilitaryOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses CommunicationDevice [0:1];

MilitaryOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses CleaningAppliance [0:1];

MilitaryOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses EngineeredArtifact [0:1];

MilitaryOrganization [0:1] IsA.ORGANIZATION.HasProductType.ARTIFACT.Subclasses AudioMediaArtifact [0:*];

MilitaryOrganization [0:1] IsA.ORGANIZATION.HasPhoneNumber PhoneNr [1:*];

MilitaryOrganization [0:1] IsA.SOCIALOBJECT.ThemeOf.SOCIALEVENT.Subclasses FinancialEvent [0:*];

Apartment [0:1] relatedTo ExteriorBuildingPart [1:*] relatedTo Garage [1:*] relatedTo Basement [1:*] relatedTo CommunicationDevice [1:*] relatedTo CleaningAppliance [1:*] relatedTo EngineeredArtifact [1:*] relatedTo AudioMediaArtifact [1:*] relatedTo AcademicOrganization [1:*] relatedTo Price [1:*] relatedTo InsurancePolicy [1:*] relatedTo Investment [1:*] relatedTo Characteristic [1:*] relatedTo Bathroom [1:*] relatedTo Bedroom [1:*] relatedTo PhoneNr [1:*] relatedTo Mileage [1:*] relatedTo Finance [1:*] relatedTo FurniturePart [1:*] relatedTo Building [1:*] relatedTo FaxNr [1:*] relatedTo FirstPersonName [1:*] relatedTo MilitaryOrganization [1:*] relatedTo Room [1:*];

Apartment [0:1] relatedTo ExteriorBuildingPart [1:*] relatedTo Garage [1:*] relatedTo Basement [1:*] relatedTo CommunicationDevice [1:*] relatedTo CleaningAppliance [1:*] relatedTo EngineeredArtifact [1:*] relatedTo AudioMediaArtifact [1:*] relatedTo Finance [1:*] relatedTo Custom [1:*] relatedTo Characteristic [1:*] relatedTo Investment [1:*] relatedTo Mileage [1:*] relatedTo FurniturePart [1:*] relatedTo Bathroom [1:*] relatedTo Bedroom [1:*] relatedTo PhoneNr [1:*] relatedTo Ceremony [1:*] relatedTo Price [1:*] relatedTo InsuranceServe [1:*] relatedTo LivingEvent [1:*] relatedTo Room [1:*];

B.1.3 Version 2: Data-Extraction Ontology with Extraneous Relationship Sets Removed

Apartment [-> object];

Apartment [0:1] IsA.ROOM.Subclasses Bathroom [1:*];

Apartment [0:1] IsA.ROOM.Subclasses Bedroom [1:*];

Apartment [0:1] HasPhoneNumber PhoneNr [1:*];

Apartment [0:1] IsA.ROOM.TypeOf RoomType [1:*];

Apartment [0:1] IsA.ROOM.IsA.INTERIOR-BUILDING-PART.IsA.BUILDING-PART.Subclasses.ADDRESS.Subclasses *US-AddressCity* [1:1];

Apartment [0:1] IsA.ROOM.IsA.INTERIOR-BUILDING-PART.IsA.BUILDING-PART.Subclasses.ADDRESS.Subclasses *US-AddressState* [1:1];

Apartment [0:1] IsA.ROOM.IsA.INTERIOR-BUILDING-PART.IsA.BUILDING-PART.Subclasses.ADDRESS.Subclasses *US-AddressZipCode* [1:*];

Building [0:1] HasParts.BUILDINGPART.Subclasses FurniturePart [0:1];

Building [0:1] IsA.ARTIFACT.CostOfProduction Price [1:*];

