ANNOTATING WEB SEARCH RESULTS

A Thesis Presented to the Department of

Computer Science and Engineering,

The African University of Science and Technology

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

By

Sam Manthalu

Abuja, Nigeria

December, 2014

ANNOTATING WEB SEARCH RESULTS

By

Sam Manthalu

A THESIS APPROVED BY THE COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

RECOMMENDED: .....

Supervisor, Professor David Amos

.....

Head, Department of Computer Science

APPROVED .....

Chief Academic Officer

.....

Date

Abstract

With more than millions of pages, the Web has become a greatly enormous information source. This information is in form of documents, images, videos as well as text. With such vast sizes of data, it is a common problem to get the right information that one wants. Oftentimes users have to search for the right content they are looking for from the Web with the help of search engines. Searching can be done manually by use of available platforms like Google or automatically in form of web crawlers.

Since the semantic web is not structured, search results can include varying types of information relating to the same query. Sometimes these results cannot be directly analyzed to meet the specific interpretation need. The search result records (SRRs) returned from the Web following manual or automatic queries are in form of web pages that hold results obtained from underlying databases. Such results can further be used in many applications such as data collection, comparison of prices etc. Thus, there is a need to make the SRRs machine processable. To achieve that, it is important that the SRRs are annotated in a meaningful fashion. Annotation adds value to the SRRs in that the collected data can be stored for further analysis and makes the collection easier to read and understand. Also annotation prepares the data for data visualization. The SRRs bearing same concepts are grouped together thus making it easier to make comparisons and analyze and go through the collection.

The purpose of this research is to find out how search results from the Web can be automatically annotated and restructured to allow for data visualization for users in a specific domain of discourse. A case study application is implemented that uses a web crawler to retrieve web pages about any topic in public health domain. This research is a continuation of the work done by Mr. Emanuel Onu in the project “Proposal of a Tool to Enhance Competitive Intelligence on the Web”.

ACKNOWLEDGEMENT

I would like to thank my supervisor Professor David Amos for continued support and guidance. His availability and willingness to help and support made this work possible. I would also like to thank the Head of Department, Computer Science and Engineering Professor M. K. Traore for his insights and assistance throughout the whole period of study.

Special thanks go to the Director and Stuff, ICT Department of Chancellor College, University of Malawi for all manner of help and support rendered to me throughout the entire study period.

I would also like to thank African University of Science and Technology for opening doors and opportunities to study. This includes all Administrative and Support Staff that have made my stay bearable.

I appreciate the entire Faculty for their effort and knowledge impartation in all courses I attended.

To the African Development Bank, thank you for your generous sponsorship.

Finally, to all fellow students, your support, help and encouragement has seen us stand together as one. Thank you.

DEDICATION

To God, the Lord Almighty who has stood by me through clouds and sunshine. To Him be all glory and honor.

"*Delight thyself also in the LORD; and He shall give thee the desires of thine heart"*

Psalm 37 : 4

To my late Dad, the man who had a dream and the dream lives after him.

To my mum, a woman of virtue, brothers and sisters whose prayers and support and love are beyond words.

To all friends and loved ones, your support has always been amazing. God bless you all.

1 Introduction

People of all walks of life use the internet for so many different tasks such as buying and selling items, social networking, digital libraries, news, etc. Researchers need information from digital libraries and other online document repositories to conduct their research and share information; scholars need books to get information and knowledge from; people communicate to one another through emails via the Web; others utilize social media to exchange information as well as having casual chat; some conduct transactions like purchasing items and paying for bills via the web. The World Wide Web is today the main "all kind of information" repository and has been so far very successful in disseminating information to humans. The Web has become the preferred medium for many database applications, such as e-commerce and digital libraries. Many database applications store information in huge databases that users' access, query and update through the Web.

The improvement in hardware technologies has seen increase in computers and server's storage capacity. As such, many web servers store a lot of data in their storage drives. In some social media websites e.g. Facebook[1], users can upload pictures, videos as well as other documents. YouTube [2] allows its users to post videos of varying lengths to their servers. There are other automated systems that collect a lot of data on daily basis. For example, bank systems need to store daily Auto Teller Machine (ATM) transactions as well as other customers' transactions. Some monitoring systems collect data about some aspect of life e.g. climate change, online shopping systems that keep information about the clients' daily shopping experience. These are some but few ways that have led to a gigantic amount of information and documents to be available on the Web.

However, due to the heterogeneity and the lack of structure of Web information sources, access to this huge collection of information has been limited to browsing and searching. That is, for one to access a document, you need to put the URL (Universal Resource Locator) in the Web Browser or making use of a search engine. The former way is suitable when you know what you are looking for and the exact location on the Web. But this is hardly the case and as such many of the Web users locate particular content they are looking for by using search engines. There are software systems that require a user to manually enter a search term and the search engine retrieves documents according to the term entered by the user; while there are also other automated search engines that make use of a Web Crawler.

There are several notable web based search engines that index web documents and are available for use by Web users. Most common ones are Google, Yahoo, AltaVista and many more. Such systems search through the collection of the documents sourced from the Surface Web – which is indexed by standard search engines as well as the Deep Web –which requires some special tools to be accessed. Most users benefit from such systems when researching information that is not known or they want to redirect to trace a website they know but can't remember its URL.

Still, there are some business disciplines such as Competitive Intelligence [3] that require particular type of information (domain specific) in order to make strategic business decisions. In such scenarios, different tools are developed in order to help in information gathering and analysis. Several other methods for searching and information retrieval to gather intelligence also work in such domain specific areas. For example, manually browsing the Internet could be the simplest method for conducting a Competitive Intelligence task. Manual browsing of the Internet to a reasonable level guarantees the quality of documents collected which in turn improves the quality of knowledge that is discoverable.[4] However, the challenge here is that a lot of time is spent. According to Onu, a survey of over 300 Competitive Intelligence professionals shows that data collection is the most time-consuming task in typical Competitive Intelligence projects, amounting to more than 30% of the total time spent in the whole project. In this case, for Competitive Intelligence professionals to manually browse the Internet to read the information on every page of a Website in order to locate useful information and also to synthesize the information, it is mentally exhausting and overwhelming.

There is undeniably a huge demand for collecting data of interest from multiple Websites across the Web. For example, an online shopping system that collects multiple result records from different item sites, there is a need to determine whether any two items retrieved in the search result records refer to the same item. For a book online shopping system, the ISBN can be compared to achieve this. If ISBNs are not available, then their titles and authors could be used instead. Such a system is also expected to list down the price of an item from each site. Thus the system needs to know the semantic of each data unit. Unfortunately, the semantic labels of data units are often not provide in the result pages. For instance, in Figure X, no semantic labels for the values of title, author, publisher, etc., are given. Having semantic labels for data units is not only important for the above record linkage task, but also for storing collected search result records into a database table (e.g., Deep web crawlers) for later analysis. Early applications require tremendous human efforts to annotate data units manually, which severely limit their scalability.

Different tools have been developed that help to search, gather, analyze, categorize, and visualize a large collection of web documents. One of such tools is one proposed by Mr. Emmanuel Onu in his paper “Proposal of a Tool to enhance Competitive Intelligence on the Web”, and the tool is called the CI Web Snooper. It is from this tool that this paper is based. This research is a continuation of the initial work from the above mentioned paper.

The CI Web Snooper is a tool for searching and retrieving Websites from the Internet that can be used for information gathering and knowledge extraction. It uses a real-time search technique so that the information it sources from the Web is update. It has four major components: User Interface, Thesaurus Model, Web Crawler and Indexer. The User Interface allows the user to

specify search query and also specify seed URLs for the Web Crawler to use in its search. The Thesaurus Model is used to model the domain of interest and is key to query reformulation and indexing of Web pages. The Web Crawler component is responsible for finding and downloading Web pages using Breadth-First search algorithm that starts from the URLs specified by the user. Figure 1 shows the structure of the CI Web Snooper and Figure 2 shows the results collected.

