Crowd Aided Web Search: Concept and Implementation

Chun-Hsiung Tseng[1], Non-member

ABSTRACT

Although keyword-based search algorithms usually do their jobs well, they may sometimes yield weird results. Despite of the fact that the Web is the largest database, comparing to relational databases, the set of search operations for the Web is still primitive. This paper proposes two ways to remedy this: first, advanced information sources should be created. The role of advanced information sources of the Web is analogous to views of relational databases. Second, we propose several data processing tools based on the concept of advanced information sources. With these two mechanisms, the researcher tries to distinguish the data-centric view from the presentation view of the Web. In the paper, both the concept and an implementation are proposed.

Keywords: CrowdIntelligence, CrowdSources, Information Extractionn

1. INTRODUCTION

The objective of existing search engines is to help Web users locate their needed information accurately and efficiently. The Web is an extremely huge database flooded with information. To locate a specific piece of information in the Web is probably more difficult than to locate a needle in a haystack. Today, people usually rely on keyword-based search mechanisms. For keyword-based searches to perform well, both computation power of machines and carefully designed algorithms are needed to process such huge amount of information. For example, one of them utilize the combination of map-reduce, page rank, inverted-index, and cloud-based infrastructure and achieve pretty good results in most general cases.

However, keyword-based search algorithms may sometimes yield weird results despite the fact that they usually do their jobs well. Why? A possible cause is that modern Web search engines focus on syntactic part rather than semantic part of Web pages. They have no idea about the semantic and context of a submitted query. Generally speaking, what keyword-based search algorithms do is recording the relationships between keywords and documents and simply return lists of relevant documents when related keywords are submitted. In some circumstances, people are not satisfied with simple keyword-based searches, and some questions can only be answered by human beings. For example, the question "where is the best place to go on vacation", may be overly complex for most modern Web search engines but pretty simple for human beings. Although there are semantic Web related technologies, wide adoption is still lacked. Furthermore, instead of simple page lists, sometimes aggregated results are preferred. An example is: people may ask "what is the average housing price in a specific area?" Such questions are also difficult for ordinary Web search engines.

The above scenarios show that, in many circumstances, people expect more from search engines. Simple page lists, the most common type of results produced by modern search engines, are sometimes not sufficient. Considering the Web as a huge database, Web pages are actually raw data within the database. The straightforward simple page lists are just unprocessed data. Compared with query technologies for relational database, operators provided by modern Web search engines are very primitive. Powerful query operators can be demanded to solve certain problems. However, complex operators are double-edged swords since they can complicate the search operation. Simplicity is also an important factor of the wide adoption of Web search engine technologies. Is it possible to find a solution to enhance the functionality and still keep simplicity? A possible solution is to keep operators simple but allow data sources to contain richer information. An example is views of relational databases, which allow preprocessing (or pre-configuration) of the original data but provide the same operators as normal relational tables. With similar concepts, it appears reasonable to improve Web search results with advanced information sources providing simple operators, that is, we can keep keyword-based interfaces but provide mechanisms to construct powerful information sources.

Definitely, the approach has drawbacks. First, skills and efforts are needed to construct these powerful information sources. Second, it will be difficult to evaluate the quality of an information source. Third, the performance of Web searches can degrade due to the increased complexity. And last but not the least, frauds can happen since ordinary users may not be able to distinguish between original Web pages and

Manuscript received on May 6, 2014 ; revised on October 4, 2014.
Final manuscript received October 29, 2014.
[1] The author is with Department of Information Management, Nanhua University, Taiwan, E-mail: lendle_tseng@seed.net.tw

modified information sources. To remedy these, in this research, a crowd-based approach is proposed. We leverage the power of crowds in several aspects:

1. sometimes, it can be difficult for inexperienced Web users to identify valuable information from Web pages; with a good annotation mechanism, the intelligence of crowds can help point out valuable parts from Web pages and thus can help ordinary users to acquire the information
2. different users can take different views for even the same set of Web resources due to their different backgrounds; in many situations, these points of views can be so diverse and rich that they can be adopted to construct different information sources and other users can benefit from them
3. if used adequately, feedbacks generated from crowd intelligence can be good tools to against frauds

In this paper, at first, the concept of crowd aided Web search is presented. Required components are shown along with their definitions and usages. Furthermore, implementations of these components are also included. To prove the concept is practical, a working example is provided.

