IAWTIC 2003 Proceedings/ISBN 1740880692: M. Mohammadian (Ed.)
12-14 February 2003, Vienna - Austria

A System For Information Extraction And Intelligent Search Using Dynamically Acquired Background Knowledge

Samhaa R. El-Beltagy, Ahmed Rafea, and Yasser Abdelhamid
Central Lab for Agricultural Expert Systems
Agricultural Research Center
Ministry of Agriculture and Land Reclamation.
Cairo, Egypt
El-Nour St. Dokki 12311,Giza Egypt
E-mail: {samhaa, rafea, yasser}@mail.claes.sci.eg

Abstract

This paper presents a simple framework for extracting information found in publications or documents that are issued in large volumes and which cover similar concepts or issues within a given domain. The general aim of the work described, is to present a model for automatically augmenting segments of these documents with metadata using dynamically acquired background domain knowledge in order to assist users in easily locating information within these documents through a structured front end. To realize this goal, both document structure as well as dynamically acquired background knowledge, are utilized. A real life example where these ideas have been applied is also presented.

1 Introduction

Enterprises and organizations often contain information rich texts, but rarely have the means by which these resources can be intelligently searched. In many cases, the search interface that is adopted is one based on keywords and though the indexing/matching techniques employed may vary in their degree of sophistication, this approach suffers from the same limitations associated with the existing Web search model [1]. This paper addresses the particular problem of trying to extract information from organizational publications that are issued in large volumes and which cover similar concepts or issues and from which information cannot be extracted through the use of the structure of a document alone. The end goal, is to enable individual sections of those documents to be automatically augmented with metadata, so that users can perform structured search using a predefined set of categories or classifications and obtain as a result, only segments or sections of documents that fit their search criteria. The class of documents targeted by this work is thus that of resources that contain a set of information entities most of which fall under known categories, but which contain no special markup to differentiate them from other information entities. The approach adopted towards this problem is to attempt to make use of background knowledge about those categories and to employ that for intelligent search. Rather than forcing predefined static background knowledge, the work presented allows for the dynamic acquisition of this knowledge as the system evolves. Our goal is thus two fold; the first is to provide the tools that can assist in ontology building and to utilize the background ontology for document indexing, and the second is to provide an intelligent interface to allow for the retrieval of the stored information.

2 Related Work

Information is vital resource to individuals and organizations as its timely location can influence key decisions that affect both. It is thus no wonder that massive research efforts have been undertaken in recent years with the aim of improving upon existing search facilitates specially among unstructured and semi-structured resources where the problem of information finding is most pronounced. Looking into ways for extracting information from semi-structured texts has been investigated in many system integration projects [2] such as TSIMMIS[3] and Lore [4]. These systems have aimed to provide an integrated view to related data scattered across various structured and semi-structured resources and have thus developed templates and wrappers to extract structured information from semi-structured texts. The primary goal of such systems was to unlock the wealth of information stored within legacy applications and to integrate those with other related/similar data available in other resources. Towards this end, specific languages, representation models, and anthologies were designed and adopted.

Also, much work has been carried out within the knowledge acquisition community with the aim of providing automatic support for the extraction of information from un-structured texts. This task is still proving to be a rather challenging one. Information Extraction (IE) systems have thus appeared with a more focused goal of supporting the task of extracting information from specific domains or for particular tasks [5]. IE systems often rely on templates, hand generated annotations, or domain dependant NLP knowledge. For example, the SoftMealy system [6] and the system presented in [7], are both IE systems that attempt to extract information from Web pages through examples of such pages all of which exhibit similar structure. These systems work when structure templates of well defined fields of content exist. For example, a page containing some country codes, may have the name of a country formatted in bold and the code for that country is formatted in italics [7]. It is possible then, to use this formatting information to extract country-code pairs. However, it is often the case that structure or formatting on its own can not be used to extract information. One of the solutions purposed to over come this obstacle, is to tag the information in a way that would enable its extraction. Indeed, XML [8] emerged to achieve precisely that. Taking this idea a step further is the approach that has been adopted by SHOE [9] [10]. SHOE is a web based knowledge representation language that can be embedded in web pages. By explicitly specifying the ontology being be used within a web page and tagging information within that page using that ontology, it is possible to appropriately extract information from that page as well as infer relations and information not explicitly represented. With respect to existing documents, the problem with the tagging approach in general, is that if its not used in conjunction with an automatic tagging mechanism, it will require the re-authoring of vast amounts of information.

