Plinkr

an Application of Semantic Search

John L. Scott

4/28/2009

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in the Department of Computer Science at New York University.

Professor Dennis Shasha, Research Advisor

Professor Zvi Kedem, Second Reader

© John L. Scott

All rights reserved, 2009

Acknowledgements

I would like to thank my research advisor, Professor Dennis Shasha, for inspiring this work and especially for his guidance and patience along the way. I would also like to thank my wife, Melissa, and my son, Addison, for their enthusiastic encouragement and endless support. Finally, I want to thank Nicholas Perito for teaching me what a thesis actually is and for his editorial contributions to this paper.

1 Table of Contents

Acknowledgements 3

1 Introduction 8

2 Background 9

2.1 The Web as Data Source 9

2.2 Web Search 9

2.2.1 Types of Search 10

2.2.2 Search Engine Results Page 10

2.3 Semantic Search 11

2.3.1 Resource Description Framework 11

3 Plinkr 12

3.1 Objective 12

3.2 Overview 12

3.3 Key Concepts 13

3.3.1 Document Sets 13

3.3.2 Entities 14

3.3.3 Snippets of Text 14

4 Architecture and Design 15

4.1 Overview 15

4.2 Search 17

4.2.1 Build Query 19
4.2.2 Submit Query to Search Engine 20
4.2.3 Google Search API 20
4.2.4 Score Document 21
4.3 Content Extraction 21
4.4 Annotation 22
4.4.1 Calais Web Service 23
4.5 Entity Extraction and Aggregation 24
4.5.1 Jena 26
4.5.2 Extract Entity 27
4.5.3 Aggregate Entity 27
4.5.4 Score Entity 29
4.6 Results Generation 29
4.6.1 Get Statistical Results 30
4.6.2 Get Entity Results 30
4.6.3 Get Snippet Results 32
5 User Interface 34
5.1 Query Form 34
5.2 Results Visualization Page 35
5.2.1 Statistics 36

5.2.2 Entity Cloud 36
5.2.3 Snippets 36
6 Implementation Details 37
6.1 Platform 37
6.2 Model 37
6.3 Entity Relationship Diagram 38
6.4 Adjustable Runtime Parameters 39
6.5 Deployment Details 40
7 Conclusion 40
7.1 Evaluation 40
7.2 Future Work 42
7.2.1 Performance 42
7.2.2 Results Quality 43
7.3 Related Work 44
7.3.1 Google 44
7.3.2 Evri 45
7.3.3 Hakia 46
8 Appendix 48
8.1 Appendix A: Google Search API 48
8.2 Appendix B: Calais Web Service 49

8.2.1 Entity Types 49

8.2.2 Entity Type Categories 49

8.2.3 Sample RDF 50

8.2.4 Document Categories 50

9 Bibliography 51

1 Introduction

The World Wide Web is a massive source of information comprised of the unstructured free-form text contained in individual Web pages. This information is traditionally searched, for various purposes, by submitting queries consisting of keywords. A search engine returns a result set of documents, ranked by relevance, that meet the criteria of the query, i.e., documents which contain the desired keywords. When a search is broad, the result set can be quite large, and the effectiveness of the document ranking becomes an increasingly important factor to the usefulness of the search results.

Web searches are often used to research a particular subject; however, with many searches, the result set contains a large number of documents, from which the user must then select and read to find the "right" information. This laborious task is further complicated when searching for information about multiple subjects.

To aid information gathering, Plinkr, a Web application, was developed to transform, analyze and refine Web search result sets. Specifically, Plinkr tackles the problem of discovering what two given subjects have in common. This seemingly straightforward operation can, in fact, be excessively complex when the relationship between two subjects is subtle. In such cases, especially, a machine can be an invaluable research assistant in bringing such subtleties to light.

This paper will show how the transformation of unstructured information to structured machine comprehensible data can be achieved using Plinkr, which extends and enriches traditional keyword search by leveraging existing semantic technologies.

2 Background

2.1 The Web as Data Source

The World Wide Web is comprised of billions of interlinked documents that together create a rich information source. Web documents are designed for human consumption and, in that regard, work very well. But from the perspective of a software agent, or more generally speaking, a machine, this information is readable as text data but not immediately understandable. A machine can recognize an integer, for instance, and can manipulate data stored as such. But Web pages are composed of strings of characters that have no real meaning to a machine. This presents a problem because the volume of data available on the Web makes it impossible to manage manually, yet the nature of this data also makes it difficult to create automated management tasks (W3C: Semantic Web Activity).

Specifically, the problem is that Web data lacks semantic structure. When the machine is a Web browser and the objective is rendering the page for display, the data in a Web page can be considered structured by the inclusion of HTML tags. These tags are recognized by the Web browser, which has been coded to treat an anchor tag a particular way, for instance. But in terms of extracting any sort of semantic meaning, Web data is unstructured or, at best, semi-structured if the document includes meta-tags that explicitly define some aspect of the document - a title or a list of keywords perhaps (Weglarz).

2.2 Web Search

The Web in "World Wide Web" refers to the fact that Web documents are linked to one another. This gives the Web as a whole its general structure and makes it possible for search engines to maintain indexes. It is this searchable aspect of the Web that makes it a viable data source.

2.2.1 Types of Search

Sometimes, World Wide Web users know the exact address of a page or Web site they would like to visit, but oftentimes a search is required to meet their objectives. These objectives may be generally categorized as navigational, transactional, or informational (Broder). A navigational search is when the user is trying to locate a particular Web site. With a transactional search, the user's intent is to perform some type of activity, such as making a purchase.

In this paper, we are concerned with informational searches, where the intent is to acquire information about some subject or subjects. In these cases, a user provides a string of keywords, or perhaps a specific phrase, to the search engine. This query is intended to denote the subject of the user's research. The user's expectation is not to be led to a particular Web site or page but rather to be presented with some number of documents that collectively provide the desired information (Guha, McCool and Miller).