2. RELATED WORKS

Although Web search engines today are usually considered efficient, in some circumstances, they are not, especially when semantics and human intelligence are of concern. Some queries simply cannot be answered by machines alone. In such cases, human input is required [1]. The research field is typically named as crowd search or crowd searching. It is not an easy task to mediate between responses from human beings and search engines, and thus the research field is very challenging.

Crowd search is highly related with social networking [2]. The opinions collected within friends and expert/local communities can be ultimately helpful for the search task. For example, the question "find all images that satisfy a given set of properties" can be difficult for machines to proceed, but with the help of human intelligence, answering the question becomes simpler [3]. A special query interface that let users pose questions and explore results spanning over multiple sources was proposed in [4]. Another type of crowd search and crowd sourcing is social bookmarking. As shown in Heymanns research work [5], social bookmarking is a recent phenomenon which has the potential to give us a great deal of data about pages on the web.

Various crowd search and crowd sourcing systems have been proposed. For example, the research of Parameswaran proposed a human intelligence-based methodology for solving the human-assisted graph search problem [6]. Amazons Amazon Mechanical Turk is a commercial product that enables computer programmers (known as Requesters) to co-ordinate the use of human intelligence to perform tasks that computers are currently unable to do [7].

Considering the Web as a huge database with plenty of information, existing query techniques are far from perfect. Today, the most widely adopted query technique is the keyword-based search. The research of Konopnicki and Shmueli [8] examined some trends in the domain of search, namely the emergence of system-level search services and of the semantic web. In the research, a SQLlike solution was surveyed. In [4], a YQL (Yahoo! Query Language [9]) is implemented. The framework consists of some interaction primitives and is aimed at supporting users in finding responses to multi-domain queries.

In additional to advanced Web query languages, some semantic-based techniques have been proposed. Trillo proposed a set of semantics techniques to group the results provided by a traditional search engine into categories defined by the different meanings of the input keywords [10]. In the research, knowledge provided by ontologies available on the web was used to dynamically define the possible categories. To infer semantic from Web pages, a frequently used technique is annotations. Chun and Warner proposed semantic metadata and annotation of Deep Web Services (DWS), a reasoning component to assess the relevance of DWS for searching the Deep Web contents, using likelihood of occurrence of data sources that contain the query terms, and present a method of ranking the DWSs [11]. Sarkas, Paparizos, and Tsaparas propsed an unsupervised mechanism for assessing the likelihood of a structured annotation [12]. Sometimes, information sources with semantic information attached will be regarded to as structured information. The research of Paparizos presented a system called HELIX for such information sources [13].

3. CROWD AIDED WEB SEARCH

Recall how people look for answers before the Internet age. Besides searching in books, newspapers, and magazines, etc., they may consult other people for suggestions and solutions. Usually, the people who are consulted are considered as possessing more knowledge or experiences about the questions. Of course, the quality of such processed information can vary greatly, but acquiring information in this way is typically efficient if the information sources (the domain experts and the mediums to carry the information) are reliable. Today, the amount of information presented to users today is much more than the last decade. We have search engines, which are good at locating the (possibly) desired pieces of information among Web resources. Standing in front of the information flood, most Web users will at first look to search engines for help. Do search engines really remedy all the symptoms caused by the explosive amount of information?

Taking a deeper look into how search engines work,

one will find that the answer is probably no. By extracting keywords from Web resources, building indexes of keywords, and tracking links between Web resources, search engines are good at handling the syntactic part and they can locate Web resources containing the given keywords efficiently. However, search engines employing keyword-based approaches actually have no idea about the semantics of the knowledge they present. A common weakness of generic Web search engines is that, with no priori understanding of the materials they processed, they perform generic search within all Web resources. Hence, Web users without sufficient domain knowledge may not be able to leverage the full power of search engines when they need domain-specific information. For example, for users with little or no IT knowledge, distinguishing "apple" that can be eaten from the brand "apple" may sometime not be easy. In such cases, domain-specific searches, i.e., searches targeting information sources of a specific domain, will give better results than generic searches. Where do these domain-specific information sources come from? Who is responsible for constructing such information sources? Apparently, relying on few domain experts will not work because of the huge amount of existing Web resources. Utilizing crowd intelligence is a possible cure; however, there may not be sufficient tools for the adoption of crowd intelligence in the Web search field today.

Where do these domain-specific information sources come from? Who is responsible for constructing such information sources? Apparently, relying on few domain experts will not work because of the huge amount of existing Web resources. Utilizing crowd intelligence is a possible cure; however, there may not be sufficient tools for the adoption of crowd intelligence in the Web search field today.

