Semantic Enrichment of Search Result: the Coupling of Semantic Store and Google Services

Khabib Mustofa, Amin Andjomshoaa, A Min Tjoa
Institute for Software Technology and Interactive Systems
TU Vienna, Austria
{khabib, andjomshoaa, amin}@ifs.tuwien.ac.at

Abstract. Search engine has proven itself to be a powerful tool in finding documents on the web. Google as one of top search engines has become very popular and then, extends its services by providing Application Programming Interface (Google Search API) and many more features. We utilized some of its services to enrich our data for SemanticLIFE system. In this paper we discuss how semantic information can be used also to enrich the search results of search engines with long-term user profile. To clarify the idea, we propose an intermediate service which combines search results of Google with information from the semantic store of the SemanticLIFE system and then present the users with enriched results, possibly in different rank and order from the original. The service is implemented according to the Service Oriented Pipeline Architecture (SOPA).

1. Introduction

Since it was introduced and widely known in 90s, the World Wide Web (WWW) has been turning into a monotonically increasing source of information. This characteristic introduced, at the same time, the increasing difficulty in locating a piece of information on it. Without an appropriate tools it seems in general holds that the more the item, the more difficult to find a desired one. Many researches were conducted to result in sufficiently handy tools for locating piece of information on the web, termed as search engine, like: Google, Yahoo, MSN, Dmoz and Altavista. According to [8], most of these search engines resulted from the research in which the method of locating information is by exhaustively crawling the internet and archiving the crawled web pages. Besides those individual services, Webcrawler provides services by offering combination of results from several search engines.

Among the popular search engines, Google is getting more famous and extends its services time by time, not only covering web pages but also videos, books, publications and even translation tools. Later, it provides also several Application Programming Interface (API) to give developers flexibility and reusability of Google data and services. Some of these services (Google Desktop, Google Search
API, Google Scholar) and their contribution in our architecture will be discussed briefly on Section 3.

Current search engines so far support search of documents based on patterns and/or word matching. Documents or web pages containing text which match the given words are possible to be hit by search engine. Thus, the smallest identifiable object is a document, not the information in the document itself. It is considered as the WWW is currently web of documents, consisting of connected documents through link. This limitation comes to expert’s mind to extend the current web with support to identification of smaller granularity of information. As the preliminary idea, to get close to the Vannevar Bush’s Memex[6], Tim Berners Lee proposed what is called Semantic Web[3], which shifted the idea of the web from web of document to web of data [9]. Voluminous data exist on the web, coming from different domain of concept: social, politics, economy, biology, mathematics and many more. The extension to the web is proposed by introducing and incorporating ontology, a formalization of common conceptualization, which is seen as a growing need to support data exchange and integration among different domain[13].

To realize the idea, many works either idea, standards or implementations have been developed. RDF/RDFS and OWL are recommendations from W3C for ontology formalization. Haystack [10] and e-Person [11] are two among several examples of application for personal information integration and sharing. CREAM, SCREAM, Semanticword, SMORE and PhotoStuff⁶ are annotation tools harvesting the Semantic web. Another effort to realize the fruitfulness of Semantic web is SemanticLIFE which will be briefly touched in Section 2.

As browsing session becomes almost daily activity and there are many other personal activities, SemanticLIFE tries to maximize its supports by covering various personal data. In this paper we will discuss its additional feature which incorporates some Google services to achieve the idea, as will be discussed on Section 4.

2. SemanticLIFE

SemanticLIFE, on its very preliminary stage, has been introduced in [1]. It is designed to store, manage and retrieve the lifetime’s information entities of individuals. It enables the acquisition and storage of data while giving annotations to email messages, browsed web pages, images, contacts, life events and other resources. It also provides intuitive and effective search mechanism based upon stored semantics, and semantically enriched user interfaces according to user’s needs. The ultimate goal of the project is to build a Personal Information Management system over a Human Lifetime using ontologies as a basis for the representation of its content.

On its latest development, SemanticLIFE has been implemented using the Eclipse Rich Client Platform (Eclipse RCP) technology following the industry standard Eclipse IDE. SemanticLIFE system benefits the most from plug-in mechanism and extension point concept of Eclipse system. The whole system has been designed as a set of interactive plug-ins that fit into the main RCP application and this guarantees flexibility and extensibility of SemanticLIFE platform.