Apartment [0:1] relatedTo ExteriorBuildingPart [1:*] relatedTo Garage [1:*] relatedTo Basement [1:*] relatedTo CommunicationDevice [1:*] relatedTo CleaningAppliance [1:*] relatedTo EngineeredArtifact [1:*] relatedTo AudioMediaArtifact [1:*] relatedTo AcademicOrganization [1:*] relatedTo Price [1:*] relatedTo InsurancePolicy [1:*] relatedTo Investment [1:*] relatedTo Characteristic [1:*] relatedTo Bathroom [1:*] relatedTo Bedroom [1:*] relatedTo PhoneNr [1:*] relatedTo Mileage [1:*] relatedTo Finance [1:*] relatedTo FurniturePart [1:*] relatedTo Building [1:*] relatedTo FaxNr [1:*] relatedTo FirstPersonName [1:*] relatedTo MilitaryOrganization [1:*] relatedTo Room [1:*];

A composition of the three italicized concepts `USAddressCity`, `USAddressState`, and `USAddressZipCode` represent the single object set `Location` in the expert-created ontology. We also underline the concept `FirstPersonName` in the last n-ary relationship set to indicate that this concept of interest only appears in this n-ary relationship set and thus we should keep this n-ary relationship set in the ontology.

B.1.4 Version 3: Expert-Created Data-Extraction Ontology

Apt [-> object];

Apt [0:1] has *Location* [1:*];

Apt [0:1] has GenderReq [1:*];

Apt [0:1] has Furnish [1:*];

Apt [0:1] has Type [1:*];

Apt [0:*] has Renter Phone [1:*];

Apt [0:*] has Renter Name [1:*];

Apt [0:1] has Utility [1:*];

Apt [0:1] has Bdrms [1:*];

Apt [0:1] has Baths [1:*];

Apt [0:1] has DateAval [1:*];

Apt [0:*] has Feature [1:*];

Apt [0:1] has Month Rate [1:*];

Apt [0:1] has Deposit [1:*];

The italicized concept `Location` indicates that it is a composition of multiple object sets in the reduced, automatically generated ontology. The underlined concepts in this version of ontology mean that these object sets are of interest but were not defined in the knowledge base.

B.2 Nation Information

B.2.1 Sample Training Records

Record 1 – from Arab DataNet (http://www.arabdatanet.com/), checked 07/29/02

Egypt, at the northeast corner of Africa on the Mediterranean Sea, is bordered on the west by Libya, on the south by the Sudan, and on the east by the Red Sea and Israel. The country's total area is equal to 386,662 square miles, and is bisected by the Nile, the longest river in the world. The climate is hot and dry, with a mild winter.At the center of the Middle East-North Africa (MENA) region, Egypt has traditionally played a leading role in regional politics and commerce. The Egyptian economy has passed through several stages since the 1952 revolution, transforming from a state-led growth economy under the leadership of Gamal Abdel Nasser to one driven by market forces under Hosni Mubarak's presidency, via an open door policy (infitah) launched by Anwar Sadat in 1974. The most populous country in the Arab world, Egypt has developed a dynamic and attractive economy, which is one of the largest in the region.

Record 2 – from Countries of the World (http://www.factmonster.com/countries.html), checked 07/29/02

Afghanistan: Geography smaller than Texas. Area: 648,000 sq. km. Capital–Kabul, Other cities–Kandahar Mazar-e-Sharif Konduz Terrain: Landlocked; mostly mountains and desert. Climate: Dry, with cold winters and hot summers. Population:17.7 million. Agriculture: Wheat, corn, barley, rice, cotton, fruit, nuts, karakul pelts, wool, mutton.