The goal of this research is to propose a platform which can simplify the tasks of aggregating the wisdom of crowds to improve Web search results. From the researchers point of view, there are several key components to complete the platform:

1. data processing tools to be used by users who want to contribute to the construction of information sources
2. a specially designed search tool set that redirects user queries to suitable information sources built by other Web users
3. a feedback and analytic system that collects user feedbacks to adjust the ranking of information sources

The following figure depicts the usage of the proposed platform:

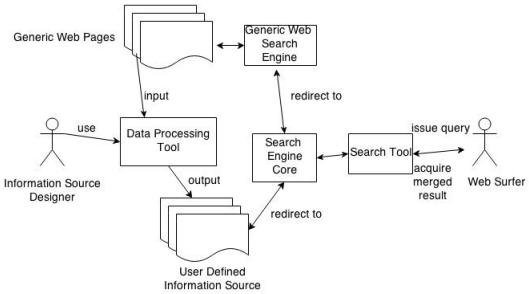

Fig.1: *Overview of the platform.*

In the following sub sections, these components will be explained in detail.

3.1 Data Processing Tools

An information source stores a set of Web resources that are considered to be related to each other. The framework introduced in this paper will be usable only if there are information sources. Data processing tools are designed for building user-defined information sources. Two types of data processing tools are included in the platform: annotation tools and aggregating tools. Annotation tools are used for adding meta-information to existing Web resources. Meta-information associates grammars or meaning to Web resources. As a result, these Web resources can be further processed by other components. On the other hand, aggregating tools aggregate information extracted from Web resources. These two types of data processing tools will be explained in detail in sub sections below. Annotation tools consist of schema sub components, annotation sub components, and aggregating tools. They will be described in detail in the following sub sections:

3.1.1 Schema sub components

Extracting information from relational databases is usually much easier than extracting information from Web pages since the formers are structured information sources while the latters are not. There are various information extraction technologies that are designed for the Web. However, most of them suffered from unstability and limited-adoption issues. To ease the data processing task, the proposed platform provides schema sub components that are used for designing and maintaining schemas. Here, the researcher would like to avoid complex schema structures such as XML schema and DTD files. Instead, the researcher proposes an Object-Oriented Schema Model (OOSM). An OOSM is defined as the following:

(ROOT_ELEMENT, (RULE)*)

That is, an OOSM consists of a root element and a list of rules. A ROOT_ELEMENT is represented as a namespace URI and a local element name pair

and is used to group OOSM instances into categories. Furthermore, a RULE is represented as the following:

(ELEMENT, CONSTRUCT1, CONSTRUCT 2, , CONSTRUCT n

That is, a RULE consists of a root ELEMENT and a set of unordered schema CONSTRUCTs. Again, an ELEMENT is represented by a namespace URI and a local element name. A schema CONSTRUCT is further defined as the following:

CONSTRUCT ::= ARRAY(ELEMENT) | ELEMENT

That is, a schema CONSTRUCT is either an array of instances of an ELEMENT or an ELEMENT alone.

Keeping simple is essential for OOSM to work. Note that the order of child COSTRUCTs of an ELEMENT is not strictly enforced. They are just regarded to as properties of an object according to the convention of the object-oriented paradigm. Furthermore, there is no complex structure defined in OOSM. Different from complex standard such as XML schema, which defines quite a few structures: sequence, choice, and all, etc., there are only two types of CONSTRUCTs defined in OOSM: array of ELEMENTs and a single ELEMENT. The reason of sticking to simplicity is that most Web users are not domain experts of the grammar-definition field, thus complex grammar model may become an obstacle of wide adoption.

Additionally, in OOSM, the content of an ELEMENT is not strictly typed. An ELEMENT simply attaches some form of meaning to the content surrounded by it. That is, it is simply a tag. Because of the diversity of Web contents, the researcher would like to claim that a strictly typed grammar model can be difficult to adopt. On the other hand, an ELEMENT itself, that is, the namespace URI and local name pair, already provides a good amount of information for the content associated with it. There are drawbacks, of course, since post-processing will be needed to extract information from an ELEMENT. However, an OOSM is just like a relation definition in a relational database and performing post-processing to extract information from a relation definition (for example, issue a sql query and then post-process results returned from the sql query) is not uncommon.