Communication within the system is based on a service-oriented design with the advantage of its loosely coupled characteristics. To compose complex solutions and scenarios from atomic services which are offered by SemanticLIFE plug-ins, the Service Oriented Pipeline Architecture (SOPA) has

⁶http://www.mindswap.org/2003/PhotoStuff
been introduced [2]. SOPA provides a paradigm to describe the system-wide service compositions and also external web services as pipelines. A pipeline in SOPA terminology is a uniquely named set of service-calls and intermediate transformations. The pipeline concept provides a higher level of abstraction between services and applications that are benefiting SOA. SOPA provides some mechanisms for orchestration of services and transformation of results. It also offers many data processing and flow management features. An overview of the system architecture is depicted in Fig. 1, while listing 1 shows an example of a pipeline.

Listing 1: Basic structure of a pipeline

```xml
<pipeline name="googleSearch">
  <parameters>
    <parameter name="keyword" type="String"/>
  </parameters>
  <call id="rawResults" service="at.slife.googleWebService" operation="query">
    <parameter>{keyword}</parameter>
  </call>
  <xsl:for-each select="/result/rawResult">
    <call id="slifeItem" service="at.slife.slifeRankService" operation="combinedRank">
      <parameter>{item}</parameter>
    </call>
  </xsl:for-each>
  <call id="sortedResult" service="at.slife.slifeRankService" operation="sort">
    <parameter>{xpath:/result/slifeItem}</parameter>
  </call>
  <transform xsl="google.xsl"/></transform>
  <serialize="html"/>
</pipeline>
```

Data with user annotation is fed into the system using a number of dedicated plug-ins from variety of data sources like Google Desktop captured data, communication logs, and other application’s metadata. The data objects are passed on by the message handler to the analysis plug-in. This plug-in contains a number of specific analysis plug-ins providing semantic mark-up by applying a bunch of feature extraction methods and indexing techniques in a cascaded manner. The semi-structured and
semantically enriched information objects are forwarded to the repository plug-in for an ontologically structured storage. In the SemanticLIFE system, data sources are stored in forms of RDF triples with their ontologies and metadata. This repository is called a meta-store

3. Google Services

In the few years, Google has positioned itself as a search engine with most complete support for users. Despite being a search engine for web, Google has extended its service by offering also: APIs for giving developers ease in developing application interacting with Google data and some other services.

In this section we will note briefly on some of Google services, their characteristics and their valuable contribution in the SemanticLIFE project.

- Google Search and Google Search API Google search is the first and world-wide well known service from Google Inc. Continuous effort to make it better deserves a world-wide reputation and become the first choice for surfer searching information on the web. It is granted as the Most Outstanding Search Engine by Search Engine Watch readers[14]. From users’ point of view, Google provides fast service, cached pages, spell checking for suggestion of possibly typo errors and suggestion of similar pages based on similarity method applied by Google. To enhance the quality and relevance of search result, Google involve many factors to calculate the rank.

Page rank calculation adopted the concept of academic citation, in the sense that the more citation to publication X can be considered representation of its importance over others. Page rank extends this idea by not counting links from all pages equally but by normalizing the number of links on a page. The formula for page rank calculation is as follows:

Let’s assume page $A$ has pages $T_1...T_n$ which point to it (i.e., are citations). The parameter $d$ is a damping factor which can be set between 0 and 1. $C(A)$ is defined as the number of links going out of page $A$. The PageRank of a page $A$ is given as follows[5]:

$$PR(A) = (1-d) + d \left( \frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)} \right)$$

The intuitive meaning would be a page can have a high page rank if there are many pages that point to it, or if there are some pages that point to it and have a high page rank.

Besides, Google claims that most search engines associate a link with the page that the link is on. Google associates it with the page the link points to, too. Furthermore, Google keeps track of some visual presentation details such as font size of words. Words in a larger or bolder font are weighted higher than other words.

However, such pagerank is likely to be abused. Based on experience, cloaking even happened on commercial search engine like Google. Cloaking is a trick in which software code hidden in the webpage detects the Google spider and presents it with different content from the page normally presented to site visitors [4]. As one of ways to reduce such effect, an intermediate service can be developed which submits query to Google via Google SOAP Search and combines other factor. In SemanticLIFE, this will capture the result, and after combined with the
semantic store, forward to another service to calculate the new rank before presenting to users. Detail architecture of this service is discussed in the next section (Section 4).

- **Google Desktop.** Google Desktop provides support to log activities on local computer. It gives easy access to information on computer and from the web, supports full text search over user’s email, files, music, photos, chats, Gmail, web pages that have been viewed by users. Considering this feature, we use Google Desktop as one of data sources to SemanticLIFE System, especially to collect data supporting for mining user preference, and later to create user profile. Using this profile, the system’s behavior will be adjusted based on semantic information available.