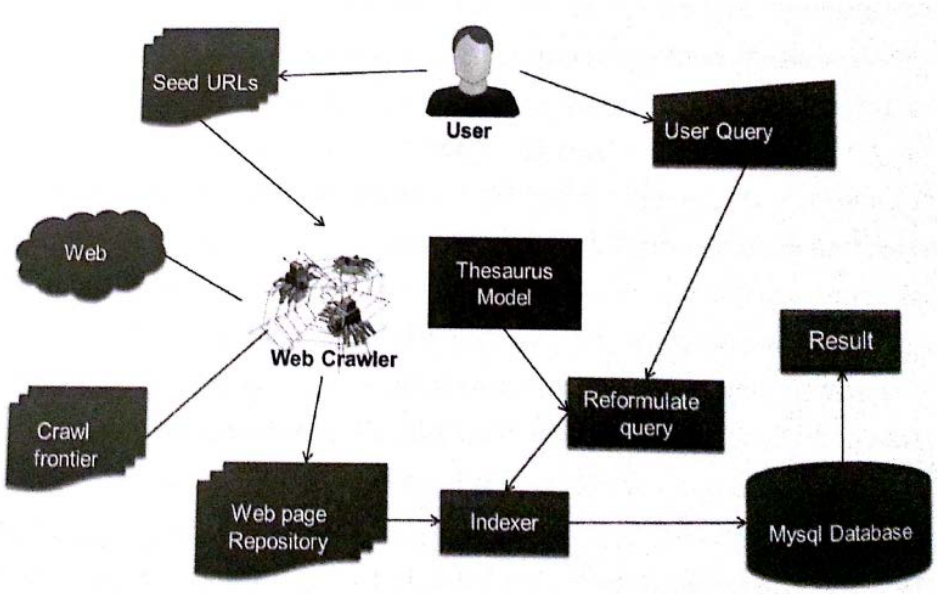


Figure 1: Structure of the CI Web Snooper

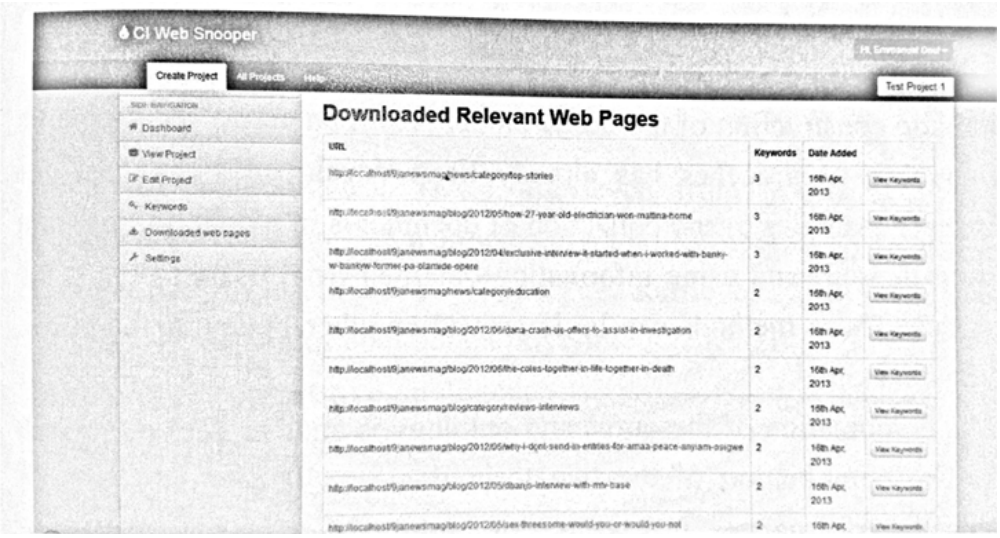


Figure 2: Downloaded web pages

The main objective of this research is to automatically annotate a collection of websites that have been retrieved from the Internet to be used for a domain specific activity. Thus a platform for

restructuring retrieved websites from a search query and automatically annotating the retrieved pages will be presented. This paper is organized into the following sections: Section 2 contains a related work and discussion. Section 3 covers detailed explanation of the approach used in developing a platform for restructuring and annotation. Section 4 covers the implementation, behavior and results from using the platform. Section 5 covers the conclusion.

2 State of the art

2.1 Related Work

Web information extraction and annotation is one of the areas under active research over the past years. A lot of work has been done on Web content annotation both manually and automatically. Many tools in intelligence gathering are available that use annotation. Olusoji et al. proposes the use of annotation among Economic Intelligence Actors in a collaborative environment for dynamic knowledge capitalization.[5] He states that in an organization, decision problems can be solved by Economic Intelligence approach which involves interactions among various actors called EI actors. These actors collaborate to ensure overall success of the decision problem solving process and they express knowledge which could be used for future use. An annotation model is thus employed for knowledge elicitation among EI actors. In this approach, users are able to add annotations on web documents in synchronous (real time) – whereby participating users are all online and asynchronous – users need not be online. Clearly, user or manual annotated systems are time consuming and not scalable but they achieve high rate of accuracy.

Mukherjee et al. proposed spatial locality and presentations to be used in annotations.[6] The approach to a web document structural analysis taken here was based to the simple observation that semantically related items in an HTML page normally exhibit consistency in presentation style and spatial locality. This applies particularly to content-rich Web sites that update frequently such as e-commerce sites since such sites are typically maintained using content management software that create HTML pages by populating templates with data from backend databases. This solution is however not applicable in document classification scenarios since this solution involves identifying multiple concepts within one document unlike classifying different documents with same class.

Some prior works have focused on automatic generation of wrappers. Wrappers are small applications or scripts capable of extracting information from web documents. However, those wrappers could only be used to extract data but not for annotations.

Lu et al. presented an automatic annotation approach that first aligns the data units on the result page into different groups such that the data in the same group have the same semantic.[7] Each

group was then annotated from different aspects and different annotations were aggregated to predict the final label for the group. An annotation wrapper for the search site is automatically constructed and used to annotate new pages from the same web database. Sriramoju followed Lu's approach to develop an application for automatically annotating search results. This approach has three phases, first is the alignment phase where data units are organized into different groups based on concepts. The second phase is the annotation phase. This takes care or making annotators that annotate web documents automatically. The third phase is the annotation wrapper generation where an annotation rule is generated for each defined concept.

Some tools have been developed that help to annotate Web documents. Such tools are implemented differently and also behave differently. Some of the common Web annotation tools are *Annotea* and *CREAM.*

Annotea

Annotea is a W3C project, which specifies infrastructure for annotation of Web documents, with emphasis on the collaborative use of annotations. The use of open standards is a very important principle for all the work of W3C to promote interoperability and extensibility. The main format for Annotea is RDF and the kinds of documents that can be annotated are limited to HTML or XML-based documents. This is restrictive for knowledge management, as much commercial data is in other formats. However, it provides in XPointer a method for locating annotations within a document. XPointer is a W3C recommendation for identifying fragments of URI resources. So long as the component of a document to which an XPointer refers is retained, the location of the associated annotation will be robust to changes in the detail of the document, but if large-scale revisions are made, annotations can easily come adrift from their anchor points.
The Annotea approach concentrates on a semi-formal style of annotation, in which annotations are free text statements about documents. These statements must have metadata (author, creation time, etc.) and may be typed according to user-defined RDF schemata of arbitrary complexity. In this respect, Annotea is not quite as formal as would be ideal for the creation of intelligent documents. The storage model proposed is a mixed one with annotations being stored as RDF held either on local machines or on public RDF servers. The Annotea framework has been instantiated in a number of tools including Amaya, Annozilla and Vannotea.

CREAM

The CREAM framework looks at the context in which annotations could be made and used as well as the format of the annotations themselves. It specifies components required by an annotation system including the annotation interface, with automatic support for annotators, document management system and annotation inference server. Like Annotea, CREAM subscribes to W3C standard formats with annotations made in RDF or OWL and XPointers used to locate annotations in text, which restricts it to web-native formats such as XML and HTML. Unlike Annotea, the authors of CREAM have considered the possibility of annotating the deep

web. This involves annotating the databases from which deep web pages are generated so that the annotations are generated automatically with the pages. As databases hold much of the legacy data in companies, this is a substantial addition. It is supported by a storage model that allows users to choose whether they want to store annotations separately on a server or embedded in a web page. This assumes more user control of the document and recognizes that users may prefer to store annotations with the source material. The CREAM framework allows for relational metadata, defined as "annotations which contain relationship instances". Relational metadata is essential for constructing knowledge bases which can be used to provide semantic services. Examples of tools based on the CREAM framework are S-CREAM and MOntoMat-Annotizer.

2.2 Annotation

Annotation can be defined as creating semantic labels within Web documents for the Semantic Web. Manual annotation is the transformation of existing syntactic resources into interlinked knowledge structures that represent relevant underlying information. [8]

The concept of annotation has been around for as long as printed media and is paramount to adding extra information to a document. The definition for annotation depends on context and usage. Olusoji et al. specify that annotation may be implicit or explicit. In explicit annotation, the meaning of the annotation made is assumed to be understood at least by a community of users – users belonging to the same field of study. While with implicit annotation, the meaning of implicit annotation may be known only to the annotator or the person who makes the annotation. For manual documents, text annotation is the practice of adding extra notes to text. It may also involve underlining, highlighting, comments and footnotes. Text annotations can include notes written for a reader's private purposes as well as shared annotations written for the purposes of collaborative writing and editing, commentary, or social reading and sharing.

For web documents, a Web annotation is an online annotation associated with a web resource, typically a web page. Using Web annotation system will allow a user to add, modify or remove information from a Web resource without modifying the resource itself. The annotation can be thought of as a layer on top of existing resource, and this annotation layer is usually visible to other users who share the same annotation system. Web annotations can be used for many purposes some of which are: to rate a Web resource, such as by its usefulness, user-friendliness, suitability for viewing by minors, how many users have accessed the resource; to improve or adapt its contents by adding or removing material; as a collaborative tool, e.g. to discuss contents of a certain resource; as a medium of social criticism, allowing Web users to reinterpret, enrich or protest against ideas that appear on the web.

Annotations in general can be roughly divided into three main elements: a body, an anchor and a marker. The body of an annotation includes reader generated content in form of symbols and text

such as hand written commentary or stars on the margin. The anchor is what indicates the actual original text to which the body of the annotation refers. Varying amounts of text can be anchored ranging from narrow sections like specific letter, word or phrase to a paragraph or whole document. The marker is the visual appearance of the anchor such as the yellow highlight or grey underline.

