A Framework for Information Extraction, Storage and Retrieval

Samhaa R. El-Beltagy[ϒ], Mohammed Said*, and Khaled Shaalan[ϒ]

[ϒ]Department of Computer Science
Faculty of Computers and Information
Cairo University
5 Tharwat Street, Orman,
Giza 12613, Egypt

*Central Lab for Agricultural Expert Systems
Agricultural Research Center
Ministry of Agriculture and Land Reclamation.
El-Nour St., P.O. Box 438, Dokki
Giza, Egypt

E-mail: *{samhaa, moh_said, shaalan}@claes.sci.eg*

Abstract

This paper presents a set of tools that were developed in order to facilitate and speed up the process of building information extraction and retrieval systems for documents that exhibit a set of predefined characteristics. Specifically, the work 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 paper presents a simple model for defining background knowledge and for using that to automatically augment segments of input documents with metadata in order to assist users in easily locating information within these documents through a structured front end. The model presented makes use of both document structure as well as dynamically acquired background knowledge to achieve its goals.

1 Introduction

Locating information in any given document in a precise and targeted manner is the goal of any advanced search system. This is particularly true within enterprises and organizations that often have volumes of information rich texts, but rarely have the means by which these resources can be intelligently searched. The simple keyword model has already demonstrated its inability to cope appropriately with such resources [1]. In this work we present a model that aims to address this problem in 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. Specifically, the goal of this work is to enable individual sections of such 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. In prior work [2], we investigated this area of work for a specific set of documents (agricultural extension documents). Using our method we've approached an accuracy of 100% when retrieving document sections based on a set of predefined categories. The presented implementation however, was not generic and thus did not enable us of extending our proposed ideas to "any" set of documents. In this paper we show how the previously described work was extended into a very flexible and generic framework that can enable: the creation of a set of specifiers for background knowledge, dynamic acquisition of instances of those specifiers, indexing and tagging of documents using these specifiers, and finally the generation of a search interface for the retrieval of stored tagged information.

2 Problem scope

The class of documents targeted by this work is 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. Taking extension agricultural documents as an example, a typical document will cover most issues 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 tagging and 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. User manuals, electrical appliance guides, and many educational materials are further examples of documents that exhibit similar features to the one described above. 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.

In the developed system, three stage are involved in carrying out the above outlined steps. In the first stage, a tool is provided to aid the system developer in ontology building. In the second stage, indexing of document sections and metadata augmentation based on background knowledge is carried out. In this stage also, dynamic acquisition of background knowledge may take place. This allows for background knowledge to evolve over time rather than force its pre-definition. In the third stage, a flexible and structured search front end is automatically generated using the provided background knowledge. The generated interface allows the stored information to be queried in an intelligent manner.

3 System Overview

The implemented system is composed of a number of components that communicate together in order to achieve the required functionality. The main components of this system are: a background knowledge editor, an indexing/meta-data tagger linked to a DBMS, and a user interface generator which creates a search front end which is also linked to a DBMS. Figure 1 shows the various components and their interactions. The tool currently has built in support for both Arabic and English.