2.2.2 Search Engine Results Page

A search concludes with the presentation of results on what is commonly referred to as the Search Engine Results Page (SERP). This page is essentially a listing of document summaries that include a title, a link to the source, and a brief summary. Since the number of results may be quite large, they are ordered by relevance and presented in subsets. This ordering is known as the SERP rank, and the highest ranking pages are presented first.

Google's PageRank algorithm was an important innovation in Web search technology, and it has played a significant role in elevating the quality of search results. It uses the Web's link structure to determine a Web page's rank, which is "an objective measure of its citation importance that corresponds well with people's subjective idea of importance. Because of this correspondence, PageRank is an excellent way to prioritize the results of web keyword searches" (Brin and Page).

2.3 Semantic Search

As mentioned earlier, the unstructured data contained in Web pages is intended for human consumption and is not readily understandable by machines. Tim Berners-Lee's vision of a Semantic Web of data - actual structured data that machines can understand - is beginning to be realized. The Semantic Web will be an extension of the World Wide Web; therefore, a semantic search can be viewed as an extension of a Web search that attempts to augment and improve traditional search results by using data from the Semantic Web (Guha, McCool and Miller).

While the Semantic Web is currently too sparse, in terms of actual published data, for a general-purpose search application, there are emerging technologies that make a semantic search possible to some degree. Various W3C specifications and recommendations have been published, many software tools are now available, and datasets such as DBpedia are being developed. The Semantic Web will eventually extend the principles of the Web from free-form documents to data (W3C: Semantic Web Activity).

2.3.1 Resource Description Framework

Presently, the problem with using the Web as a data source is that Web data is unstructured and therefore not machine-understandable. The World Wide Web Consortium (W3C) has proposed a solution to this problem that involves using metadata, or data about data, to describe Web resources. The Resource Description Framework (RDF) is a collection of W3C specifications that support the processing of metadata and the exchange of machine-understandable information on the Web (W3C: Semantic Web Activity). For our purposes, RDF provides facilities that enable the automated processing of Web resources.

3 Plinkr

3.1 Objective

The remainder of this paper will show how the developed research tool, Plinkr, facilitates the process of discovering the intersection of information between two subjects. This intersection represents what the subjects have in common and thus effectively captures the relationships between them.

When the relation between two subjects is subtle, a human researcher might have to spend an excessive amount of time reading various documents, highlighting key ideas, listing references to other subjects, and noting pertinent facts and events to draw any conclusions about the relation between the subjects in question. The objective of Plinkr is to present an abstraction of the information obtained by a traditional Web search in such a way that these research tasks are partially automated, thus making the research process more efficient. Additionally, it is our hope that Plinkr discovers relationships that might not be readily apparent without the deep statistical analysis we apply.

3.2 Overview

Plinkr was developed to be a Web application that extends a traditional Web search by using semantic search technology. When Plinkr is initiated, it first performs multiple Web searches to create the pertinent data source. The free-form text contained within each resulting document is given structure via semantic tagging, which is the process of identifying and classifying resources or entities into some category that specifies intention and meaning (Ekeklint). Each entity is then assigned a score that reflects its overall relevance. Key entities within the corpus of the resultant documents are thus abstracted and presented as a sort of metadata. This process effectively distills an unmanageable quantity of text into something more useful. Users can then explore the metadata to isolate the content most relevant for their research goals.

3.3 Key Concepts

3.3.1 Document Sets

Several concepts are relied upon to support the thesis of this paper. Key among them is how we partition our data into multiple sets of documents. We begin with the data collected by multiple traditional Web searches. This becomes our data source. But rather than displaying the result sets as lists of document summaries, as a search engine would, we generate metadata that describes each result set as a whole. This enables a more concise and high-level presentation of the data, as will be described in following sections, as well as a means for determining where the results sets have some commonality.

Given two subjects, we generate three distinct sets of documents by performing three distinct searches: one that includes the first subject exclusive of the other, a second that includes the second subject exclusive of the first, and a third search that includes both subjects. If results are produced for the document set that includes both subjects, "C" in Figure 1, then this metadata is considered the most relevant and, hence, the most valuable. For the document sets that include each subject exclusively, "A" and "B" in Figure 1, only the metadata that intersect are considered relevant: anything not contained in the intersection is discarded as irrelevant.

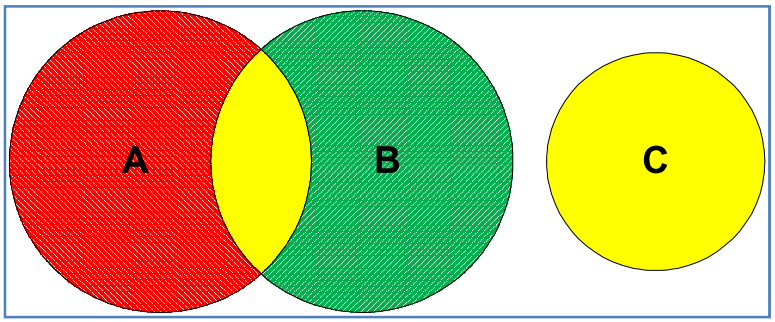

Figure 1: Document Sets Intersecting

Thus, Plinkr compiles metadata from these document sets that capture what, if any, relation exists between the subjects being queried. By exposing only the metadata perceived as relevant, i.e., the union of C with the intersection of A and B, we effectively reduce the amount of data presented and provide a means for a researcher to isolate key relationships between two subjects.

3.3.2 Entities

Any text-based document is likely to contain references to known or named entities, such as people, organizations, products, industry terms, and places. It is difficult, if not impossible, to convey information about some subject without making such references.