There are many different ways of annotating documents. Information Technology (IT) based tools define different ways of how to annotate Web documents depending on the need to use the tool. Likewise, Web documents have also their own means of annotation. Some of the common ways are: Footnote or Endnote – interfaces that display annotations below the corresponding text; Aligned Annotations – that display comments and notes horizontally in the text margins or columns; Interlinear Annotations – that attach annotations directly into text; Sticky Note interfaces; Voice Annotations – in which reviewers record annotations and embed them directly within the document; Pen or Digital Ink based interfaces that allow writing directly on the document on the screen.

2.3 Benefits of annotation

Annotation plays an important role in improving the way content is understood and how knowledge is transferred. Semantic annotation formally identifies concepts and relations between concepts in documents, and is intended primarily for use by machines. For example, a semantic annotation might relate “Paris” in a text to an ontology which both identifies it as the abstract concept “City” and links it to the instance “France” of the abstract concept “Country”, thus removing any ambiguity about which “Paris” it refers to.

Ontology-based semantic annotations also allow us to resolve anomalies in searches, e.g. if a document collection were annotated using a geographical ontology, it would become easy to distinguish “Niger” the country from “Niger” the river in searches, because they would be annotated with references to different concepts in the ontology.

Annotations can be used to provide automated services. For example, they can be processed using natural language generation software to automatically draft textual reports about the patient, the diagnostic information that is available and assessments made about the data by the medical team, a task which usually consumes doctors’ valuable time

Annotating helps one to understand: In order to condense a concept and put it down in your own words, you must first understand it. It also helps one to remember some concepts easily. Something that is well annotated or for example a piece of text that is well labeled, or an image that has several descriptors like location, date and other tags, it is easier to remember such as compared to those that don’t have any annotation attached to it.

2.4 Annotation Phases

For a set of Search Result Records (SRR) the automatic annotation approach has three major phases and there are *alignment phase*, *annotation phase* and *the wrapper generation phase* as illustrated in the figure below.

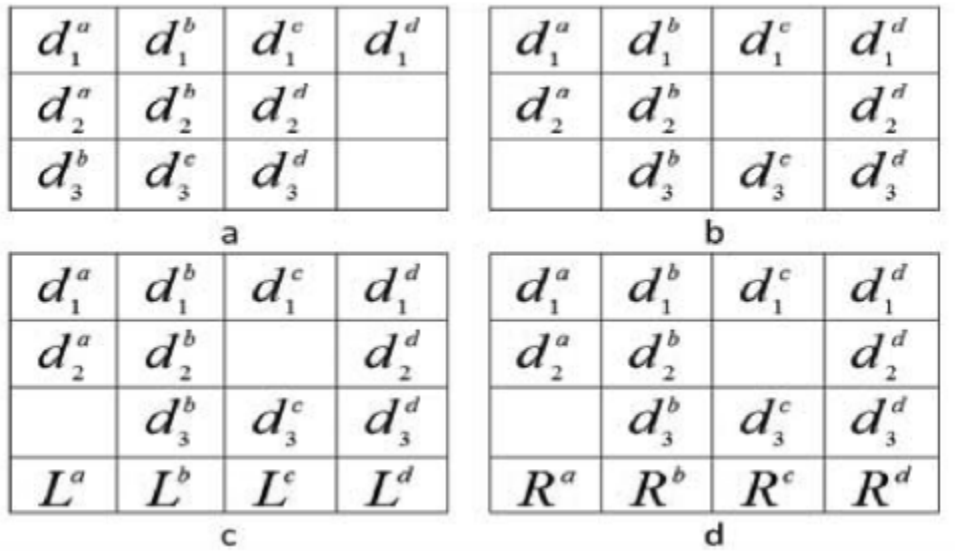


Figure 3: Search Result Records phases

The Alignment Phase

In alignment phase, the data units are identified in the Search Result Records and are organized into different groups with each group corresponding to a different concept (e.g. titles of books are grouped together). Figure 2b above shows the result of this phase having each column containing data units of the same concept across all Search Result Records. A data unit is a piece of text that semantically represents one concept of an entity. It corresponds to value of a record under an attribute. Figure below has Search Result Records with several data units e.g., the first book record has data units “Talking back to the machine: Computers and Human Aspiration”, “Peter J. Denning”, etc. Having data of the same semantic grouped together helps to identify the common patterns and features among the data units. These common features form the basis of annotation.

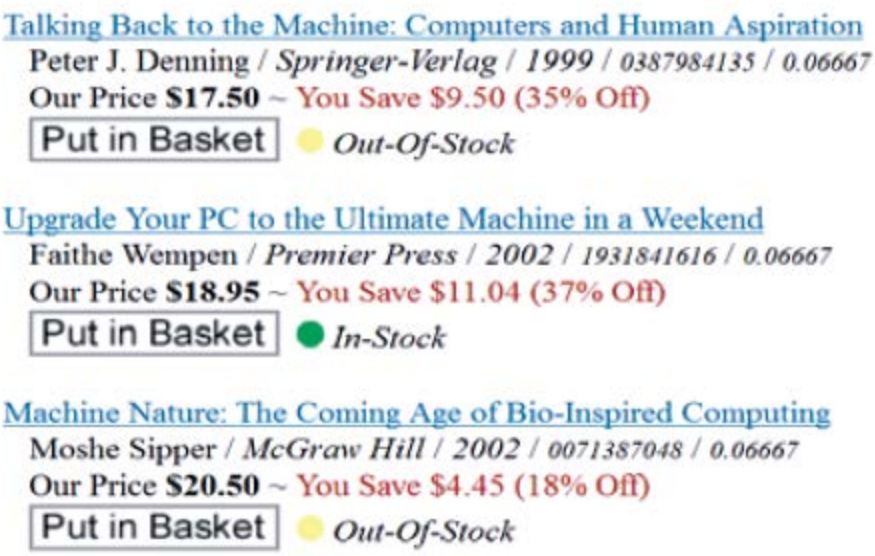


Figure 4: Example of SRR

Annotation Phase

In annotation phase several basic annotators are used with each exploiting one type of features. Every annotator is used to predict a label for the data units within the organized groups. A probability model is used to determine the most appropriate label. Figure 2c shows the result of the annotation phase where every group assigned with a semantic label L_j.

The Annotation Wrapper Generation

In this phase, for each identified concept, a rule R^j that describes how to extract the data units of this concept in the result page is generated and an appropriate semantic label is identified. A wrapper is used to annotate the data units retrieved from same web database for new queries and thus performs annotation quickly. Figure 2d shows a rule R^j for search results on a page.

2.5 Data Units and Text Node

In order to achieve accurate annotation, data alignment is an important step. Most existing data alignment techniques are based on one or few features but the most commonly used feature is the HTML tags. The working assumption is that the sub-trees corresponding to two data units in different search result records but with the same concept usually have the same structure. Each search result record has a tag structure that determines how the contents of the search result records are displayed on the web browser. Each node in such a tag structure is either a tag node or a text node. A tag node corresponds to an HTML tag surrounded by “<” and “>” in HTML

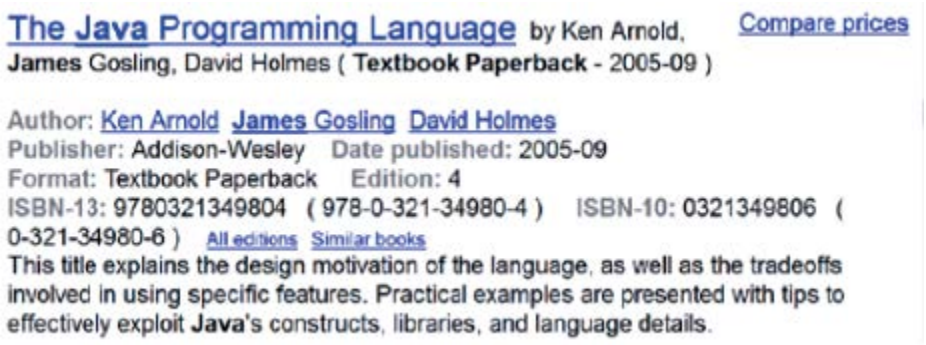
source and the text node is the text outside the "<" and ">". Text nodes are visible elements on the webpage and data units are located in the text nodes.

A text node may contain a varying amount of data units. Different text nodes hold different amounts of data units and there are four major relationships between data unit and a text node

- *One – to – One relationship*
  In this type of relationship, each text node contains exactly one data unit or in other words, the text belonging to a node contains the value of a single attribute. For example, there can be a text surrounded by a pair of <a> and tags like a title of a book in figure Y above. Such a text is a value for the title attribute. Such kinds of nodes are referred to as atomic text nodes. Thus an atomic text node is equivalent to a data unit.