3.1.2 Annotation sub components

Schemas must be associated with data to work. Annotation sub components are designed for specifying such associations. As shown above, in OOSM, an ELEMENT annotates a portion of a Web page. For Web users, the steps of annotating a Web page are thus:

1. choose a ROOT_ELEMENT from the schema category; the ROOT_ELEMENT thus identifies a specific schema and represents a specific concept
2. annotate the Web page with ELEMENTs defined in the selected schema according to rules included in the schema; it is not required to fully annotate a Web page, that is, some ELEMENTs can be skipped, and values of these ELEMENTs are simply null
3. note that chained annotations can be achieved if an ELEMENT defined in a schema is the ROOT_ELEMENT of another schema

Compared with existing tagging systems, it seems as if the proposed annotating mechanism restricts Web users freedom of choosing annotations for Web resources. However, the design makes annotation information more usable. By adhering to rules specified in the selected schema, annotated Web resources can be further processed by software modules or domain experts. Furthermore, despite of the additional constraints, the simplicity in the schema structure minimizes the inconveniences.

The proposed annotation system should perform well if only presentational or nominal information is needed. However, the system may be insufficient for extraction detailed information. For example, problems can occur if an application has to extract numeric information for performing calculation since the contents of ELEMENTs are not strongly typed. To remedy this, further postprocessing can be needed. Therefore, aggregating tools are also included in the proposed platform.

3.1.3 Aggregating Tools

Aggregating tools support the following operations: data extraction/normalization, aggregation, join, and projection. Data extraction operations construct datacentric views for sets of Web resources. Here, the researcher would like to distinguish between presentation-centric and data-centric view of Web resources. Presentation technologies, such as HTML and CSS constitute the presentation-centric view. For ordinary Web users, this is the only view in their daily Web usages. Presentation-centric view is appealing to human beings but difficult to process for software agents. On the other hand, data-centric view is more valuable to software agents since it consists of meaningful information extracted from Web pages. In the proposed system, annotation information contributed by Web users is utilized. Annotation information along with the annotated data and the associated OOSM forms the datacentric view of the specified set of Web resources.

Data extraction cannot be done without data normalization. Data on Web pages can be incomplete, inconsistent, and noisy. For example, to demonstrate the same piece of information, different Web pages can use different formats or different units of mea-

surement. Even with grammar information, such issues can still exist. One solution is to define very detailed grammar rules. With such rules, users can specify very detailed information such as formats or units and thus no ambiguity can happen. However, such type of grammar model will be too complicated to be used by ordinary Web users, not to mention wide adoption. To keep the grammar model lean and to prevent ambiguity, using data normalization mechanisms to fix issues within data on Web pages may be preferred.

In the proposed project, in addition to the data extraction layer, a data normalization layer is also included in the aggregation tool. Web users specify normalization rules by writing or reusing simple scripts. Several types of script functions are pre-built including: object graph traversing functions, tag cleaning functions, type conversion functions, and primitive data manipulation functions.

After data extraction, the original html page will be converted to a JSON-like object graph. Hence, object graph traversing functions are designed for accessing properties on the graph. A property may store a DOM node. If only the text portion of DOM node is needed, tag cleaning functions can be used to remove all non-text nodes. Type conversion functions convert texts to other primitive-typed data. After that, the data can be processed by primitive data manipulation functions.

Aggregation operations definitely play an important role in data processing. These operations become available only after data being extracted and normalized, that is, they can only be applied to data-centric views of Web resources. In the proposed project, several aggregation operations such as min(), max(), sum(), and avg(), etc. are defined. With these aggregation functions, Web users can define a set of related Web resources and get further information from them.

Regarding to data-centric views of Web resources as tables in relational models, the join operation should also be available. With the join operation, cross cut analysis among Web resources can be performed. Such analysis is valuable if joined information across Web resources associated with different grammars is desired. For example, with the operation, information from news sites and travel agent sites can be integrated together. The join operator introduces new grammar models by merging participating models.

Here, the projection operation refers to the action the selects a partial set of ELEMENTs defined in the same schema from different Web resources. The operation is required since in some information sources, only part of information defined in the included Web pages will be needed.