- **Google Scholar.** Publication is one of valuable information for researcher, and SemanticLIFE is aimed initially at supporting this area of domain. Seeing the large amount of data available through Google Scholar, we developed wrapper and API to parse and extract data from it. Users who do browsing can store the page like usual webpage. Furthermore, if the browsed page is search result from Google Scholar, the page will be parsed. This mechanism is triggered by checking the DNS or host of the pages’ source. Knowing that the host is Google Scholar, some plugin will be activated to parse the page, and return XML containing publications information in the result page. This XML will later be converted to RDF to be stored in meta-store. An application is also proposed to bridge between Google Scholar and Semantic store like that for the web search using Google SOAP Search API.

## 4. Proposed Solution

As discussed in previous sections the site’s ranking in Google’s search results is automatically determined by some predefined algorithms that use thousands of factors to calculate the relevance measure of result to a given query; however the user profile and interests are not included in rank factors. Google has tried to bridge this gap by introducing Personalized Search\(^7\) that is part of Google’s ongoing effort to make our search experience more relevant to us. Personalized Search enhances the search results based on what we have been searching for in the past. The weakness of this approach is that it enhances the results only by considering search history and does not apply other personal parameters. In this section we will introduce SemanticLIFE’s approach that enriches the rank of search results based on long term user behaviour.

To illustrate and evaluate the SemanticLIFE approach we will step by step sketch a typical scenario for an academic literature search. Recently Google has provided a specific search engine called ”Google Scholar” that makes it possible to explore many academic resources like peer-reviewed papers, theses, books, abstracts and articles, from academic publishers, professional societies, preprint repositories, universities and other scholarly organizations. In this use case the order of search results could be greatly influenced by user profile including interests, fields of study, projects, etc. Some interesting indicators to assess the results’ rank are:

- **Similar Authors:** Since the scientific papers are more or less well structured and follow a general schema, useful information can be extracted from semantic repository. Google Desktop also creates a full text index on all documents that makes the process of author matching much easier. Semantic repository contains information of all previously viewed papers and especially

\(^7\)
http://www.google.com/support/bin/topic.py?topic=1593
the paper authors. It is quite relevant that the publications with recognized authors should precede those of unknown authors. This could be considered as a measure to show the user’s interest to publications of a specific author.

- **Keywords**: literature key words can also be checked against the research interests of user captured in user profile. Also the snippet of the text accompanying search results can bind the item to domain ontology and annotations. The more bindings we can find between item and semantic space, the higher rank will be assigned to that item.

- **Cited literatures**: if the search results has been already cited by an existing literature in semantic space, or search results refers to an existing literature in semantic space then that search result sounds more relevant.

- **Inferred relations**: Sometimes two concepts are not associated directly to each other; i.e. the relationship between such items should be logically established by some rules. For example if the email address of paper author in search result is given as amin@ifs.tuwien.ac.at, and the domain ifs.tuwien.ac.at is annotated as a project partner then the paper might be more interesting for the user. Another possibility is to check the semantic space and web history to find out if any web pages from this domain have been visited.

- **Contacts**: It is also interesting to know if the user has had any mail contact with the paper authors. If so, then the literature deserves a higher rank among other search results.

### 4.1. SemanticLIFE Enrichment Process

In this section we will have a closer look at some SemanticLIFE components that are included in ranking process and explain their roles in detail.

The search process in proposed scenario will start from SemanticLIFE’s Google API component. The Google API of SemanticLIFE platform is capable of running queries against Google Scholar, and returns the results ordered by Google rank in XML format. The search results usually include author and publisher information, a snippet of result document, publication year, etc. this result should be scanned to assess a better rank for search results using user profile.

SemanticLIFE platform is equipped with a user profile plug-in that tracks the dynamic, long term user activities to find out the user interests which might even change from time to time. The information captured by this plug-in can later on be refined and annotated by the user to make a more precise user behavioural model. User profile is basically built upon the items that have been captured by Google Desktop and are stored in SemanticLIFE’s RDF repository. This includes the detailed information about emails, local files, contacts, etc. To enrich the ranking method of Google search with user behavior model, we combine the results of Google Search with user profile by means of SOPA pipelines. SOPA coordinates and composes the search results via introducing a semantic pipeline. The following figure shows a simple pipeline that applies the required ranking paradigm to search results:
4.2. Semantic Ranking

4.2.1. Frequency ranking

In this method the number of relevant items and their timestamps will be used to create a ranking measure. This method benefits from the domain ontology by taking the related items into account. For example consider the case that we are looking for information about “Semantic Web”. In traditional search result ranking methods only the items including the term Semantic Web could be considered as result items (full text search). But using a domain ontology we can find out more about a term, for example we may also add all the items containing any of the following terms too: Ontology, OWL, RDF and even though “Semantisches Web” (the term “Semantic Web” in German language). Mathematically speaking our result set will be union of all result sets of related concepts: Result set \( R = \bigcup_{C_i} F_{C_i}(C) \) where \( C_i \) is related class to our target class with “same-as” or “subclass-of” predicates and \( F \) denotes the frequency of the corresponding term. To apply the notion of time to the result set items we may give a heavier weight to recent items for result ranking which converges to the forgetting mode of memory. For example a research topic that has been searched for, one hour ago should have a higher importance than web pages that are browsed one year ago and contain the research topic name. For this purpose a monotonic decreasing function has to be used to weight the result set (as explained above, including all semantically related concepts) and calculation of the total frequency rank.