- *One-to-Many relationship*
  In this kind of relationship, multiple data units are wrapped in one text node. Data units with different semantics are grouped into one text node. A person's name can be followed by his/her age then followed by salary amount, then followed by his city. In this case, name, age, salary and city fall under different concepts but they describe same person. In such a case we have a One-to-Many relationship. In figure x above line two or a search record result number 3 has "/MacGraw Hill/2002/0071387048/ 0.0667" is a single text node. It consists of four semantic data units like Publisher, Publication Date, ISBN and Relevance Score. It can be observed that if data units of attributes x1, x2, … xn in one search result record are grouped into a composite text node, then it is mostly true that data units of the same attributes in the next search result records are encoded as composite text nodes and that the data units follow the same order. So, there is need to separate such type of nodes into atomic data units in order to annotate them.

- *Many-to-One relationship*
  In this particular relationship, multiple text nodes together form a data unit. The figure below illustrates this case.


The Java Programming Language by Ken Arnold, James Gosling, David Holmes (Textbook Paperback - 2005-09) Compare prices

Author: Ken Arnold James Gosling David Holmes
Publisher: Addison-Wesley Date published: 2005-09
Format: Textbook Paperback Edition: 4
ISBN-13: 9780321349804 (978-0-321-34980-4) ISBN-10: 0321349806 (0-321-34980-6) All editions Similar books
This title explains the design motivation of the language, as well as the tradeoffs involved in using specific features. Practical examples are presented with tips to effectively exploit Java's constructs, libraries, and language details.

Figure 5: Relationships

The value of Author attribute is contained in multiple text nodes with each embedded inside a separate pair of <a> and HTML tags. Also, the "Java" term in the title is bold and surrounded by HTML and tags. Thus the title is split into three text nodes. For the purpose of annotation and extraction, there is need to remove such type of tags which are also referred to as "decorative tags".

- *One-to-Nothing relationship*
  In this relationship, text nodes are not part of any data in search result record. Like in the figure y above, text like Author and Publisher they are not part of the search result record and are not data units but semantic labels describing the meaning of the corresponding data units.

2.6 Data Units and Text Node Features

There are five major features shared by data units belonging to the same concept in all search result records and they can automatically be obtained.

- ***Data Content***
  Data units and text nodes that have same concept often share certain keywords. This fact can be explained in two ways. First, the data units corresponding to the search field where the user enters a search condition usually contain the search keywords. It can be seen from figure YY above that all the search results contain the text from search field which is "machine". We can see that all the titles have this keyword. Also, certain data units are designed with a leading label to make it easier for users to understand what the corresponding value is. In such cases, text nodes that contain data units of the same concept usually have same leading label. For example, in Fig. 1, the price of every book has leading words "Our Price" in the same text node.

- ***Presentation Style***
  This feature describes how the data units are displayed in the web browser. Some of its descriptive features are *font face, font size, font color, font weight, text decoration and italics.* Text decoration includes features like whether text is underlined or drop shadow etc. Data units expressing the dame concept usually have the same style.

- ***Data type***
  Although each data unit is expressed like a text string in HTML code, each has its own semantic type. Basic types include the following: *Date, Time, Currency, Integer, Decimal, Percentage, Symbol,* and *String*. String type is further defined in All-Capitalized-String, First- Letter-Capitalized-String, and Ordinary String. The data type of a composite text node is the concatenation of the data types of all its data units. For

example, the data type of the text node “Premier Press/2002/1931841616/0.06667” in Fig. 1 is <First-Letter-Capitalized-String> <Symbol> <Integer> <Symbol> <Integer> <Symbol> <Decimal>. Data units falling under the same concept or having text nodes involving same set of concepts usually have the same data type.

- ***Tag Path***
  A tag path of a text node is a sequence of tags traversing from the root of the SRR to the corresponding node in the tag tree.

- ***Adjacency***
  Adjacency refers to the data units that are immediately before and after in the SRR. For a given data unit d in an SRR, let d_p and d_s denote the data units immediately before and after d in the SRR, respectively. We refer d_p and d_s as the preceding and succeeding data units of d, respectively. Consider two data units d_1 and d_2 from two separate SRRs. It can be observed that if d_p^1 and d_p^2 belong to the same concept and/or d_s^1 and d_s^2 belong to the same concept, then it is more likely that d_1 and d_2 also belong to the same concept.

2.7 Data Annotation

The data annotation is based on the concept that the data units corresponding to the same attribute always share some common features. These common features are the basis of annotators. There are multiple basic annotators, and each annotator is used for identifying a specific feature. Every basic annotator is used to produce a label for the units within their group. Lu et al. identifies basic annotators used for annotating the database namely Table annotator, Query- based annotator, Schema annotator, Frequency- based annotator, Prefix/Suffix annotator and Common annotator.

- ***Table Annotator***
  Most websites with database behind them providing data often use a table to organize search record results. The table is divided into rows and columns and each row represents a search result record. Each column usually contains a header at the top that indicates the meaning of that column. Data units of the same concepts are well aligned with its corresponding column header. This special feature of the table layout can be utilized to annotate search result records.

Manufacture	Model	Class	Year	City	State	Price
HONDA	accord LX	4 DOOR	1998	playa del rey	CA	$11,500
HONDA	ACCORD LX	4 DOOR	1994	Spokane	WA	$ 7,500
HONDA	Accord Lx	4 DOOR	1997	Winona ake	ID	$ 8,700
HONDA	Accord LX	4 DOOR	1994	Cave Creek	AZ	$ 5,999
HONDA	Accord	4 DOOR	1999	Pomona	CA	$17,500

Figure 6. Search Record Results in table format

Since the physical position information of each data unit is obtained during search result records extraction, this information can be utilized to associate each data unit with its corresponding header.

- ***Query Based Annotator***
  The basic idea of this annotator is that the returned SRRs from a WDB are always related to the specified query. Specifically, the query terms entered in the search attributes on the local search interface of the WDB will most likely appear in some retrieved SRRs. For example, in Fig. 1, query term "machine" is submitted through the Title field on the search interface of the WDB and all three titles of the returned SRRs contain this query term. Thus, we can use the name of search field Title to annotate the title values of these SRRs. In general, query terms against an attribute may be entered to a textbox or chosen from a selection list on the local search interface.

- ***Schema Value Annotator***
  Many attributes on a search interface have predefined values on the interface. For example, the attribute *Authors* may have a set of predefined values which are names of authors in its selection list.

- ***Frequency Based Annotator***
  The data units with the higher frequency are likely to be attribute names, as part of the template program for generating records, while the data units with the lower frequency most probably come from databases as embedded values. In Fig. 1, "Our Price" appears in the three records and the followed price values are all different in these records. In other words, the adjacent units have different occurrence frequencies. Following this argument, "Our Price" can be recognized as the label of the value immediately following it.

- ***In-Text Prefix / Suffix Annotator***
  In some cases, a piece of data is encoded with its label to form a single unit without any obvious separator between the label and the value, but it contains both the label and the value. Such nodes may occur in all or multiple SRRs. After data alignment, all such nodes would be aligned together to form a group. The in-text prefix/suffix annotator

checks whether all data units in the aligned group share the same prefix or suffix. If the same prefix is confirmed and it is not a delimiter, then it is removed from all the data units in the group and is used as the label to annotate values following it. If the same suffix is identified and if the number of data units having the same suffix match the number of data units inside the next group, the suffix is used to annotate the data units inside the next group.

- ***Common Knowledge Annotator***
  Some data units on the result page are self-explanatory because of the common knowledge shared by human beings. For example, “in stock” and “out of stock” occur in many SRRs from e-commerce sites. Human users understand that it is about the availability of the product because this is common knowledge. So our common knowledge annotator tries to exploit this situation by using some predefined common concepts. Each common concept contains a label and a set of patterns or values. For example, a country concept has a label “country” and a set of values such as “U.S.A.,” “Canada,” and so on. Given a group of data units from the alignment step, if all the data units match the pattern or value of a concept, the label of this concept is assigned to the data units of this group.

2.8 Annotation Process

Web pages collected by the Web crawler are parsed, and the links and text content are extracted from the page. The links in form of URLs are passed to the frontier where there are queued up for being processed by the Web crawler. The extracted text is stored in the database table as keywords after removing stop words. Stop words are words that are too short and common like pronouns. Example would be pronouns *like he, they, her* etc. this process leaves keywords which are stored in a MySQL database table. So when a user performs a query, the search terms are searched against stored keywords in the database table. When relevant documents are found, key information about the documents is retrieved from the main page and is passed for annotation. The figure below shows a summary of the steps.