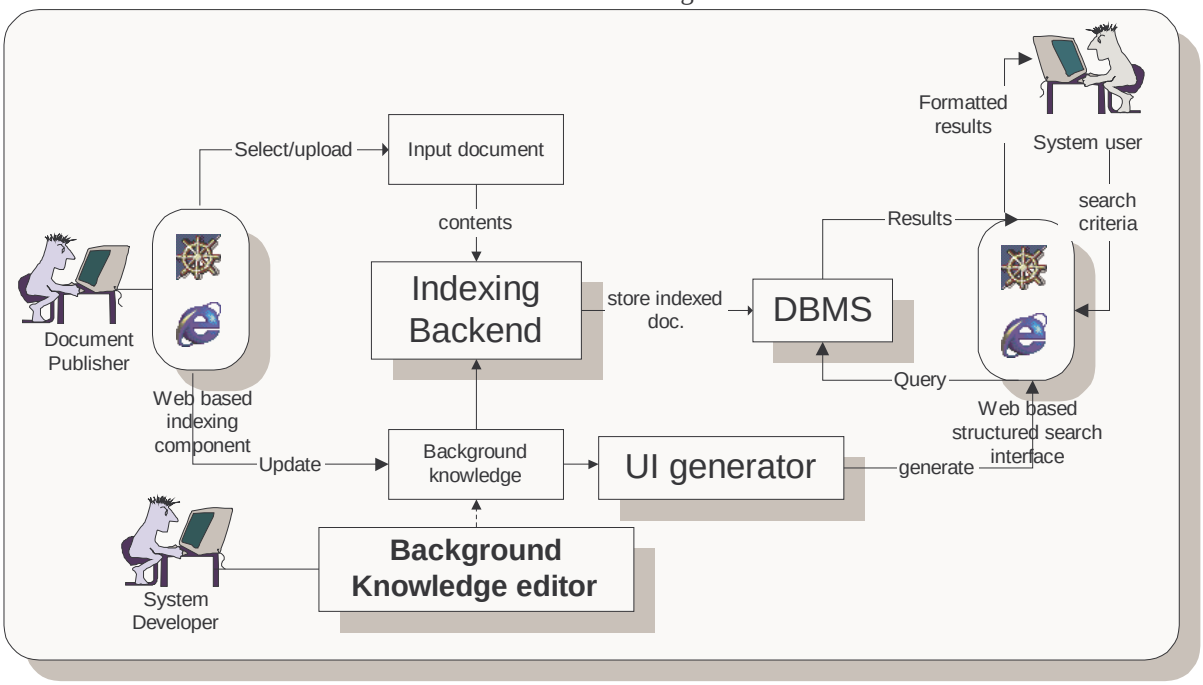

Figure 1: System components and interactions

3.1 The Background Knowledge editor

The background editor serves as the starting point for the development of any information extraction system within the presented framework. Using this tool, the system developer creates the basic categories under which classification can later take place. Once the main categories are created, instances of these categories can be filled also using this editor. As mentioned before, not all instances need to be defined at the beginning. Instead, these could be acquired dynamically during the indexing phase. When defining a category, the developer has the option of defining all terms that match the category both in English and in Arabic thus representing a "same as" relationship between the category's name and all equivalent names. The developer also has the option of specifying a category as a "parent" category, meaning that if the category is encountered in a document, all sections that fall underneath it can be automatically classified as instances of this category even if no instance information has been explicitly entered for this category. This feature is particularly useful for categories whose instances will not usually generalize over the entered set of documents. Using the agricultural extension documents example, the category of 'Varieties' illustrates this point. 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 some 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'.