3.2 Search Tool

It is without doubt that how to integrate the proposed platform with existing search engines can be an issue. In this research, a set of prototyping search tools are implemented. The tool set includes a search engine core and a front-end tool. Instead of building a generic Web search engine on our own, we utilize the functionalities provided by modern search engines. The proposed system works in the following way:

1. users define advanced information sources and publish them to the Web; a published advanced information source has to declare Web resources that can be processed by it
2. search request redirected to integrated generic Web search engine; the query will be executed and a list of returned Web pages will be intercepted by our tool
3. urls of returned Web pages will be used to determine the set of advanced information sources that can be used; only applicable advanced information sources should be contacted
4. use user-preference and accumulated scores given by the crowd to determine the target set of advanced information sources
5. search engine users then have two sets of returned results: one from the integrated generic Web search engine and the other from the advanced information sources selected
6. users utilize the front-end tool to navigate within the returned set of results and provide feedback to the search engine core

The functionalities provided by the search engine core and the front-end tool will be explained in the following sub sections respectively.

The search engine core handles queries. Due to the specialty of the proposed platform, the search engine core has to touch two different sets of information sources. One is the generic Web and the other is advanced information sources built by crowds. Therefore, for a single query, the search engine core generates two sets of results: generic results and aggregated results. The success of existing Web search engines in processing generic Web pages makes it unreasonable to re-invent the wheel. As a result, we focus on aggregated results while leave generic results to generic Web search engines, i.e., for the generic part, we simply use the application interfaces provided by generic Web search engines. In addition to querying against crowd-made information sources, the search engine core should cache the results generated by them for maximizing performance. Aggregated results are ordered according to the scores recorded in the feedback system. The order of generic results is basically preserved except that Web pages referenced by aggregated results will be prioritized and thus will have higher ranks.

The front-end tool provides several important operations. First, users use its user interface to issue queries. Once query results are returned, besides navigating among them, users can change query criterion by modifying keywords or adjusting categories. Once different keywords are issued, a new query session will be started. On the other hand, adjusting categories results in re-querying matched user-defined information sources. Since the search engine core prioritizes generic search results referenced by results from user-defined information sources, adjusting categories results in reordering of generic results, too. One thing to be noted is that this step will not result in issuing queries to integrated generic Web search engine again. Furthermore, the front-end tool is also used for giving feedbacks. There are three types of feedbacks: tagging, category selection, and giving scores. Feedbacks will be used for ordering results returned from querying advanced information sources and will indirectly affects the order of results from the integrated generic Web search engine. The feedback system will be explained in detail in the next section.

4. SYSTEM DESIGN AND EXAMPLE

For proof of concept, a set of tools has been implemented. They can be roughly divided into two categories: the data processing tools and the search tools. The former contains a schema construction tool, the OOSM Builder, and an annotation tool, the OOSM Mapper. The latter is mainly a database. In this section, these tools along with a working example will be presented.

4.1 Design of Data Processing Tools

The figure below demonstrates our OOSM builder application:

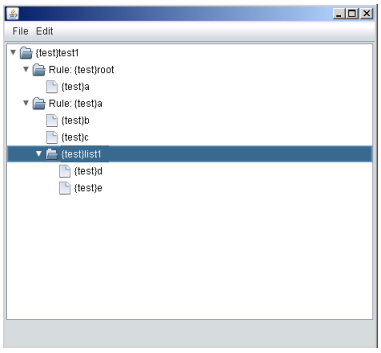

Fig.2: *OOSM Builder Application.*

OOSM Builder is used for designing schemas. As shown above, the schema in design is organized as a list of rules. In the demonstrated schema, the root element, {test}test1, consists of two named rules: {test}root and {test}a. Each rule is then presented as a list of element/element lists. For example, rule {test}a illustrated in figure 2 contains an element list that is named as {test}lists1. Note that in OOSM, a list is actually a repeatable, optional group of elements. The result schema listed below is stored in the JSON format:


```
{
 "rootElement" : {
  "name" : {
   "namespaceURI" : "test",
   "localPart" : "root",
   "prefix" : ""
  }
 },
 "rules" : [
 {
  "headingElement" : {
   "name" : {
    "namespaceURI" : "test",
    "localPart" : "root",
    "prefix" : ""
   }
  },
   "constructs" : [
    {
    "name" : {
     "namespaceURI" : "test",
     "localPart" : "a",
     "prefix" : ""
    }
   }
  ]
 },
 {
  "headingElement" : {
   "name" : {
    "namespaceURI" : "test",
    "localPart" : "a",
    "prefix" : ""
   }
  },
   "constructs" : [
    {
     "name" : {
      "namespaceURI" : "test",
      "localPart" : "b",
      "prefix" : ""
       }
     },
     {
     "name" : {
      "namespaceURI" : "test",
      "localPart" : "c",
      "prefix" : ""
      }
     },
     {
     "elements" : [
      {
       "name" : {
```

```
                "namespaceURI" : "test",
                "localPart" : "d",
                "prefix" : ""
                }
              },
             {
              "name" : {
                "namespaceURI" : "test",
                "localPart" : "e",
                "prefix" : ""
              }
             }
            ],
             "name" : {
               "namespaceURI" : "test",
               "localPart" : "list1",
               "prefix" : ""
             }
            }
           ]
          }
         ],
         "name" : {
          "namespaceURI" : "test",
          "localPart" : "test1",
          "prefix" : ""
         },
          "description" : "description"
}
```