Another fundamental concept revolves around the notion that entities are highly relevant pieces of information and that any entities referenced within a document provide some abstract representation of the information being conveyed. Further, we assume that the more frequently an entity is referenced, the more relevant it is as a piece of information. By discovering the most relevant entities within a document or set of documents, we provide a framework for presenting a summary of the information being conveyed.

It is important to note that the subjects being researched are themselves assumed to be entities and, in fact, are treated as the most relevant entities of all.

3.3.3 Snippets of Text

When parsing textual content for information, some words and sentences are bound to be more relevant than others. For instance, when reading about a particular person, a researcher might highlight some small percentage of the words in a document, most likely in groups of contiguous words. We will refer to these groups of words as “snippets,” since they may or may not constitute a sentence or may include several sentences.

As mentioned in the previous section, we assume that if a particular entity is referenced frequently in a document or in a set of documents, then it is treated as more relevant than less frequently referenced entities. We extend this concept by assuming that any words in close proximity to a relevant entity constitute an equally relevant piece of information in the form of a snippet of text. Further, when two relevant entities are within some relatively small distance of one another, we assume that the words between these two entities are also relevant.

Plinkr was developed to provide an efficient means to quickly identify those snippets of text that pertain to the most relevant entities and, thus, those that a researcher will ultimately find most valuable.

4 Architecture and Design

4.1 Overview

Plinkr relies on traditional Web search to define a corpus of documents but extends the search engine results page by adding a layer of semantic metadata which can be more effectively analyzed and refined. In this regard, Plinkr can be thought of as a hybrid approach to search, bridging the gap between the current state of the art and the emerging semantic search technologies.

The application consists of five main components as illustrated in Figure 2 and detailed in the following sections.

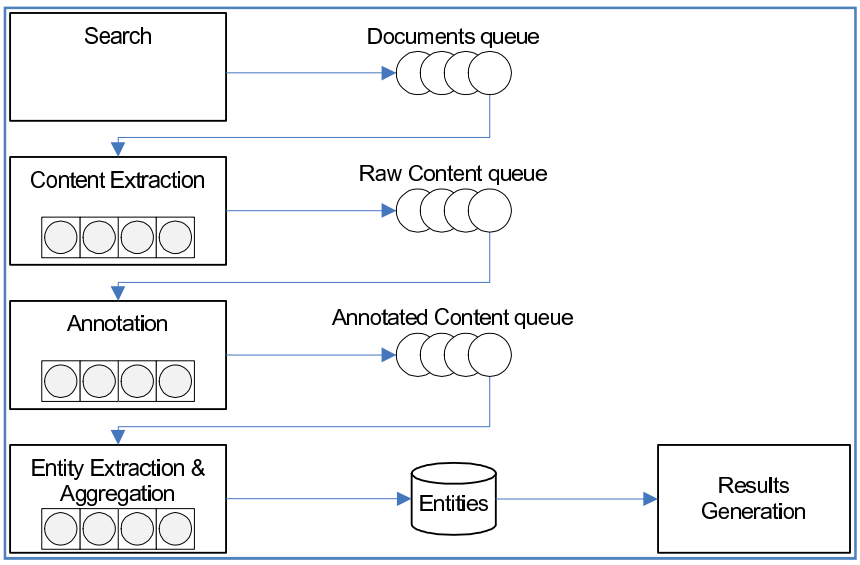

Figure 2: Components Overview

Each component provides some input for another and consequently there is a sequential progression once a query is initiated.

The components can be viewed as a series of transformations that get applied to the individual documents. These transformations take place at different speeds depending on the size and complexity of the document being processed. To avoid the obvious bottlenecks that would occur with a synchronous system, the Content Extraction, Annotation, and Entity Extraction and Aggregation components have been designed as thread pools, each being fed by a task queue and subsequently adding to a task queue that will feed some downstream component. A document transformation therefore becomes the smallest unit of work and this design prevents any single transformation from stalling the process as a whole.

Performance is a significant challenge due the potentially large number of documents available as raw data, for any given search, as well as the overhead associated with each of the individual transformations. While the sequential progression is necessarily retained, the architecture described

here allows for asynchronous processing and a certain degree of parallelism which greatly improves performance, robustness, and scalability.

4.2 Search

Search is the first component activated once a user query is submitted and no subsequent processing can take place until the first document is added to the documents queue. The diagram in Figure 3 provides a high-level view of what takes place within this component.