B.2.2 Version 1: Automatically Generated Data-Extraction Ontology

```
Nation [-> object];
AnimalProduct [0:1] MaterialOf.ANIMALDERIVEDSUBSTANCE.Subclasses Wool [0:*];
AnimalProduct [0:1] IsA.PHYSICALOBJECT.Location.GEOLOGICALENTITY.Subclasses Field [0:*];
AnimalProduct [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses EnvironmentalEntity [0:*];
AnimalProduct [1:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];
```

AnimalProduct [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Region [0:*];
AnimalProduct [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses CapitalCity [0:*];
AnimalProduct [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];
AnimalProduct [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];
Area [1:1] IsA.SIZE.SizeOf.GEOGRAPHICAL-ENTITY.Subclasses.GEOPOLITICAL-ENTITY.Subclasses Nation [0:1]
CapitalCity [1:1] IsA.CITY.PartOf Nation [0:1];
CapitalCity [0:1] IsA.PHYSICALOBJECT.Location.GEOLOGICALENTITY.Subclasses Field [0:*];
CapitalCity [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses EnvironmentalEntity [0:*];
CapitalCity [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Region [0:*];
CapitalCity [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];
CapitalCity [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];
Corn [0:1] IsA.PLANTDERIVEDSUBSTANCE.MadeOf PlantPart [0:*];
Corn [0:1] IsA.PLANTDERIVEDSUBSTANCE.StateOfMatter.SOLID.Subclasses Cotton [0:*];
Corn [0:1] IsA.PHYSICALOBJECT.Location.GEOLOGICALENTITY.Subclasses Field [0:*];
Corn [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses EnvironmentalEntity [0:*];
Corn [1:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];
Corn [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Region [0:*];
Corn [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];
Corn [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];
Cotton [0:1] IsA.NATURALFABRIC.MadeOf Wool [0:*];
Cotton [0:1] IsA.PHYSICALOBJECT.Location.GEOLOGICALENTITY.Subclasses Field [0:*];
Cotton [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses EnvironmentalEntity [0:*];
Cotton [1:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];
Cotton [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Region [0:*];
Cotton [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];
Cotton [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];
CountryName [1:*] NameOf Nation [0:*];
CountryName [0:1] IsA.NAMEPLACE.NameOf.PLACE.Subclasses EnvironmentalEntity [0:*];
CountryName [0:1] IsA.NAMEPLACE.NameOf.PLACE.Subclasses Field [0:*];
CountryName [0:1] IsA.NAMEPLACE.NameOf.PLACE.Subclasses Region [0:*];
CountryName [0:1] IsA.NAMEPLACE.NameOf.PLACE.Subclasses CapitalCity [0:*];
CropPlant [0:1] IsA.PLANT.HasParts PlantPart [0:*];
CropPlant [0:1] IsA.PLANT.MadeOf.MATERIAL.Subclasses Wool [0:*];
CropPlant [0:1] IsA.PLANT.MadeOf.MATERIAL.Subclasses FruitFoodstuff [0:*];
CropPlant [0:1] IsA.PLANT.MadeOf.MATERIAL.Subclasses Corn [0:*];
CropPlant [0:1] IsA.PLANT.MadeOf.MATERIAL.Subclasses WheatFlour [0:*];
CropPlant [0:1] IsA.PLANT.MadeOf.MATERIAL.Subclasses Rice [0:*];
CropPlant [0:1] IsA.PHYSICALOBJECT.Location.GEOLOGICALENTITY.Subclasses Field [0:*];
CropPlant [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses EnvironmentalEntity [0:*];
CropPlant [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];
CropPlant [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Region [0:*];
EnvironmentalEntity [1:1] IsA.GEOGRAPHICALENTITY.LocationOf Nation [0:*];
EnvironmentalEntity [0:1] IsA.GEOGRAPHICALENTITY.PartOf.GEOPOLITICALENTITY.Subclasses Region [0:*];
EnvironmentalEntity [0:1] IsA.PHYSICALOBJECT.Location.GEOLOGICALENTITY.Subclasses Field [0:*];
EnvironmentalEntity [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];

EnvironmentalEntity [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];

Field [1:1] IsA.GEOLOGICALENTITY.LocationOf.SOCIALOBJECT.Subclasses Nation [0:*];

Field [0:1] IsA.GEOLOGICALENTITY.LocationOf.SOCIALOBJECT.Subclasses Region [0:*];

Field [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];

Field [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];