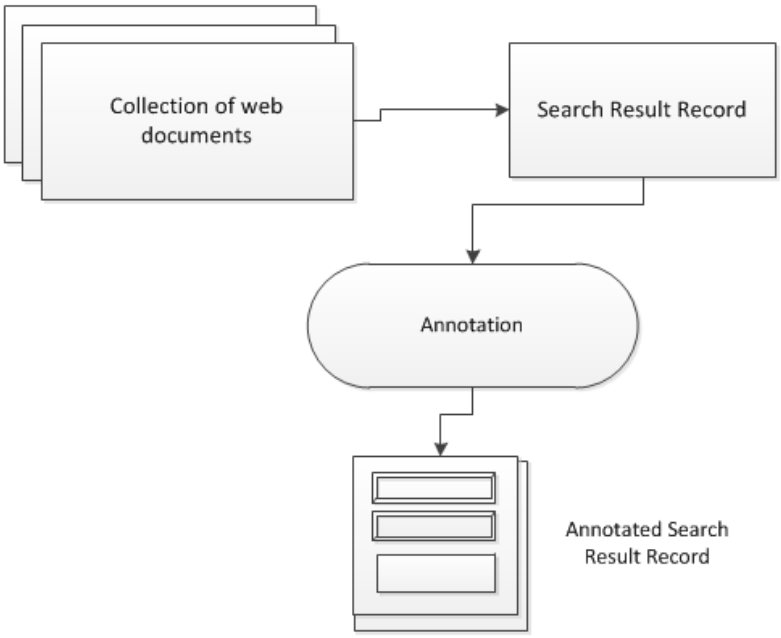


Figure 7: Annotation process

The annotations are made on objects of the search result page. The data objects that are annotated are the *Relevance, Title, Summary, URL and Page Size.* A search result record matching a query is given as an input to the annotation process and the result is an annotated record. When a user performs a search, the search terms are marched with the keywords stored in the MySQL database by the indexer. In the process, if there is a match relevant to the supplied query, a collection of Web pages are identified and collected as search results to the query. Necessary information that needs to be displayed to the user in form of search results is identified from each individual page. This information represents a Web page and will be annotated.

An annotation is done on an entry in the search result record. This can be denoted E_i, that is to say each amongst *Relevance, Title, Summary, URL and Page Size* can be regarded as an entry. An annotation on entry E_i can be denoted as A_i. Since these data objects are similar on each page, the annotation process is the same for each search result record and these annotations are automatically added to the search results. An annotation process for the search result record can be defined as:

$$A_{process} \rightarrow (A_i, E_i, SRR)$$

Where:

A_i is the annotation on an entry in the search result record
E_i is an *i* entry in the search record result
SRR is a search record result.

That is to say, an annotation process involves adding an annotation A_i on an entry E_i on a search result record SRR which is supplied as an input to the annotation process.
The output of the annotation process is an annotated search result record that can be displayed to a user.

3 Proposal

There are so many tools available for Information Retrieval. Some of them involve searching and query input as a starting point in retrieving required documents. Also, there are many tools available for annotating Web documents. Some of these tools are free while some are not. Most of the available annotating tools are manual. That is, some tools require that they be installed as a tool bar in a web browser so that a user can annotate a page by clicking on the tool from the tool bar. Most of such systems are manual in that they require a user to enter text or whatever content as annotation to that particular page. Some of such tools have a database system whereby all annotations for a particular page are stored. If a user browsing the Internet has that such annotation tool installed in his/her toolbar, he/she will be able to see all the annotations done by other users for that particular page. In this case, added knowledge through annotations on the page is transferred through many users.

Some annotation tools available are automatic. Different schemes are used to come about with tools that will automatically annotate content from different Web databases. In this paper, a tool is implemented that annotates search record results from different websites. It also classifies the search record results into different given categories. This implementation is done with respect to the domain of public health. The focus is to search information from news websites dealing with public health news. The collected results are then refined further by being classified into different categories. E.g. news articles containing information about Vaccine will fall into the category of Vaccine. Likewise any article that has information about some disease will fall under the disease category. This will help the searcher in reducing the number of pages he / she has to browse in order to get to the required result.

The implementation consists of a web crawler that crawls to gather websites in a specific domain. It also has a search engine that searches the available information from the websites that have been crawled on. The search engine requires a search term or query in form of text to be submitted via a form and results matching to that query are displayed. When a collection of search results from a particular query is being displayed, the search record results are automatically classified into different categories based on the theme of the content in the Web document. This implementation is done in PHP language and has a MySQL database backend.

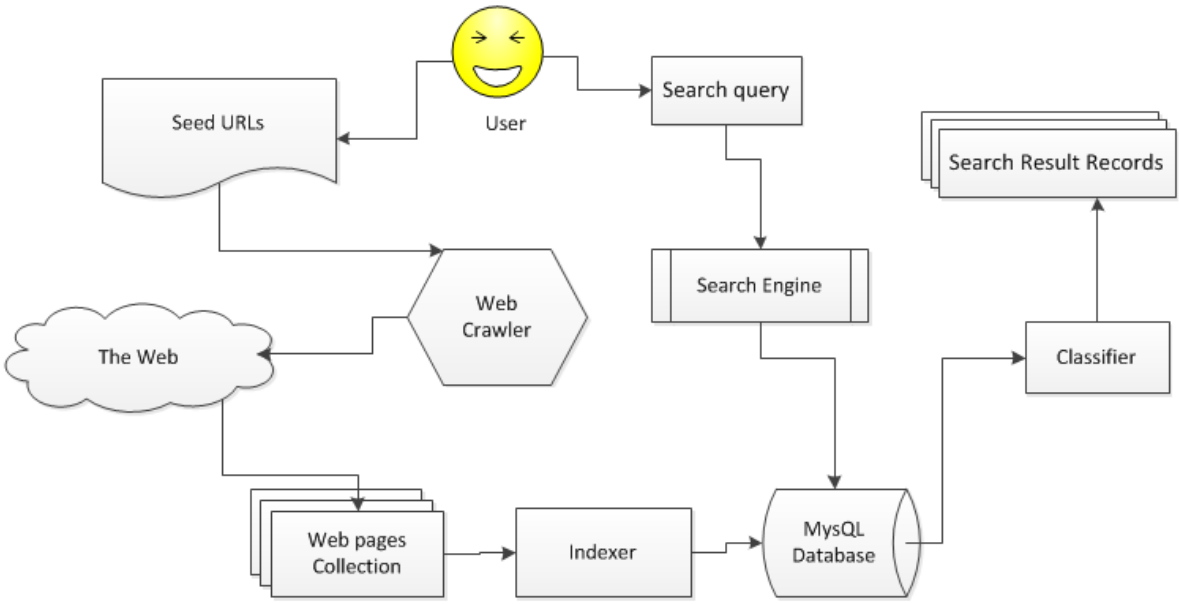


Figure 8: Architectural Design

3.1 Components of the System

Web Crawler

Web crawler is a tool or software that has the task of traversing the Web and collecting documents that it finds. It uses a Breadth-First search algorithm that starts from the seed URLs specified by the user. Depending on the settings, a Web crawler follows all links in the collected documents up until the stopping point is met or it can be made to be crawling forever. It is also known as a Web spider. A Web crawler starts with a list of URLs to visit, called the *seeds*. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the *crawl frontier*. URLs from the frontier are recursively visited according to a set of policies. If the crawler is performing archiving of websites it copies and saves the information as it goes. Such archives are usually stored such that they can be viewed, read and navigated as they were on the live web, but are preserved as snapshots.

In this implementation, a PHP tool called PHP Sphider is used for a Web crawler [9]. Sphider is implemented in PHP language and also uses some JavaScript. It uses a MySQL database for storing indexed keywords. It has several functionalities some of which are stemming: that is the process of reducing inflected or derived words to their stem or root form. For example, a stemming algorithm reduces words *fishing, fishery, fished* to their root word *fish.*

Conceptually, the algorithm executed by a web crawler is extremely simple: select a URL from a set of candidates, download the associated web pages, extract the URLs (hyperlinks) contained therein, and add those URLs that have not been encountered before to the candidate set. Different parts are used by a standard web crawler in its crawling process. The figure below illustrates some components.

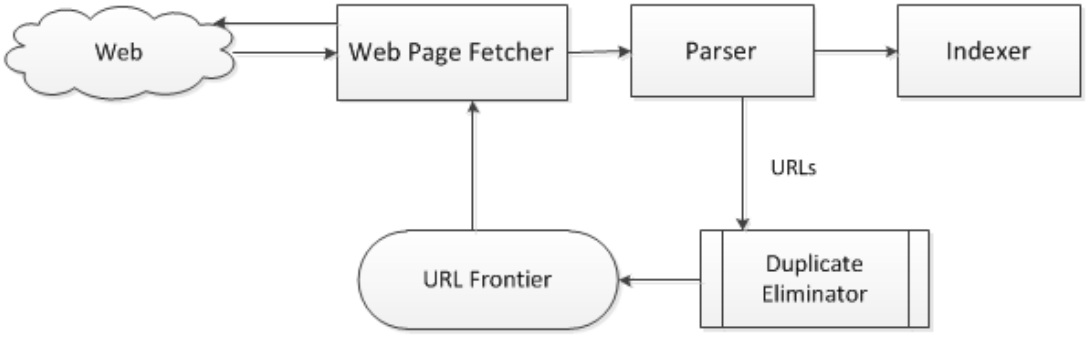


Figure 9: A Web crawler architecture

The URL frontier stores information about all links that are yet to be visited by the Web crawler. These links originate from the Seed URLs that a user sets as the starting point for the Web crawler to start fetching Web documents. With the help of the Web Page Fetcher, all found documents that the Web crawler is allowed to fetch are collected via the HTTP. The Parser is used to get links and text from the Web pages collected. The text extracted from Web documents is passed to the indexer for indexing while URLs are passed to the Duplicate Eliminator, which is also called the Normalizer. The aim is to make sure that no multiple URLs collected referrer to the same Web document.