OOSM Mapper is used for annotating a HTML source against a selected schema. The figure below shows the user interface of OOSM Mapper:

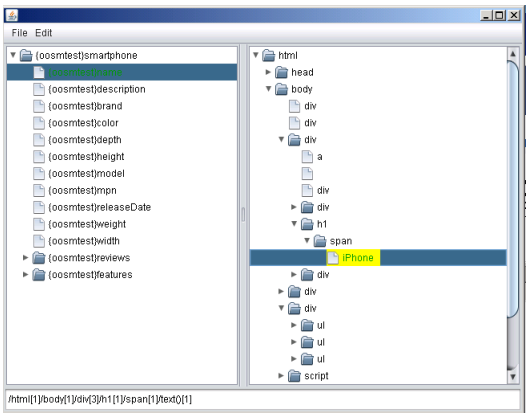

Fig.3: *OOSM Mapper Application.*

OOSM Mapper is designed around the project-centric concept. A project contains an OOSM file and a HTML document. Users either specify the file path or the URL of the HTML documents to be annotated. If a URL is specified, OOSM Mapper will automatically fetch the document from the specified URL. Note that, a normalization process will be performed against the retrieved document to avoid common HTML errors.

To annotate the target HTML document, one simply selects a schema node from the left panel and a HTML node from the right panel, and then trigger the binding dialog shown in figure 4. In the binding dialog, there are three input fields: ID, Definition, and Target. The ID field shows the id of the current binding, which is managed by OOSM Mapper automatically. The Definition field shows the schema node used for annotating the selected HTML node. The Target field shows the specification of the annotated target node. By default, the Target field shows the xpath of the selected HTML node. However, users can enter any valid xpath expression here, including xpath functions. The design will be useful when it is inadequate to annotate a whole node. For example, users may want to annotate a schema node to only a sub string of the selected HTML node.

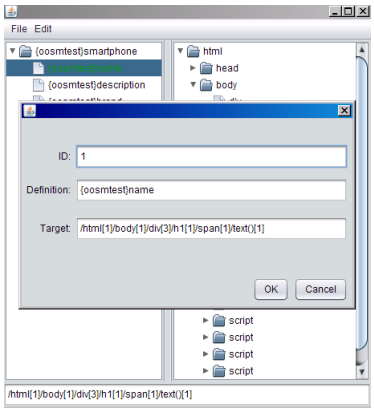

Fig.4: *The Binding Dialog.*

Finally, to view the extracted results, one simply opens the show result dialog.

4.2 Design of Search Tool

A naïve approach to design a search tool will be simply inserting the extracted data into a relational database. Then, plain SQL queries can be used for searching. However, the dynamic nature of the Web makes the design of schemas of relational databases difficult. Additionally, the possibly huge amount of extracted information can make relational databases too slow to be usable. Furthermore, the extracted result is wrapped into an instance of the elaborate.tag_analysis.oosm.instance.binding. EvaluatedObject class, which in fact contains a map of (key, value) pairs. The EvaluatedObject can in turn generate results in either HTML or JSON format. Hence, a more natural way is to put extracted information into a NoSQL database and rely on the search capabilities provided by the database. In this research, the MongoDB database is chosen as the underlying database. MongoDB accepts JSON-style documents and provides a rich set of query functionalities that can fit our requirements. Last but not the least, the perfor-

mance of MongoDB is good and it supports replicas, which is essential for dealing with a very large amount of data.