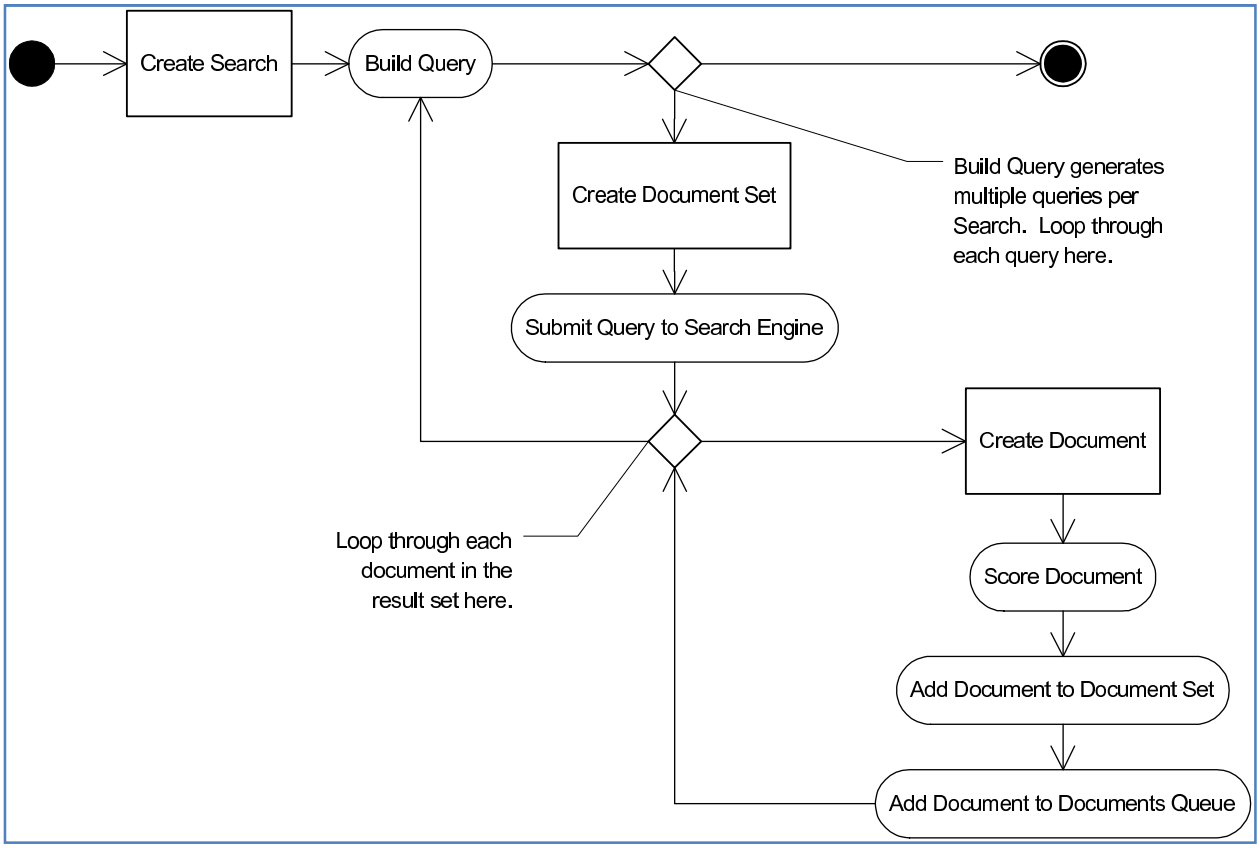

Figure 3: Search Activity Diagram

Action states in rectangular boxes like *Create Search* and *Create Document* indicate the instantiation, and possibly persistence, of an object. Other action states such as *Build Query* and *Score Document* will be discussed in more detail in the following sections. Diamond shaped boxes indicate a decision state

and in this particular diagram are used to denote loops – the decision is to repeat some series of actions or to return to another state.

The following class diagram includes the classes of objects that are relevant to this component and serves to illustrate the data, denoted as class attributes, acquired during this process:

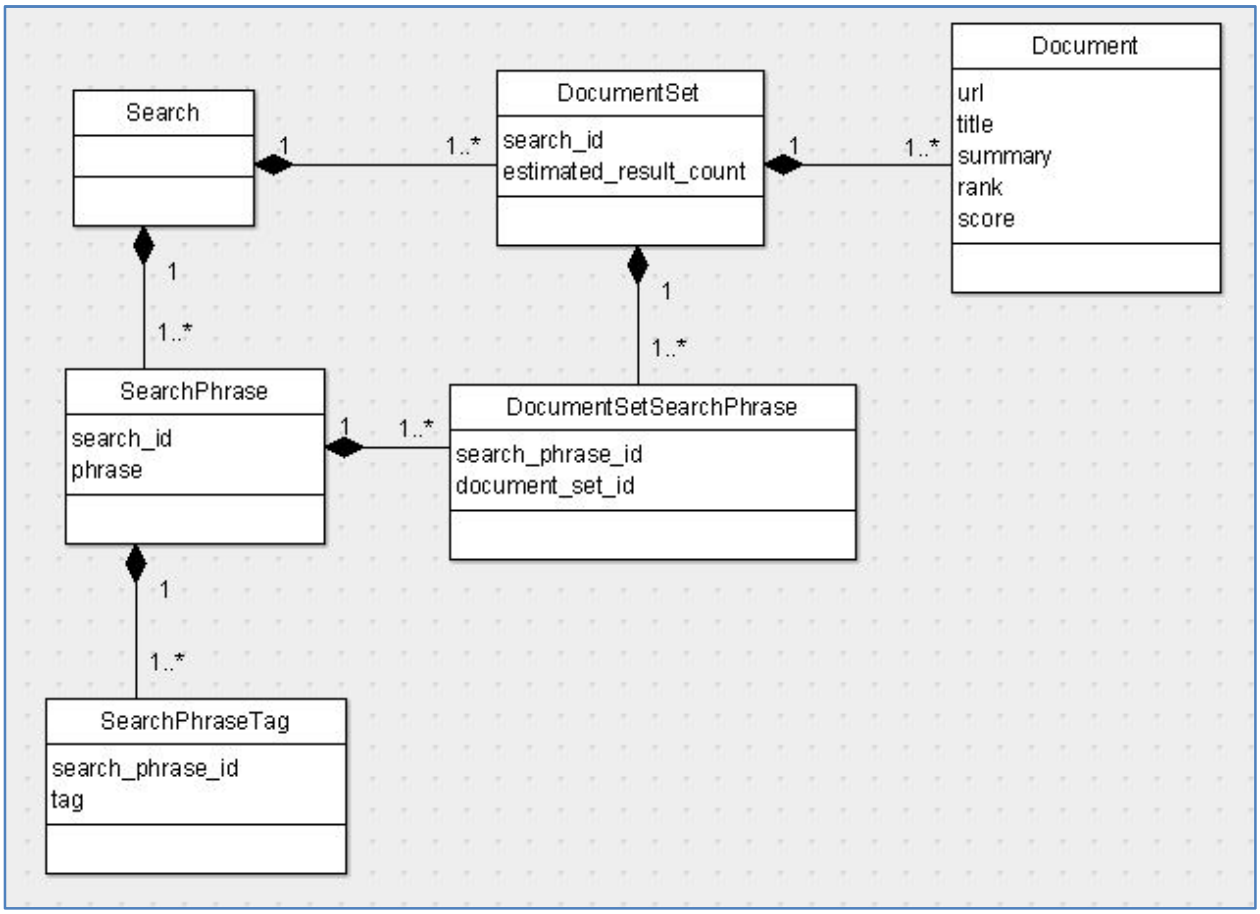

Figure 4: Search Class Diagram

Data of particular note are the document set's estimated result count and the document's rank and score. The details of acquiring these attributes are discussed in depth in the following sections.