3 Problem Scope and Definition

It is often the case that a broad range of documents containing useful information exists, but with no way to access individual segments of these documents directly using targeted or structured search. Any document is typically divided into a number of sections and sub-sections. For example, documents that cover common problems related to various electrical appliances and their solutions will usually have sections for each class of problems, each of which will have subsections that cover a specific problem belonging to that class. Without targeted search, a user interested in finding a solution to a particular problem related to a specific electrical appliance, must first try to locate the specific document that covers common problems and their solutions for that appliance, and then begin the tedious task of browsing that document in order to locate the problem he/she is interested in. A search engine that would allow the user to select the appliance for which he/she is attempting to find a solution, then allow the user to select the specific problem he/she is interested in, and finally return the exact section that covers that problem, would certainly save the user valuable time and effort. The same interface, may also allow a user to compare how a given problem is solved across a range of appliances.

Moving beyond this simple and hypothetical example, in this work, we've had to address a real problem related to agricultural extension documents issued primarily to assist farmers in cultivating and caring for certain crops. Each document is information rich with respect to the crop which it covers. Depending on the importance of a given crop and how involved the issues related to it are, a crop may have more than one document to address it. Because of the wealth of information contained within these documents, they're often used by researchers as well by farmers and extension workers. A typical document will cover most aspects related to cultivating a crop, starting from land preparation to harvest. Each section within a document targets a given problem or issue, and each sub-section embodies a specialization of that issue. For example, a section called 'Diseases', will have as its subsections most diseases that are likely to affect a given crop. Similarly, a section covering operations, will cover all agricultural operations that apply to that crop. In this case and in similar cases, there are two elements that can work in the advantage of an intelligent search. The first is that the main elements of search can be identified before hand over a broad class of documents. 'Diseases' and 'Operations' are two examples of search categories that can be readily identified. The second element, is that individual mapping of instances related to the categories, are more or less the same across all documents and are featured in either section or subsection headings. For instance, 'Fertilization', 'Irrigation', and 'Land Preparation' , all belong to the class of agricultural operations, while 'Powdery Mildew' belongs to the class of agricultural diseases. These classes and their instances will usually generalize across all crops. So, the individual instances of these general categories embody background knowledge that can be added to individual document segments as meta-data. There are some cases however, when a general category can be identified, but the instances of which will rarely recur in many documents. Crop 'Varieties' is an example. In most extension documents, there is usually a section on varieties with various sub sections on each variety and it's different features. The name of a crop variety is specific to that crop and as such cannot be used as a general search term. To enable the location of information on any given variety for a given crop, the hierarchy of the document itself can be utilized to infer that each subsection of any section covering 'Varieties' is an instance of the general category 'variety'.

Generally speaking, augmenting various document sections with metadata involves a number of steps which can be summarized as follows:

- Identifying the various categories onto which various document sections can be mapped.
- Acquiring and representing background knowledge in a way that can facilitate the mapping of various document sections into the identified categories.
- Segmenting various documents, and employing background knowledge to map each document section to its corresponding category
- Storing structured index information in a persistent data store such as a database, or converting the document into an alternate representation (ex. XML).
- Providing a user interface to enable search across indexed documents.

4 Modeling Blocks

In this work, it was important to adopt a flexible yet powerful way of representing both background information as well as a document. XML [8] was thus adopted to represent both. Background information is stored in an XML file which is used to represent index terms. The file has the following structure:

```
<indexTerms>
        <general_category indexChildNodes= "true" >
                <name> diseases </name>
                <sameAs> disorders </sameAs>
        </general_category>

        <general_category indexChildNodes= "true" >
                <name> Varieties </name>
        </general_category>

        <disease indexChildNodes= "false" >
                <name>Powdery Mildew</name>
                <sameAs> aSynonym </sameAs>
                <sameAs> ……….. </sameAs>
        </disease>
        …
        …
        <operation indexChildNodes= "false" >
                <name> aNameOfanOperation </name>
                <sameAs> aSynonym </sameAs>
        </operation>
        …
        …
        <pest indexChildNodes= "false" >
                <name> aNameOfaPest </name>
        …
        …
        </pest>
        …
        …
</indexTerms>
```

Figure 1: The XML representation of background knowledge

This representation, despite its simplicity, allows for the mapping of various phrases to their corresponding categories, as well as provides a simple thesaurus using the <sameAs> tag. The *indexChildNodes* can be used to specify whether or not specializations of a given term should be indexed as belonging to that term, i.e. whether or not a document's hierarchy is to be utilized.