The Indexer module used to index text extracted from Web pages. The text is stored in database and the Indexer uses the power of regular expressions to do word comparison. Text from search query is used and compared with existing keywords taken from Web pages. Indexing helps the search engines to determine the relevancy of the retrieved documents in relation to the supplied query. Different algorithms of varying complexities are used to calculate the relevancy of the retrieved Web page. The pages which are calculated to have higher relevancy are displayed at the top or the search result page and ones with low relevancy fall below.

Search Engine

This is where most of the work is done. An interface is provided for entering a search term and once the search button is clicked, the search query is checked against already available content. In this implementation, documents are sourced and indexed by a Web crawler and the resulting

text is stored in a MySQL database. The search query is first sanitized to check if it is suitable to be used. For example, the engine tries to avoid a query composed of common words like ‘the’. Such words and also too short words are not needed to be used as search terms. Search terms containing characters like ‘a’ would be ignored as being too short.

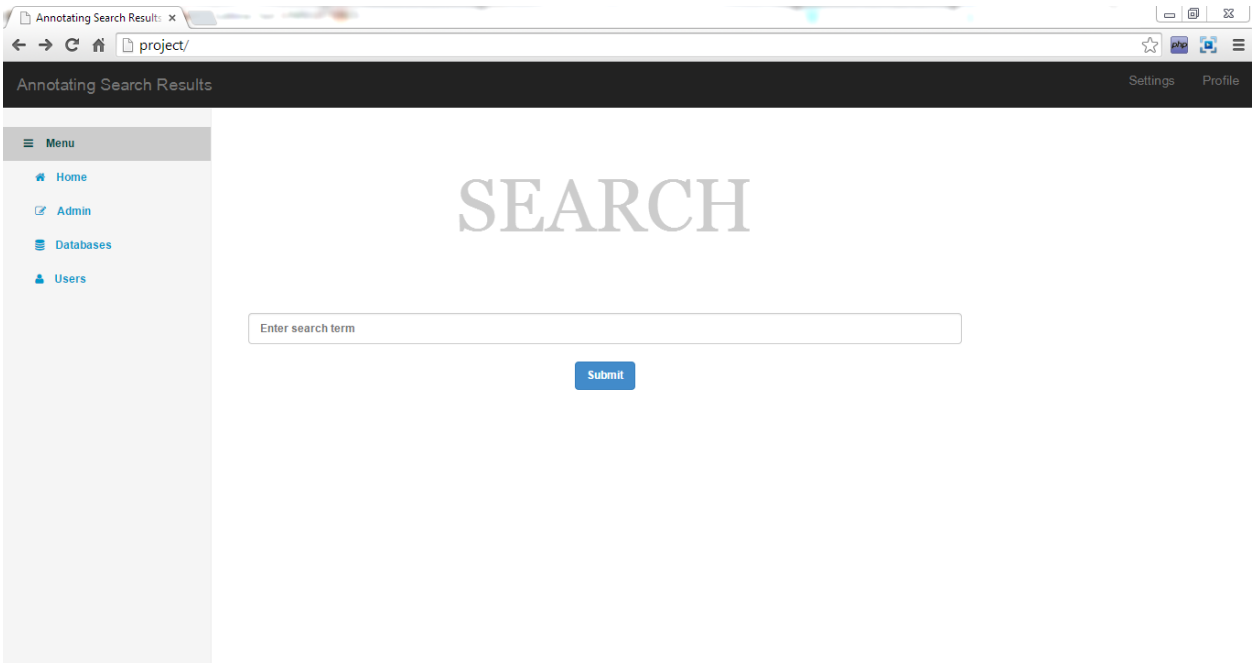


Figure 10: Search Page Interface

The purpose of this paper is to find ways of adding value to search results obtained either by querying search engines or by use of Web crawlers and also restructuring the data in the search result records for visualization. Some of the ways already discussed that add value to Web documents are annotating the search results as well as categorizing the search result records into different groups that have same semantic concepts. It is at the search engine level in this implementation that annotation is taking place. Table annotation already discussed above is used to annotate the search result records.

For a Web document, there are several features that can be used to annotate the document. The URL of a Website, title of the website, file type and many more can help to identify a Web page and assign an appropriate label.

3.2 Retrieving results

Results obtained from the database following a search query are displayed on the search result page in form of multiple records. The results follow a tabular format whereby each row represents a segment of a single record. In this implementation, each search result record consists of segments of data items that portray information about the actual Web page. Each search result record is indicated by a number. The figure below illustrates how search result records are displayed in a page and how they are structured.

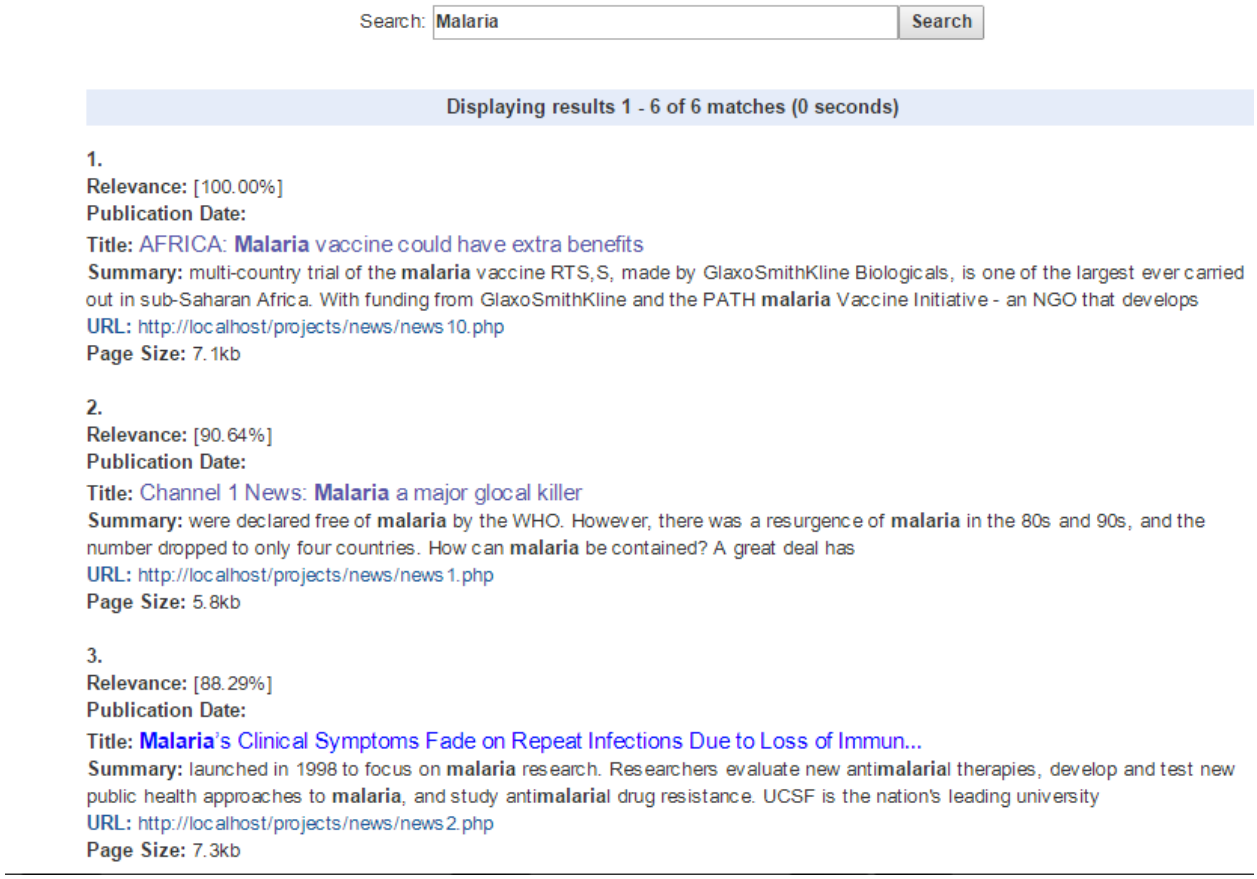


Figure 11: Structure of a search result

As can be seen from the picture above, each search result record is divided into various segments in the name of *Relevance*, *Publication Date*, *Title*, *Summary*, *URL* and *Page Size*. These date segments are some of the key information that gives a user of this tool an overall picture of what the main page contains, in terms of concepts as well as content, how relevant the page is to the given search query, the domain of the page etc. Each search result record is automatically annotated with the said labels. Each Web page whose record falls in the result set is dissected and relevant records that represent that Web page are automatically attached labels to denote that the following data item is about.

Annotating each data item with a label gives a user a hint about which the following data item is. This also makes it easier to save the records in a database in the case of some information retrieval tools since data items having same concept fall in the same category and are assigned same label. For example, in the following figure, Title *"Malaria's Clinical Symptoms Fade on Repeat Infections Due to Loss of Imm…"* and Title *"Milestone for child malaria vaccine"* are data items that fall under the same concept of being title.

Figure 12: Alignment of data items

3.3 Data units displayed in a search result record

Data items having the same concept are aligned in the similar position and given same label. This makes it easier to differentiate to data items belonging to different records but having the same concept. The following section briefly describes the data items appearing the search result.