4.2.1 Build Query

In order to provide data for the three document sets described earlier, three distinct search queries are constructed from the two user-submitted search phrases and any associated keywords:

1. query1 = phrase1 AND tags1 AND phrase2 AND tags2
2. query2 = phrase1 AND tags1 AND NOT phrase2
3. query3 = phrase2 AND tags2 AND NOT phrase1

Since they contain both phrase1 and phrase2, the most valuable data will come from documents returned by query1. But it possible that query1 will not produce any documents therefore necessitating query2 and query3 which provide data specifically relevant to each phrase exclusive of the other. If any keywords were provided, they are appended to the appropriate queries and serve to narrow the scope of the search.

As an example, given the user specified search parameters in Table 1,

Search Phrase	Keywords
Richard Stallman	Free software, GNU
Linus Torvald	Linux

Table 1: Query Parameters Submitted to Plinkr

We would construct the query strings listed in Table 2.

	Google Query
1	“Richard Stallman” “Linus Torvald” “free software” OR “gnu” OR “Linux”
2	“Richard Stallman” “free software” OR “gnu” –“Linus Torvald”
3	“Linus Torvald” Linux –“Richard Stallman”

Table 2: Query Strings Submitted to Google

4.2.2 Submit Query to Search Engine

The three queries are individually submitted to the search engine which returns three result sets of documents. Each of these result sets corresponds to a unique document set object.

We have elected to use the Google Search API (Google: Google AJAX Search API) as the search engine but any search engine with an adequate API would suffice. Of course, the quality of the results retuned is directly proportional to the quality of the results we ultimately present to the user. Multiple search engines could be used concurrently but this was not pursued because of the additional complexity it would create for a seemingly minimal return.

4.2.3 Google Search API

Google's API returns a maximum of 64 results per query which is a limiting factor in our design. These results include document URLs, titles, and summaries. In addition, some information about the search as whole is returned such as the estimated number of documents that match the query. A sample of the JavaScript Object Notation (JSON) formatted results provided by Google can be seen in Appendix A.

The data we explicitly capture from the Google results with a document object is the URL, the title, and the summary. The URL is needed by the content extraction component to access to the raw text content and also to provide the user with means to view the source Web page. The title and summary are preserved for display purposes.

For the document set object, we capture just the estimated result count which indicates the "estimated number of results that match the current query" (Google: Google AJAX Search API). This number is used in calculating a score for each document as described in the next section.

As discussed in the Background section, the order in which the search results are presented is based on the SERP rank of the individual Web pages. This is valuable data in terms of ascertaining the relevance

of information within that document. We make use of this by assigning a rank to each document beginning with 1 for the first document returned within each document set.

4.2.4 Score Document

Each document is assigned a score that reflects its relative importance within a document set. The document score is based on its rank, **r**, as determined from the order of results provided by Google. A higher document score indicates a more relevant document and so we use the inverse of **r** as follows:

$$score = \frac{1}{r}$$

The document score is used to calculate an entity score in a subsequent component and so we wish to normalize the rank in order to reduce its impact. Using min-max normalization the equation then becomes:

$$score = \left(\frac{1}{\frac{r - min}{max - min} * (new_max - new_min) + new_min} \right)$$

Where **min** is 1, the lowest rank value; **max** is a variable representing the highest rank value; **new_min** is 1 and **new_max** is 10. This ensures a document score between 0.1 and 1.0.

4.3 Content Extraction

As illustrated in Figure 5, the content extraction component takes document objects from the documents queue, retrieves the content associated with the URL attribute and adds the resulting document content objects to the raw content queue.

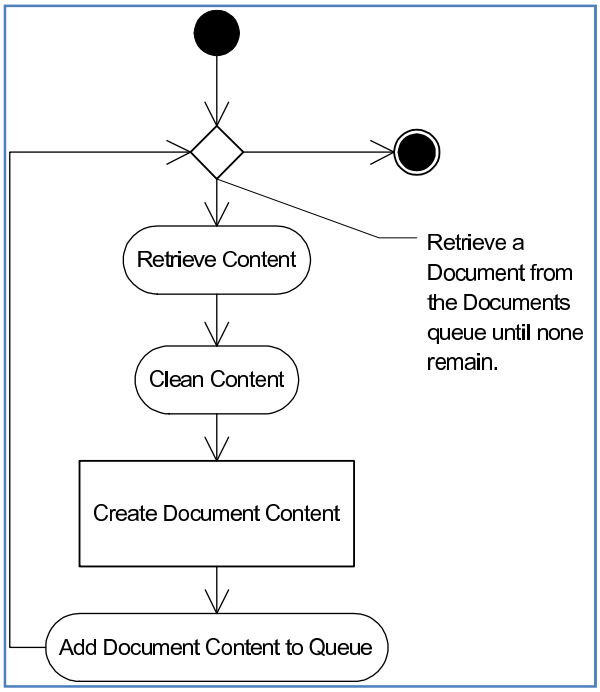

Figure 5: Content Extraction Activity Diagram

The raw content is retrieved from a Web server and then goes through various cleaning operations to remove unwanted HTML code, special characters, and whitespace. This process is commonly referred to as "Web scraping" and presents few challenges. However, since the source pages are completely heterogeneous, the process must be general purpose and cannot rely on any formatting cues that might allow for more precise extraction.