In addition to defining categories and their instances, the background knowledge editor can also be used to modify category and instance information at any point in time. When the user selects to save the background knowledge, an XML [3] file is created is created for that knowledge. XML was adopted as it provides a flexible yet powerful way of representing both background information as well as a document content. A simple example of the generated file is shown below:

```
- <Knowledge DomainName="Agriculture">
  - <Category Name="diseases" Lang="En" IndexChildNode="True">
      <SameAs lang="En">disorders</SameAs>
      <SameAs lang="Ar">امراض</SameAs>
    - <Instance Name="stem rust" Lang="En">
        <SameAs lang="Ar">الساق صدا</SameAs>
      </Instance>
    </Category>
  </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. In addition to generating this XML, the editor also creates an Access DB file with appropriate tables and initializes it with information entered by the system developer.

3.2 The Indexing/ Metadata tagger Component

This indexing/metadata tagger, is the component responsible for augmenting input documents with meta-data using created background knowledge and with storing the result in a persistent data store. Using this component a user can select any html documents he/she wants indexed and tagged. The user also selects the background knowledge he/she wants applied in the tagging process. The indexer starts off by segmenting the document into sections using the document's html heading tags. The indexer then tries to classify each section by mapping its header to categories or instances contained in the background knowledge. To carry out the mapping process pattern matching techniques are applied. 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. But if the matching fails the system can update its information through a simple "learning module" that allows the document publisher to update the background knowledge by classifying the un-mapped section. Updating background knowledge can involve the creation of new category instances, or the creation of synonyms to associate with existing ones. Initially, some background

knowledge could be acquired from a domain expert. Alternatively, it can be completely learned through the indexing process (which also requires usage by someone who is familiar with the domain). Figure 2 summarizes the operation of this component.

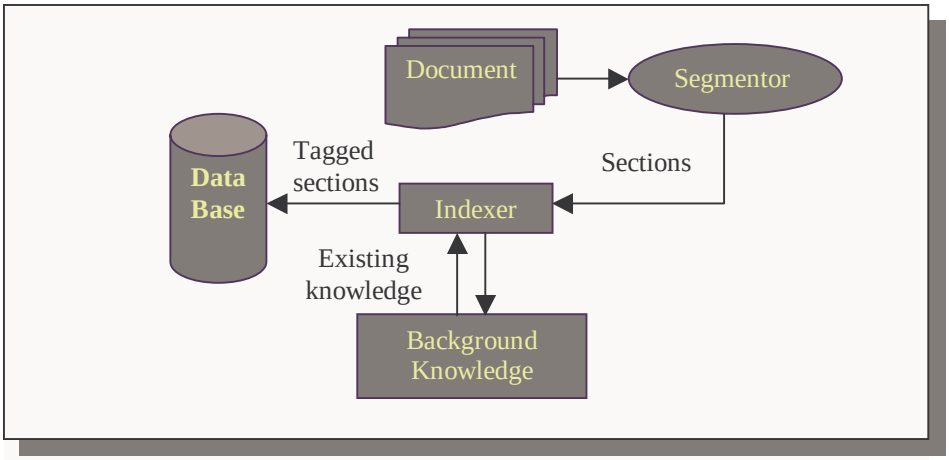

Figure 3: The indexer/tagger component in action

3.3 The Search interface

User interface development, usually takes up significant time in the implementation of any system. As the goal of developing this framework was to facilitate and expedite the process of building an IE system given that the input documents exhibit the characteristics mentioned in section 2, automating the process to user interface development was recognized as an important part of achieving this goal. To automate this process, use has been made of the created background knowledge as it clearly defines the structure through which search can be carried out. Using this UI, any user can rapidly retrieve his/her required information by selecting one or more values for the index parameters, where the index parameters are those of pre-defined background categories. Using pull down menus for each category, users can be very specific in their query by entering the precise instance for a given category in which they are interested. The number of selected parameters defines whether the query will be a loose or a specific one. The more specific the query, the lesser the number of records returned. After a query is entered, it is converted to SQL and dispatched to the database in which indexing information has been stored. The result is 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 on 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.

4 Background

Due to its importance, a lot of work has been carried out in the area of information extraction. Information extraction can be defined as a process which takes as an input unstructured or semi-structured texts and produces as output clear structured data. The produced data maybe used for any number of purposes such as storage in a database, direct display to users, or for analysis by some intermediary process. Looking into ways for extracting information from unstructured or semi-structured texts has been investigated long before the term information extraction came into existence. For example, many system integration projects [4] such as TSIMMIS[5] and Lore [6] have looked into ways for extracting information from semi-structured texts. 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 and representation models were designed and adopted. Similarly, 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. When information Extraction (IE) systems appeared, they did so with the more focused goal of supporting the

task of extracting information from specific domains or for particular tasks [7]. Most IE systems rely on templates, hand generated annotations, or domain dependant NLP knowledge. For example, the SoftMealy system [8] and the system presented in [9], 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 [9]. 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. This is the approach we've aimed to followed when developing the outlined system. The most closely related work to the one described in this work is our previous work in this area [2]. However, there are many differences and enhancements that have been brought forth by the developed framework. The following table highlights the major differences between the work presented herein and that presented in [2]:

Previous system	**Presented System**
Can support only one application	Can support any number of applications
Application features hardwired into source code	No hard wiring of any application features required
Database tables and records fixed	Database tables and records generated dynamically
Background knowledge edited manually using any XML editor	Background knowledge edited using an easy to use graphical UI that requires no knowledge of XML
User interface hand crafted	User interface generated automatically
Developed in Java and ASP	Developed in C# and ASP .Net

5 Conclusion

This paper presented an easy to use, and flexible framework for the development of information extraction systems for any class of publications that are issued in large volumes and which cover similar concepts or issues. By allowing a system developer to define/his or her main categories and later dynamically acquire instances of these categories, the framework allows the IE system to evolve over time. Also, by hiding all implementation details from the system developer and automating the creation of a user interface for search, the framework allows a user with very limited computer skills to develop a powerful information extraction and retrieval system in very little time.

References

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

[2] S. El-Beltagy, A. Rafea, and Y. Abdelhamid, "Chapter 13: Using Dynamically Acquired Background Knowledge For Information Extraction And Intelligent Search," in *Intelligent Agents for Data Mining and Information Retrieval*, M. Mohammadian, Ed. Hershey, PA, USA: Idea Group Publishing, 2004.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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