- ***Relevance***

  In information retrieval it is important to know if the retrieved information meets the users' needs. This task of judging how well the retrieved information matches the user's query is called *Relevance.* Documents that have a higher relevance or are a close match to the user's query are placed higher in the result page. In this implementation, the search engine calculates the weight or relevance of the query by considering relative weight of a word in the title of a webpage, relative weight of a word in the domain name, relative weight of a word in the path name and relative weight of a word in meta keywords.

- ***Publication Date***

  Publication date is important as it helps the user decide where the article is outdated or is necessary to visit. A publication date plays major roles in search results. For example, in news event detection and tracking systems, the publication time of a news story is usually one of the most important indicators that help determine whether two news stories refer to the same event. As another example, in news meta-searching systems where the news articles retrieved from multiple sources need to be reorganized into a single ranked list, the publication time is a critical factor used to rank those news articles.

- ***Title***

  The title returned in the search result record gives the overall theme of the content of a Website. In most cases especially with news articles, the title gives of the Web page is usually the title of the news story featured. It is from this basis that title for a Web page in the search result page is being used to classify search result records into different categories based on their concepts.

- ***Summary***

  The summary in the search result records gives a user a review of how the whole content is like in the actual Web document by just displaying part of the text. It is important because it captures the keywords that the user used in the search query.

- ***URL***

  The URL (Universal Resource Locator) is what can be termed as the address for that page. Depending on the domain of the Web site, it can either be a top lever domain (TLD) or not. The URL can be used to detect the country of the news item since top level domains can have country extensions. For example, a website in Malawi can have a URL like www.abc.mw. The '.mw' indicates the country code for Malawi hence a quick glance at the URL can help tell some information.

- ***Page Size***

  The page size indicates the size of the page to be downloaded.

3.4 Classifying Search Result Records

One of the ways of adding value to the search results is categorizing the search results. Categorizing search results is important in so many ways to the user. It narrows down the users' choice to specific possible relevant results out of the many possible result records. When the results are categorized, a user has to select out of the provided categorized which significantly narrows down options. Searchers review search results to predict which web pages will be topical, authoritative and high quality. If the desired item is ranked far down the list, searchers are unlikely to find it, since they rarely look beyond the top 10-20 results. If, however, a search result record falls in a visible and meaningfully labeled category (perhaps as part of an overview), the searcher can navigate directly to the category and then to the desired item, rather than linearly scanning the entire list, which could involve requesting multiple additional pages from the server. Categorization allows users to see at a glance where their search term did not yield results, for example that there is no result for "cabinet making" in the "graduate" area of the University of Maryland hierarchy. This can help them avoid examining results that are not relevant to their information need. [10]

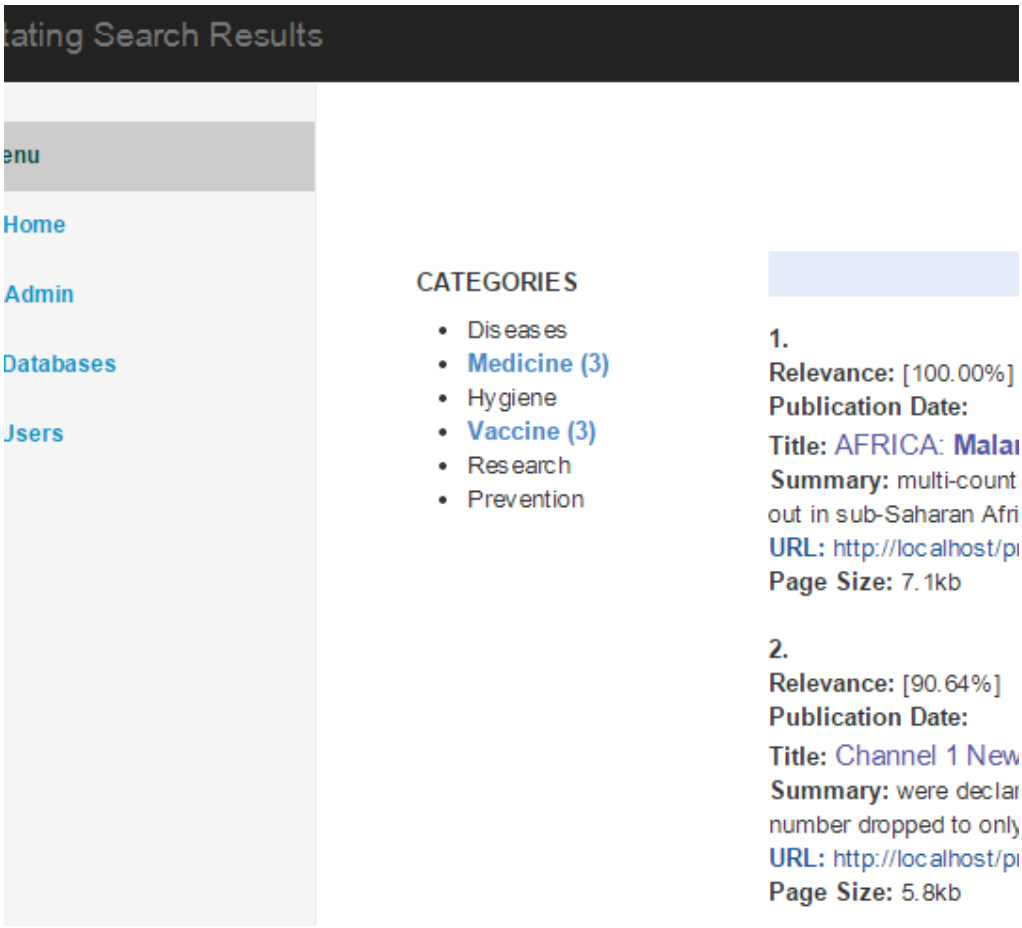


Figure 13: Categories in search results

In this implementation, the classification of various categories is done using a PHP library called PHP Classifier.11 It is developed in PHP and utilizes classes to tokenize strings and classify given input. This classification is makes use of already existing categories and utilizes machine learning techniques to be able to classify given content. In this case, a supervised model is used to classify the search result records. Supervised data analysis is used to estimate an unknown dependency from known input-output data. For example, input variables might include the quantities of different articles bought by a particular customer, the date they made the purchase, the location and the price they paid. Output variables might include an indication of whether the customer responds to a sales campaign or not. Output variables are also known as targets in data mining.

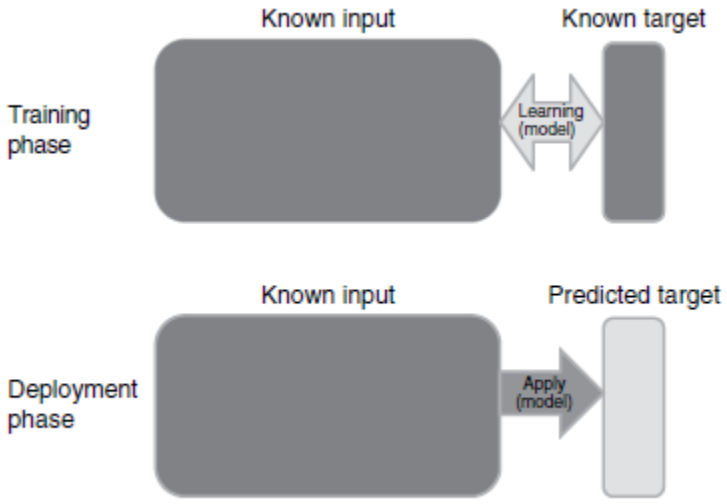


Figure 14: Supervised learning model

The categories for this particular implementation were selected topics in the domain of public health and these are *Diseases, Hygiene, Medicine, Vaccine, Research and Prevention.* The system is given training data with which it uses as an input in order to classify given text. The text used for determining the category into which a particular record appears is the *title* of that particular search result record. The choice of a title is because it is relatively shorter and gives an overall description of the content of the whole Web page. In most cases, titles contain key words that are used in classifying the result record.

The classification process uses the Naïve Bayes Algorithm. Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. The algorithm uses conditional probabilities in order to determine in which category an event or text falls. It assumes that words appear in a sentence or paragraph independent of each other. In other words, given a word in a sentence, the algorithm assumes that the word coming after it is independent of it. Conditional probability

states that given an event A, what would be the probability of event B happening. This notion can be applied to search results in that given some training data, what are the chances that given search result record falls in some category.