4.4 Annotation

The annotation component transforms the raw unstructured content into semantic metadata in RDF format and is a cornerstone of the application because it adds structure and meaning to what would otherwise be a string of characters.

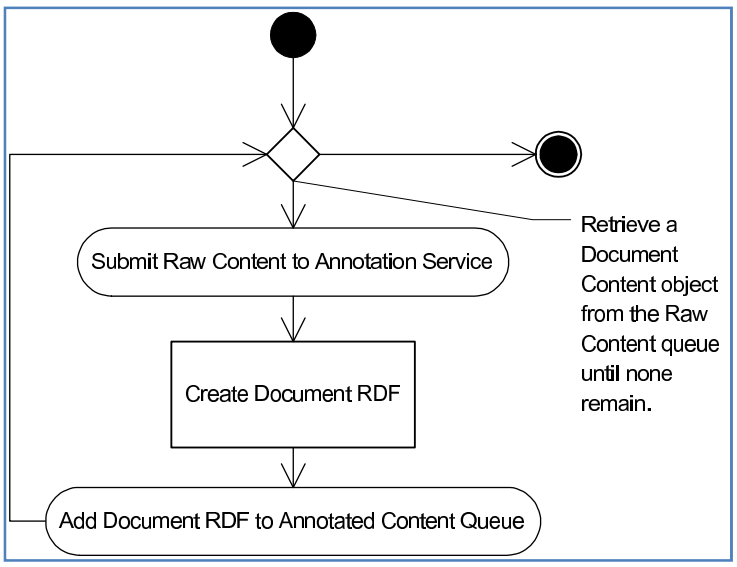

Figure 6: Annotation Activity Diagram

The process of annotating the raw content is a complex problem that is beyond the scope of this project. Consequently, for this component we rely on a third party Web service as detailed in the following section.

4.4.1 Calais Web Service

The Calais Web Service (Calais: Frequently Asked Questions) analyzes unstructured text, such as that found in Web pages, and returns semantic metadata. This is accomplished by using natural language processing and machine learning techniques to identify entities, facts and events in the text. This metadata is extracted from the text and returned in RDF format. A sample of RDF formatted results along with other details about the data provided by the Calais Web Service is listed in the Appendix B.

Each unique entity discovered is included as resource in the RDF. The name of the entity is provided along with its type. The types of entities identified by the Calais Web Service include Person, Organization, City, and Product. When resolving the identity of an entity, Calais employs disambiguation for limited entity types such as Company, Geographical, and Product. For example, references to “IBM” and “International Business Machines” would resolve to the same unique entity. This is particularly

valuable to our analysis since the frequency with which an entity is referenced is an important parameter and disambiguation serves consolidates entities with different names that actually refer to the same thing.

Each unique entity is associated with one or more instances of occurrence within the text. Anyplace the entity is referenced, whether directly by name or indirectly by pronoun, is considered an instance. Each instance is included as a resource in the RDF and contains the actual detection – or snippet of text the entity is referenced within – along with offset and length values so the entity can be located in the source text. This data is used extensively in the Results Generation component during Snippet Extraction.

We are also provided with a relevance score for each entity which is included in the RDF as individual resources. This score represents the relative importance of the entity in the context of the document being processed and is another important parameter in our analysis as described in the following section.

Finally, Calais provides a document categorization which we use as part of the statistical analysis of the corpus of all documents. This is a limited taxonomy used to identify what a document is about in a general sense. Examples of such categories include Politics, Sports, and Business Finance.

4.5 Entity Extraction and Aggregation

The annotation component discussed in the previous section is a foundation of the application because it supplies the essential metadata. But the entity extraction and aggregation component is a key innovation in the sense that it determines what information is representational of the document set as a whole. This component takes RDF as its input and returns aggregated and scored entities, referred to as document set entities, for each document set as illustrated in Figure 7.

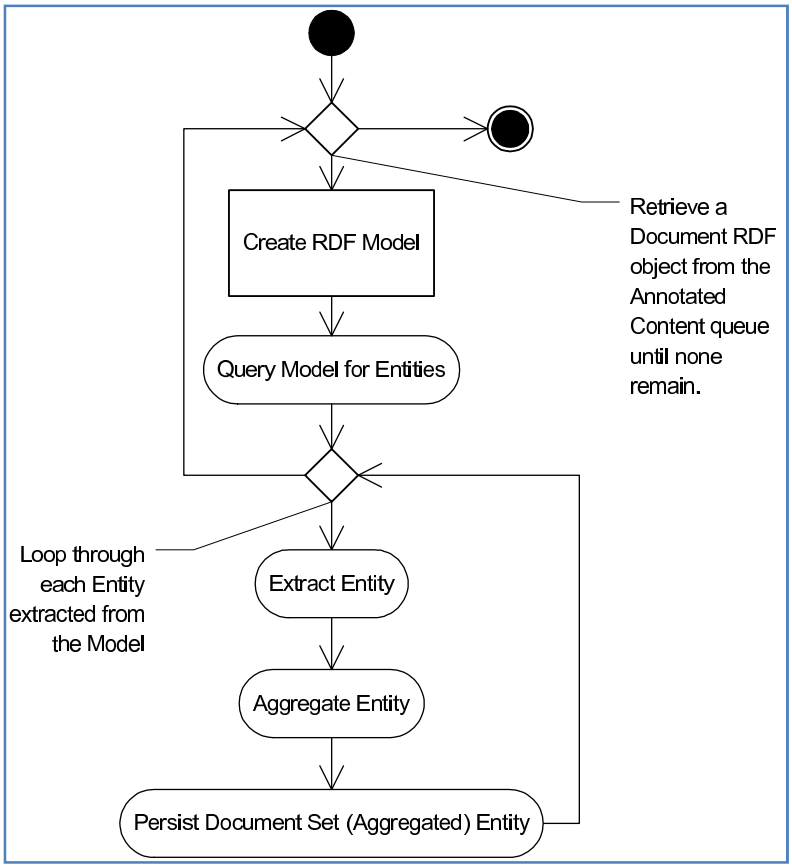

Figure 7: Entity Extraction and Aggregation Activity Diagram

The following class diagram includes the classes of objects that are relevant to this component:

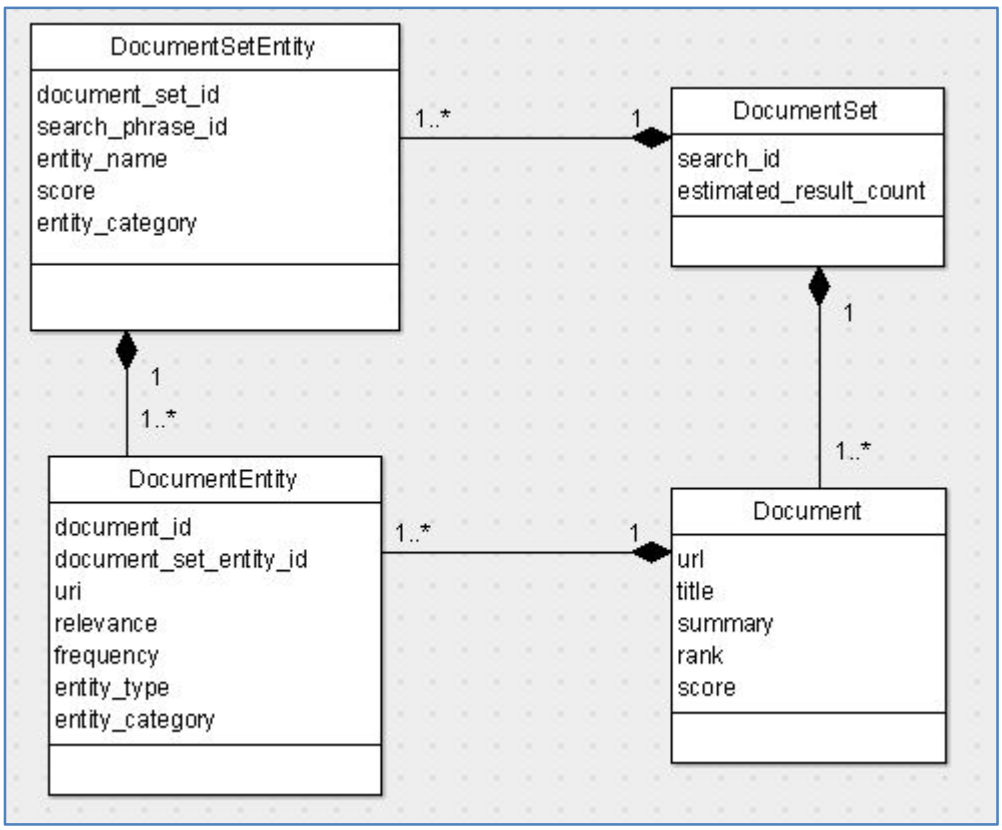

Figure 8: Entity Extraction and Aggregation Class Diagram

A document entity's relevance, entity type, and frequency are all provided by the RDF of a particular document. A document set entity represents the aggregation of one or more document entities. These objects have a score and entity category which are both used in determining which entities get displayed to the user. The process of assigning values to these attributes is discussed in the following sections.

4.5.1 Jena

In order to facilitate the extraction of data from RDF we use Jena (Jena: A Semantic Web Framework for Java), an open source semantic Web framework that provides an RDF API along with a SPARQL query engine. SPARQL is an RDF query language. Internally, Jena models an RDF graph as a set of statements where each statement asserts a fact about a resource.

4.5.2 Extract Entity

Given the RDF for a particular document, we use Jena to create a model which can then be queried to obtain a list of distinct entities. Additional queries are used to obtain the Calais generated relevance scores and the number of times an entity was referenced, its frequency, within the document. We also assign each entity a category based on its Calais defined type. We have elected to limit this more general categorization to "person", "place" or "thing". See Appendix B for the type to category mappings used. This combined data describes each distinct entity and is encapsulated in a document entity object.

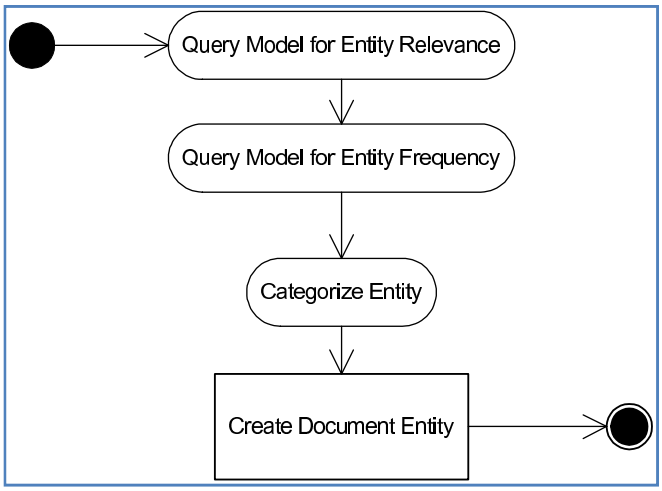

Figure 9: Entity Extraction Activity Diagram

Figure 9 illustrates the process of extracting a particular entity from the model. This process is repeated for each unique entity found in each document.