FruitFoodstuff [0:1] IsA.PLANTDERIVEDSUBSTANCE.MadeOf PlantPart [0:*];

FruitFoodstuff [1:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];

FruitFoodstuff [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Region [0:*];

FruitFoodstuff [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];

FruitFoodstuff [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];

Nation [0:*] HasParts City [1:*];

Nation [0:*] IsA.GEOPOLITICAL-ENTITY.IsA.GEOGRAPHICAL-ENTITY.BordersOn.GEOGRAPHICAL-ENTITY.Subclasses.
GEOPOLITICAL-ENTITY.Subclasses Nation [0:*]

Nation [0:1] HasPopulation Population [1:1];

Nation [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Region [1:1];

Nation [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [1:*];

Nation [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [1:*];

PlantPart [0:1] MaterialOf.PLANTDERIVEDSUBSTANCE.Subclasses WheatFlour [0:*];

PlantPart [0:1] MaterialOf.PLANTDERIVEDSUBSTANCE.Subclasses Rice [0:*];

PlantPart [0:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Region [0:*];

PlantPart [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];

PlantPart [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];

Region [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];

Region [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];

Rice [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];

Rice [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];

WheatFlour [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];

WheatFlour [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];

Wool [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [0:*];

Wool [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [0:*];

AnimalProduct [0:1] relatedTo Area [0:*] relatedTo Nation [0:*] relatedTo Region [0:*] relatedTo CapitalCity [0:*] relatedTo FruitFoodstuff [0:*] relatedTo Corn [0:*] relatedTo WheatFlour [0:*] relatedTo Rice [0:*] relatedTo Wool [0:*] relatedTo Cotton [0:*] relatedTo PlantPart [0:*] relatedTo EnvironmentalEntity [0:*] relatedTo Field [0:*] relatedTo CropPlant [0:*] relatedTo NaturalHazard [0:*] relatedTo Weather [0:*] relatedTo FieldOfStudy [0:*] relatedTo CountryName [0:*] relatedTo FinancialCapital [0:*];

B.2.3 Version 2: Data-Extraction Ontology with Extraneous Relationship Sets Removed

Nation [-> object];

AnimalProduct [1:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];

Corn [1:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:1];

Cotton [1:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:1];

CropPlant [1:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];

FruitFoodstuff [1:1] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];

Area [1:1] IsA.SIZE.SizeOf.GEOGRAPHICAL-ENTITY.Subclasses.GEOPOLITICAL-ENTITY.Subclasses Nation [0:1]

CapitalCity [1:1] IsA.CITY.PartOf Nation [0:1];

CountryName [1:*] NameOf Nation [0:*];

Nation [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses NaturalHazard [1:*];

Nation [0:*] HasParts City [1:*];

Nation [0:*] IsA.GEOPOLITICAL-ENTITY.IsA.GEOGRAPHICAL-ENTITY.BordersOn.GEOGRAPHICAL-ENTITY.Subclasses.GEOPOLITICAL-ENTITY.Subclasses Nation [0:*]

Nation [0:1] HasPopulation Population [1:1];

Nation [0:1] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Weather [1:*];

A union of the five italicized concepts `AnimalProduct`, `Corn`, `Cotton`, `CropPlant` and `FruitFoodstuff` represent the single object set `AgriculturalProduct` in the expert-created ontology.

B.2.4 Version 3: Expert-Created Data-Extraction Ontology

Nation [-> object];

Nation [0:1] has Name [1:1];

Nation [0:1] has Size [1:*];

Nation [0:1] has Capital [1:1];

Nation [0:*] has City [1:*];

Nation [0:1] has Population [1:*];

Nation [0:1] has Climate [1:*];

Nation [0:1] has Terrain [1:*];

Nation [0:*] has SurroundingCountry [0:*];

Nation [0:*] has *AgriculturalProduct* [1:*];

The italicized concept `AgriculturalProduct` indicates that it is a union of multiple object sets in the reduced, automatically generated ontology.