A document will have the XML representation illustrated in Figure 2

```
<doc>
        <title> aTitle </title>
        <section>
                <id>102328933656>/id>
                <level>1</level> ✍the level of a section within a document hierarchy ✍
                <heading> the text heading of the section </heading>
                <text> a pure text representation of the contents of the section </text>
                <html> <![CDATA[ the html text representation of this section ]] < /html>
        </section>
        <section>
                …..
                …..
        </section>
</doc>
```

Figure 2: The XML representation of an un-indexed document

5 System Overview

The implemented system is a distributed one, in which a number of components communicate together to achieve the required functionality. The main components of this system are: an indexing user interface, an indexing backend linked to a DBMS, and a search front end also linked to a DBMS. Figure 3 shows the various components each of which is described in the following sub-sections, and their interactions.

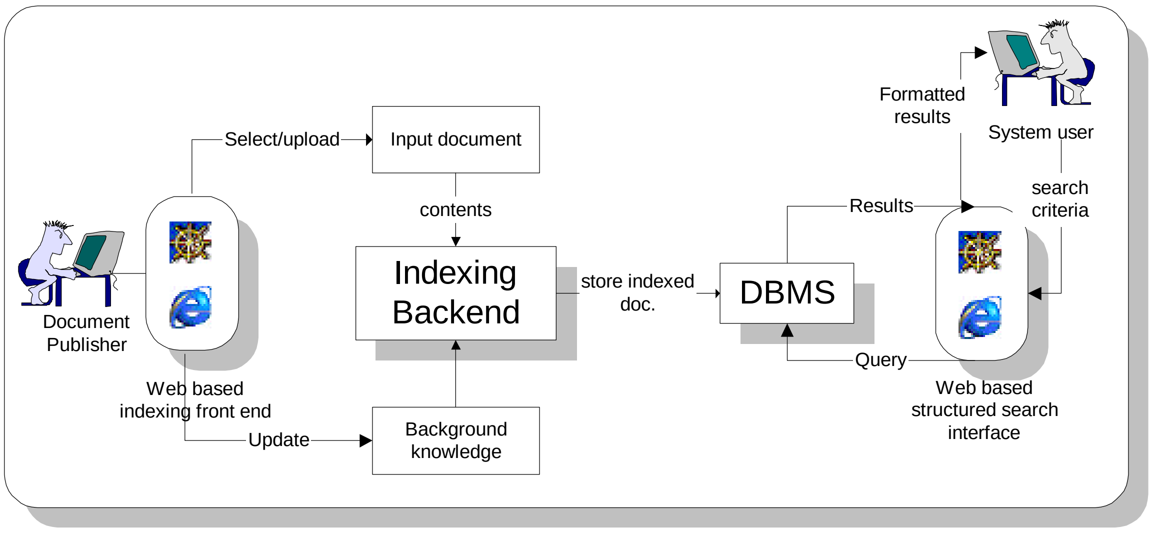

Figure 3: System components and interactions

5.1 The Indexing Backend

The indexing backend is the component responsible for augmenting input documents with meta-data using background knowledge. The indexing backend is implemented in Java as a multithreaded HTTP server that is capable of receiving indexing requests embedded in HTTP requests. On startup, the system loads the XML representation of background knowledge into a set of dictionaries and data structures that can facilitate the indexing process. A request to this component will contain the URL of the file that requires indexing as well as the name of the crop for which this file belongs. Before carrying out any indexing, the component starts reading the specified document and breaks it down to the structure specified in figure 2.

Following this segmentation phase, pattern matching techniques are applied to match section heading titles with index terms. An index record for each section is created, with each record containing fields for every pre-identified category (one for diseases, another for operations, etc). Should a match be made between a heading and one of the input index terms, then the category of the section will be deduced and the field designated for that category will be filled with an ID pointing to the specific instance against which a match was made. A single section may match with more than one category. After the analysis of a given section is completed and a record is created accordingly, the record along with a pointer to the specific section for which it was derived are sent to a remote storage component (a database) where they are kept. After analysis of the whole document is completed, an HTML page is returned to the user. Within that page, all section and subsection headings are displayed and besides each, it is indicated whether that section has been indexed or not and if indexed, whether indexing was performed directly or indirectly (through the use of hierarchical information). Sections that have not been indexed are hyperlinked to an interface which allows the user to edit their text so as to update the background knowledge and re-index the input document. Updating background knowledge can involve the creation of a new category instances, or the creation of synonyms to associate with existing ones. The update request is encoded in a URL sent to indexing backend over HTTP. The indexing backend subsequently 'learns' this new information and updates its background knowledge file. Initially, some background knowledge could be acquired from a domain expert, or it could be completely learned through the indexing process (which also requires usage by someone familiar with the domain).