4.3 Example

Here, a working example is used to further clarify our concept. The example is based on the smartphone ontology obtained from http://www.productontology.org/. The originally schema is in RDF format, which is more complex than OOSM, so we simplified the schema by removing the relation information in the schema and converted it into OOSM format. Then, the Wikipedia page for iphone is chosen as the target document. The schema is shown below:

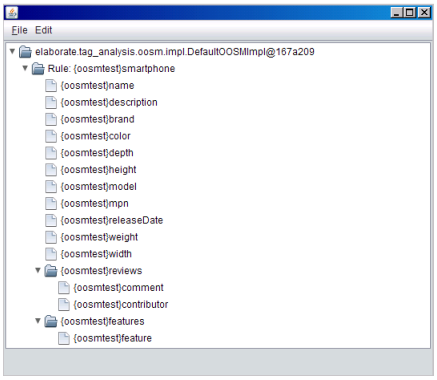

Fig.5: *The (Modified) SmartPhone Schema.*

As shown above, reviews and features are designed as element lists since there can be more than one of them. Then, we designed the bindings. Since the text nodes of the Wikipedia page are usually long paragraphs, we have to use the substring xpath function to extract needed information from the page.

4.4 Search Example

For concept-proof, a sample search engine that is shipped with the smartphone schema presented above has been built. In the current stage, the functionality of selecting the desired information source is not implemented, so users have to select their target information sources manually. The current implementation supports both full-text queries and attribute-oriented queries. The search engine is hosted by our own Web server and the url is "http://ilab1.twgogo.org:8080/OOSMWebSearchClient/s martphone/index.jsp. The figure below shows the homepage:

When a full-text query like "iphone is received, our search engine simply looks into the "description element of the smartphone schema. On the other hand, we rely on the underlying mongo database for attribute queries. With schema information, users can issue powerful queries such as:

Fig.6: *The OOSM Smartphone Search Page.*

$$\{ \text{“}\{oosmtest\}weight\text{”} : \{\$lt : 130\} \}$$

The above query can be used for looking for a smart phone with weight less than 130 gram. The result is shown below:

Fig.7: *Result of the Sample Query.*

Furthermore, clicking the "See Detail link will display the original JSON document of the selected entry. Compared with the google search engine, benefiting from the schema information, the proposed method not only provides richer query functionalities through attributeoriented queries but also returns more precise information when the target domain is known in advance.

5. DISCUSSIONS

5.1 Integration with Existing Search Engines

Today, there are already several big companies dominating the search engine market. Re-inventing the wheel, that is, building our own search engines is not practical. Instead, the goal of this research is to complement existing search engines. The core concept of this research is that advanced information sources, information sources made by domain experts, are helpful for domain-specific search. The provided tools simplify the construction of advanced information sources. When the target domain is known clearly, users simply point to the desired information sources managed by our system for querying. The next goal is to build a recommendation system for appropriate information sources. When users submit

queries, the recommendation system extracts labels from result lists returned by search engines and use labels for suggesting suitable information sources.

5.2 Feedback and Analytic System

Feedbacks from users are used for determining the ordering of search results. The current implementation provides three types of feedbacks to interact with end users: tagging, category-re-selection, and scoring. Tagging reflects what users think an aggregate or generic result is about. By collecting tags provided by users and performing analysis, possible categorizations of Web resources can be inferred. Tagging has two different types of effects: direct and indirect. For ordinary users, assigning tags to plain Web pages is the natural behavior. The direct effect of tag assigning is to determine which categories a Web page belongs to. However, since assigning new tags to a Web page may change the current categorization results, tag assigning also has indirect effect on information sources containing the Web page.

A problem of the auto-categorizing system mentioned above is that it is very difficult if not impossible to make the result accurate. As a result, allowing evolving of categorizes becomes an important factor of success of categorization systems. The indirect effect mentioned above forms the category-re-selection process and can be viewed as a complementary mechanism. Once a user decides to switch to a different category, the action reveals that the user does not agree with the keyword-category association recorded in the system. Hence, the information should be used to adjust the automatically generated categorizations. This complementary system will be beneficial to the categorization accuracy, but the adoption of the information should be conservative since behaviors of Web users can be very diverse.

Additionally, a straightforward scoring system is also implemented. The scoring system is only applied to userdefined advanced information sources since we have no intention to mess up with generic Web search mechanisms. However, as stated in former sections, the ordering of aggregated results will affect the ordering of generic results, only that such ordering is just presentational and the internal ordering recorded by the integrated Web search engine will be kept untouched.

5.3 DOM Modification Problem

We can roughly categorize annotation systems into two types: internal annotation system and external annotation system. The former requires annotations to be directly embedded into Web pages. Such annotation systems are more stable since whenever a Web page is modified, annotations associated with it can also be adjusted to fit the modification. However, internal annotation system can be inconvenient. The major problem is it affects the way ordinary Web users authoring Web pages. On the other hand, the latter approach is usually more adoptable. With external annotation systems, Web pages themselves are kept untouched. Besides, another benefit of external annotation systems is they allow multiple sets of annotations to be attached to the same Web page.