4.2.2. Relevance Rank

This ranking method used is built upon the matches between ontology of the search results and the user profile ontology based on his life-time history. For this purpose we will use a computational model that assesses similarity by adding a semantic measure to “feature matching process” and is based on ratio model that is introduced by Tversky(1977) as discussed in [12]. To assess the semantic distance between two concepts (classes) A and B, we will first build an extension set for each of classes. This extension set includes all semantic terms that are somehow related to the core concept. For example the extension set of “Semantic Web” concept which is denoted as Ext(“Semantic Web”), will include all related features that are related to Semantic Web. Based on these assumptions and discussion in [12], the Semantic distance between search item concepts (A) and related user profile
concepts (B) can be determined by:

\[ D(A, B) = \frac{|\text{Ext}(A) \cap \text{Ext}(B)|}{|\text{Ext}(A) \cap \text{Ext}(B)| + \alpha|\text{Ext}(A) - \text{Ext}(B)| + (1 - \alpha)|\text{Ext}(B) - \text{Ext}(A)|} \]

Where \( \alpha \) is a real number between 0 and 1. The distance function returns a measure between zero and one, depends on the similarity of concepts. For a user searching for a research topic the situation is shown in the Figure 3. The left side of the figure depicts the search results that in this case are publications. The right side shows the User Profile ontology. The dotted lines between the different concepts demonstrate the semantic relationships between them. For example, the user profile contacts could significantly influence the attractiveness of a paper in search results. Using the introduced measure it is possible to make rational hypothesis on user choices and rank the user choices.

4.2.3. Combined Rank

To establish a uniform rank from frequency and relevance ranks we introduce a combined rank that takes both ranks into account. Consider that the semantic distance between users profile and "literatures selected features" (introduced in previous section) is calculated and depicted as \( d_1, d_2, \ldots, d_n \) where \( d_i \) is the semantic distance between i-th selected indicator. Also the frequency rank for each indicator is calculated and denoted by \( f_i \). Then the combined rank for each indicator is \( f_i \ast d_i \). Table 1 shows the indicators and related ranks which have been resulted from either relevance rank or frequency rank. As an optional parameter it is also possible to assign a weight to each indicator to give it a higher influence in overall rank value. Hence, the total rank for each paper can be defined as follows

\[ \text{TotalRank} = \sum_{i=1}^{n} d_i \ast f_i \ast w_i, \quad n = \text{number of indicator} \]

Semantic matching and ontology migration are two major challenges in semantic web field and now they are centre of our attention for next phase of SemanticLIFE project.

5. Related works

Google has been growing as one of top 3 most favourite and powerful search engines[14]. Its several services have been seen as good chance by developers to integrate many ideas with great amount of data Google has. Among the ideas is a project Home Page Finder [8]. The project aimed at providing users with tool for locating website of researchers or conferences. It sends queries via Google API
and then does some after-fetch processes to select best candidates before presenting to users. Guided Google [7] is another implementation of Google API. It offers services of combinatorial search and search by host. Combinatorial search will do multiple queries to Google and combine the result as one, with the search key of each query is the permutation of the original constructing words. With search of 3 words, it will request 6 queries to Google. Zhang et.al [15, 17] propose OntoSearch to bridge the gap of locating some concepts in the ontology while the tool for searching such information is not available. Ontosearch limits its queries to Google by constraining the file type of the ontology (RDFs, DAML or OWL).

Sometimes some routine to extract information is not available in API. Wrapper is alternative to the approach. Tungare[16] developed a .Net wrapper for extracting Google Desktop data.

6. Conclusion and Future Works

In this paper we proposed an approach to get different ranking of Google Search Result. The ranking is obtained by considering semantic information, such as relevance. Long term user profile stored semantically based on ontology can play its role in determining the behavior of the system. Using the proposed approach, user search result are more relevant and closer to the desired information.

As the ongoing project, the result so far is still partial. Nevertheless, when it is accomplished, it can be extended to other web resources than Google Scholars and Google Desktop, such as: Google AdWords and Google Map.

Acknowledgement

This work was supported by ASEA-UNINET Scholarship.

References