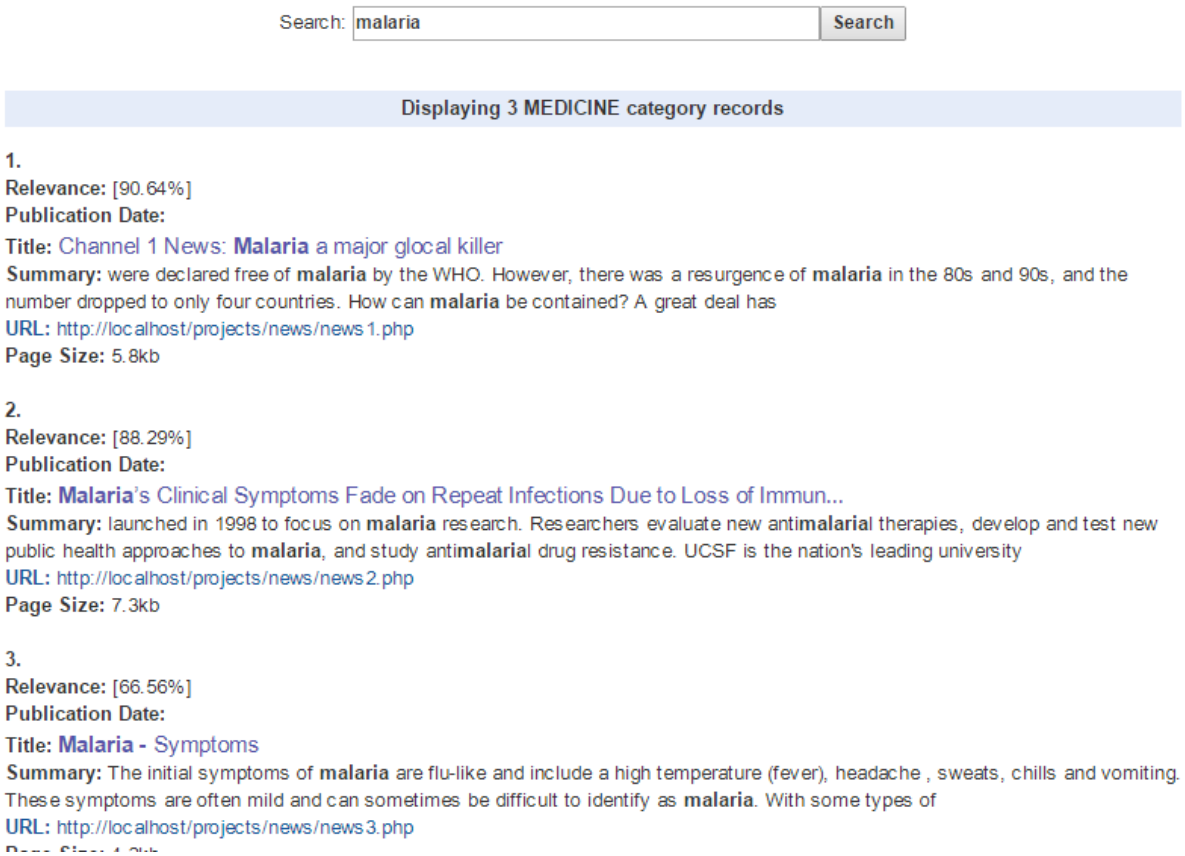


Figure 15: displaying results from a category

The figure above illustrates search result records that fall under a particular category after being classified using the Naïve Bayes classifier. The more sufficient training data given to the classifier, the more accurate it becomes.

4 Conclusion

In this paper, a discussion on how to add value to search results collected as a result of querying or searching the Web was made. This is a continuation to the work done by Mr. Emmanuel Onu in his research paper entitled “Proposal of a Tool to Enhance Competitive Intelligence on the Web”. In the said research, a tool was implemented that fetched a collection of websites based on an enhanced query. The query was reformulated using a thesaurus model and a web crawler was used to fetch websites relating to the query. The collected results were displayed as plain URLs of the actual Web pages. In this paper, further research was done on how to add value to the results brought about by querying the Internet. Two aspects were viewed: Annotating search results to add value to them and categorizing search result records for simplified identifying of relevant search result records. The annotation was done automatically by extracting contents of the requested Web page and retrieving specified data items of value to be displayed on the Web page. Annotation technologies were reviewed and Table Annotation was used to label data units retrieved following a search query.

An implementation of a search interface was made that consisted of a Web Crawler for fetching and indexing Web documents, a Search Engine for searching and displaying results, and a Classifier for categorizing search result records. The implementation was done in PHP with MySQL as the backend database. The results showed an improvement on earlier work, where by a tool for collecting data in competitive intelligence field can have search results well annotated and classified.

[1] Content available at: www.facebook.com

[2] Content available at: www.youtube.com

[3] Competitive intelligence is the action of defining, gathering, analyzing, and distributing intelligence about products, customers, competitors, and any aspect of the environment needed to support executives and managers making strategic decisions for an organization. Sourced at: en.wikipedia.org/wiki/Competitive_intelligence

[4] Onu E.I. (2013). Proposal of a Tool to Enhance Competitive Intelligence on the Web: CI Web Snooper. *Contextes, lagues et cultures dans l'organisation des Connaissances.* 115-134

[5] Olusoji Okunoye, Bolanle Oladejo, Victor Odumuyiwa. Dynamic Knowledge Capitalization through Annotation among Economic Intelligence Actors in a Collaborative Environment. Colloque International Veille Strat_egique Scienti_que et Technologique - VSST 2010, Oct 2010, Toulouse, France. pp.1-17. <inria-00546809>

[6] S. Mukherjee, I.V. Ramakrishnan, and A. Singh, "Bootstrapping Semantic Annotation for Content-Rich HTML Documents," Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2005

[7] Yiyao Lu, Hai He, Hongkun Zhao, Weiyi Meng and Clement Yu, (2013). Annotating Search Results from Web Databases. IEEE Transactions On Knowledge And Data Engineering, Vol. 25, NO. 3.p1-14.

[8] Uren V., Cimiano P., Iria P., Handschuh S., Vargas-Vera M., Motta E., Ciravegna F., (2005). Semantic annotation for knowledge management: Requirements and a survey of the state of the art. Journal of Web Semantics, WEBSEM-71. P1-15

[9] Content available at: www.sphider.eu. Accessed on 4th October, 2014.

[10] Kules B., Kustanowitz J. and Ben Shneiderman, (2006). Categorizing Web Search Results into Meaningful and Stable Categories Using Fast-Feature Techniques. Retrieved from: http://hcil2.cs.umd.edu/trs/2006-15/2006-15.pdf

[11] PHP Classifier is a library written in PHP that implements the Naïve Bayes Algorithm for classifying items into categories. Content available at: https://codeload.github.com/Dachande663/PHP-Classifier/zip/master.

References

Sriramoju1 S. B., (2014). An Application for Annotating Web Search Results. *International Journal of Innovative Research in Computer and Communication Engineering.* 2(3). 3306-3312.

Embley D.W., Campbell D.M., Jiang Y.S., Liddle S.W., Lonsdale D.W., Smith R.D., (1999). Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering (31).* 227-251

Jadhao1 S., Kulkarni R. P., (2014). Review of Semantic Web, Annotation Methods and Automatic Annotation for Web Search Results. *International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622. International Conference on Industrial Automation and Computing (ICIAC- 12-13th April 2014)*

Meagher P., (2004). Implement Bayesian inference using PHP, Part 1. Build intelligent Web applications through conditional probability. IBM Developer Works. Document available at: http://www.ibm.com/developerworks/library/wa-bayes1/wa-bayes1-pdf.pdf.

Handschuh S., Volz R., Staab S., (2004). Annotation for the Deep Web. *IEEE INTELLIGENT SYSTEMS*. Pp 43-48.

Handschuh S. and Staab S., (2002). "Authoring and Annotation of Web Pages in CREAM," *Proc. 11th Int'l World Wide Web Conf.*, ACM Press, pp. 462–473.

OKUNOYE O. B., DAVID A. & UWADIA C. *AMTEA: Tool for Creating and Exploiting Annotations in the Context of Economic Intelligence (Competitive Intelligence).* IEEE IRI 2010 Conference, Las Vegas, USA; August 4 – 6, 2010

J. Wang and F.H. Lochovsky, "Data Extraction and Label Assignment for Web Databases," Proc. 12th Int'l Conf. World Wide Web (WWW), 2003.

Y. Lu, H. He, H. Zhao, W. Meng, and C. Yu, "Annotating Structured Data of the Deep Web," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), 2007.

H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu, "Fully Automatic Wrapper Generation for Search Engines," Proc. Int'l Conf. World Wide Web (WWW), 2005.

L. Liu, C. Pu, and W. Han, "XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources," Proc. IEEE 16th Int'l Conf. Data Eng. (ICDE), 2001.

Z. Wu et al., "Towards Automatic Incorporation of Search Engines into a Large-Scale Metasearch Engine," Proc. IEEE/WIC Int'l Conf. Web Intelligence (WI '03), 2003.

S. Mukherjee, I.V. Ramakrishnan, and A. Singh), (2005). "Bootstrapping Semantic Annotation for Content-Rich HTML Documents," Proc. IEEE Int'l Conf. Data Eng. (ICDE.

Cardoso J, Sheth A.P., (2006). Semantic Web Services, Processes and Applications. Available at http://www.springer.com/978-0-387-30239-3

J. Kahan, M.-J. Koivunen, E. Prud'Hommeaux, R. Swick, Annotea: an open RDF infrastructure for shared web annotations, in: Proceedings of the 10th International World Wide Web Conference (WWW 2001), Hong Kong, 2001.

Bincy S Kalloor, Shiji C.G., (2014). A Survey on Data Annotation for Web Databases. *International Journal of Engineering and Innovative Technology (IJEIT). 4(3).* 133-135

Boraste P.P.,(2014). A Survey on Data Annotation for the Web Databases. *IOSR Journal of Computer Engineering (IOSR-JCE). 16(2).* 68-70