However, such annotation systems typically suffer from the DOM modification problem. That is, once the structure of a Web page is changed, the applied annotations can become invalid since usually a DOM path is used for associating an annotation to a piece of the Web page. Web pages are modified frequently especially for those dynamically generated pages. The situation makes it really difficult to develop and maintain external annotation systems.

In the proposed mechanism, a special schema system named as OOSM is adopted. By creating mappings between an OOSM and a Web page, the annotation process is accomplished. The mapping process does not require positioning annotations accurately. The nature of OOSM makes it easier to deal with DOM modification since it allows annotating a sub tree in a DOM. The design can cause problems since it produces weaklytyped results. Hence, to make OOSM effective, data processing tools mentioned above is required. Compared with the DOM modification problem, the researchers would like to claim that OOSM is still valuable since the weaklytypes issue can be remedied with scripting functionalities.

6. CONCLUSIONS AND FUTURE WORK

In this paper, a Web search mechanism elaborating crowd wisdom is proposed. Our approach provides a way for users to define advanced information sources, which are just like views of relational databases. Based on advanced information sources, some more feature-rich search operations are also provided. The researcher claims that the proposed mechanism is more efficient and powerful than ordinary keyword-based search. The current implementations are mostly desktop-based GUI applications. In the future, it is planned to port the current implementations to be Web-based for easier integration and adoption.

ACKNOWLEDGEMENT

First, I have to acknowledge NSCs support for the completion of this research. Furthermore, the paper was prepared in collaboration with my lab members in Department of Information Management, Nanhua University, Taiwan. I would like to acknowledge the following persons who have made the completion of this research possible: Chu-Chun Chuang, Jia-Hua Wu, Han-Ci Syu, and Yan-Ru Jiang.

References

[1] J. Franklin, D. Kossmann, T. Kraska, S. Ramesh and R. Xin, "CrowdDB: answering queries with crowdsourcing," *In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*,pp. 61-72, 2011.

[2] A. Bozzon, M. Brambilla and S. Ceri, "Answering search queries with CrowdSearcher," *In Proceedings of the 21st international conference on World Wide Web*, pp. 1009-1018, 2012.

[3] A. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh and J. Widom, "CrowdScreen: algorithms for filtering data with humans," *In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 361-372, 2012.

[4] A. Bozzon, M. Brambilla, S. Ceri and P. Fraternali, "Liquid query: multi-domain exploratory search on the web," *Proceedings of the 19th international conference on World wide web*, pp. 161-170, 2010.

[5] P. Heymann, G. Koutrika and H. Garcia, "Can social bookmarking improve web search?," *In Proceedings of the 2008 International Conference on Web Search and Data Mining*, pp. 195-206, 2008.

[6] A. Parameswaran, A. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom, "Human-assisted graph search: it's okay to ask questions," *The Proceedings of the VLDB Endowment*,vol.4, pp. 267-278, 2011.

[7] Amazon, https://www.mturk.com/mturk/welcome

[8] D. Konopnicki and O. Shmueli, "Database-inspired search," *In Proceedings of the 31st international conference on Very large data bases*, pp. 2-12, 2005.

[9] Yahoo, Yahoo! Query Language, http://developer.yahoo.com/yql/

[10] R. Trillo, L. Po, S. Ilarri, S. Bergamaschi and E. Mena, "Using semantic techniques to access web data," *Information Systems 36*, pp. 117-133, 2011.

[11] S. Chun and J. Warner, "Semantic Annotation and Search for Deep Web Services," *In Proceedings of the 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and EServices*, pp. 389-395, 2008.

[12] N. Sarkas, S. Paparizos and P. Tsaparas, "Structured annotations of web queries," *In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 771-782, 2010.

[13] S. Paparizos, A. Ntoulas, J. Shafer and R. Agrawal, "Answering web queries using structured data sources," *In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 1127-1130, 2009.

Chun-Hsiung Tseng received his B.S. in computer science from National National ChengChi University, and received both M.S. and Ph.D. in computer science from National Taiwan University. He was a research assistant of Institute of Information Science, Academia Sinica in 2003-2010. He was a faculty member of Department of Computer Information and Network Engineering, Lunghwa University of Science and Technology in 2010-2013. His current position is a faculty member of Department of Information Management, Nanhua University. His research interests include big data analysis, crowd intelligence, e-learning systems, and Web information extraction.