5.2 The Indexing User Interface

Since it is anticipated that those who will request document indexing will do so remotely, a Web interface for facilitating the indexing and uploading of extension documents was implemented. This interface simply allows a user to select an extension document from their local machine, uploads it to a Web server, and then indexes this document through communication with the indexing backend.

5.3 The Search Front end

A Web search front end is provided to allow users to rapidly fetch their required information from the extension documents by selecting one or more values for index parameters, where the index parameters are those of the crop name as well as pre-defined indexing categories. The number of selected parameters defines whether the query will be a loose or a specific one. The more specific the query, the less number of records are returned. After a query is entered, it is converted to SQL and dispatched to the DB in which indexing information has been stored. The result is displayed in the form of an html page containing a list of index records that match the entered query. The output includes the following: the heading title of the matching section, a sample from the matching paragraph, a hyperlink to the source section. On following the hyperlink, only the text of the selected section will be displayed. However, depending of the level of a section, extra information that defines the context of the section as part of the whole document, might be displayed. In addition, a hyperlink to the source document will always be displayed.

6 Conclusion and Future Work

This paper addresses the particular problem of attempting to locate, using a user friendly structured interface, information in organizational publications that are issued in large volumes and which cover similar concepts or issues. The general aim of the work described, is to present a model for automatically augmenting segments of these documents with metadata using dynamically acquired background domain knowledge in order to assist users in easily locating information within these documents through a structured front end. The technique used to achieve this goal, is a simple but powerful one, which could be generalized to apply to any collection of documents that cover similar concepts within a known domain.

In building our prototype, the main categories under which extension document headings could be classified, were hard wired into the code and for each of these categories, a table was created in the database. To enable our technique to work with any kind of document, we intend to remove any hardwired information and to allow for the definition of categories by the indexing user, i.e by a user that knows enough about the domain and the documents. We will also extend our tool in order to enable it to automatically create any required DB tables, and to dynamically generate the search interface. This will make our tool more generic and will enable its application in any domain.

Another area of future work that we intend to pursue, is that of the agentification of the search component. By doing so, we will allow other agents within an agent based framework to make use of it. For example, an expert system agent may use this service to link its conclusions with information available about these conclusions within the brochures.

Acknowledgments
This work has been supported by FAO grant TCP/EGY/065

References

[1] S. El-Beltagy, "Context, Queries, and the Web," University of Southampton, Southampton, UK ECSTR-IAM01-002, 2000.

[2] S. El-Beltagy, "Approaches to System Integration For Distributed Information Management," University of Southampton, Southampton, UK ECSTR MM98/7, 1998.

[3] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS," presented at AAAI Symposium on Information Gathering, Stanford, California, USA, 1995.

[4] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, "Lore: A Database Management System for Semistructured Data," *SIGMOD Record*, vol. 26, pp. 54-66, 1997.

[5] M. Vargas-vera, J. Domingue, Y. Kalfoglou, E. Motta, and S. Buckingham Shum, "Template-driven Information Extraction for Populating Ontologies," presented at IJCAI 2001 workshop on Ontologies Learning, Seattle, USA., 2001.

[6] C. Hsu, “Initial results on wrapping semistructured web pages with finite-state transducers and contextual rules,” presented at AAAI-98 Workshop on AI and Information Integration, Madison, WI, USA, 1998.

[7] N. Kushmerick, D. S. Weld, and R. B. Doorenbos, “Wrapper Induction for Information Extraction,” presented at Intl. Joint Conference on Artificial Intelligence (IJCAI), 1997.

[8] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, “Extensible Markup Language (XML) 1.0,” World Wide Web Consortium http://www.w3.org/TR/1998/REC-xml-19980210, 1998.

[9] J. Heflin and J. Hendler, “Dynamic Ontologies on the Web,” presented at *the Seventeenth National Conference on Artificial Intelligence (AAAI-2000),* Menlo Park, CA, USA, 2000.

[10] J. Heflin and J. Hendler, “Searching the Web with SHOE,” presented at AAAI-2000 Workshop on AI for Web Search, 2000.