4.5.3 Aggregate Entity

As the individual document entities are extracted from the RDF, they are simultaneously aggregated at the document set level as represented by document set entities. Given the complexity of the disambiguation problem, we use the naïve approach of aggregation based on entity name as the unique

identifier. As mentioned previously, the Calais Web Service does provide disambiguation for certain entity types and we take advantage of that here when applicable.

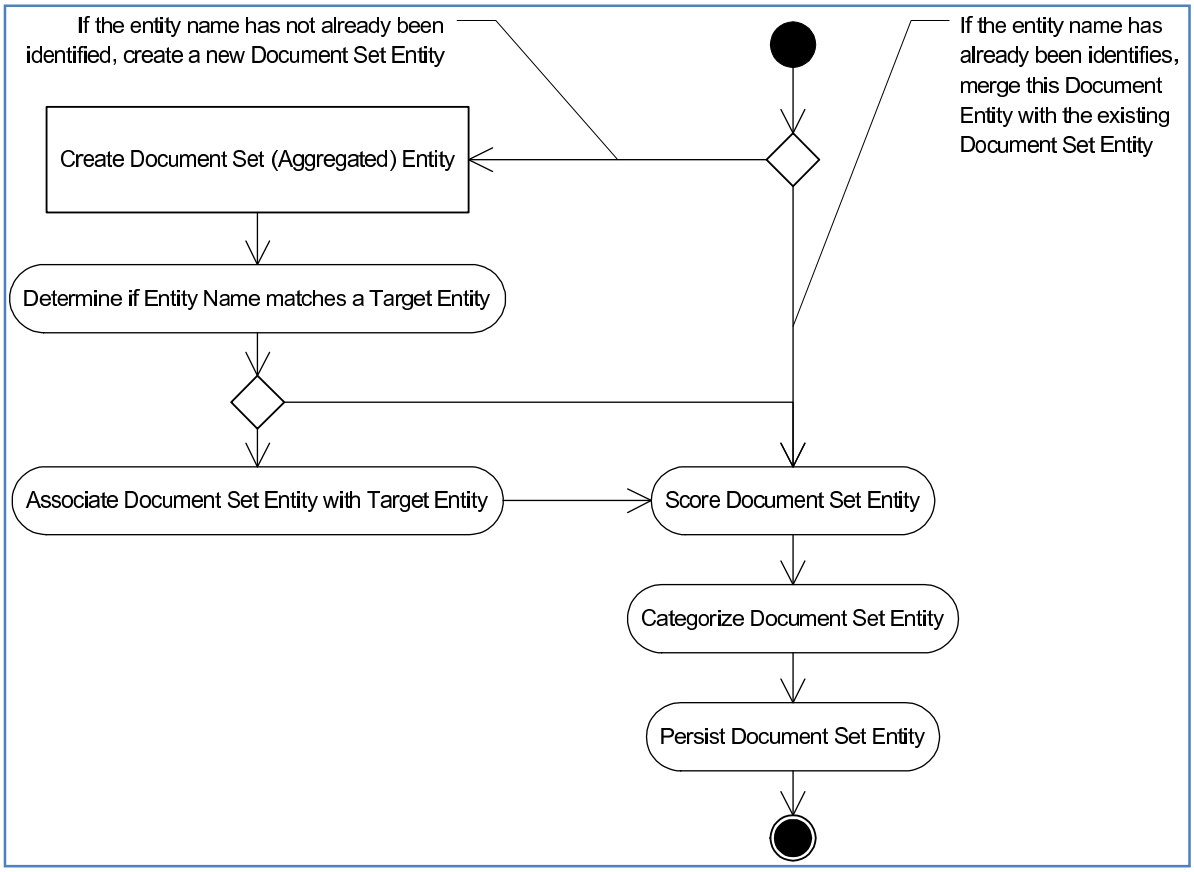

Figure 10: Entity Aggregation Activity Diagram

As each new document entity is created, we use the entity name to determine if a corresponding document set entity exists. If it does not, a new document set entity is created; otherwise the document entity data is merged with the document set entity data and the score is recalculated. The details of calculating the score are outlined in the next section.

When a document set entity is created, we determine if it is a reference to one of the target entities, i.e., one of the subjects being searched. Because we'd rather present too much data to the user than miss an important piece of information, it is preferable to incorrectly identify an entity as a being a

target than miss an actual match. In order to compensate for various usages of the target entity names we relax the tolerance and use regular expression matching. For example, we would want "Hillary Rodham Clinton" and "Hillary Clinton" to be considered a match and we effectively accomplish this.

4.5.4 Score Entity

In order to ascertain the relevance of an entity within a document set, each document set entity is assigned a score as follows:

$$score = avg(r) * avg(d) * f_n$$

The Calais Web Service provides a value, between 0 and 1, that represents the relevance, **r**, of a document entity within a document. Since a document set entity is composed of multiple document entities, we use the average value of **r**. Similarly, we use the average document score, **d**, to account for the relevance of the source document from which each document entity was obtained. Finally, we factor in the number of distinct documents, or document frequency, **f**, that the document entities were found in. In order to prevent **f** from out weighing the other factors, we use min-max normalization to obtain $\mathbf{f_n}$, a value between 0.1 and 1.0.

4.6 Results Generation

As with the previous components discussed, the results generation component runs continuously as the search progresses. But rather than working from a task queue, it generates real time results based on the current state of processing. A more comprehensive approach might be to simply generate the results once after all processing has completed. However, since processing can take a substantial amount of time (see the Evaluation subsection in the Conclusion and Future Work section), intermittently generating and displaying the current results serves to keep the user engaged.

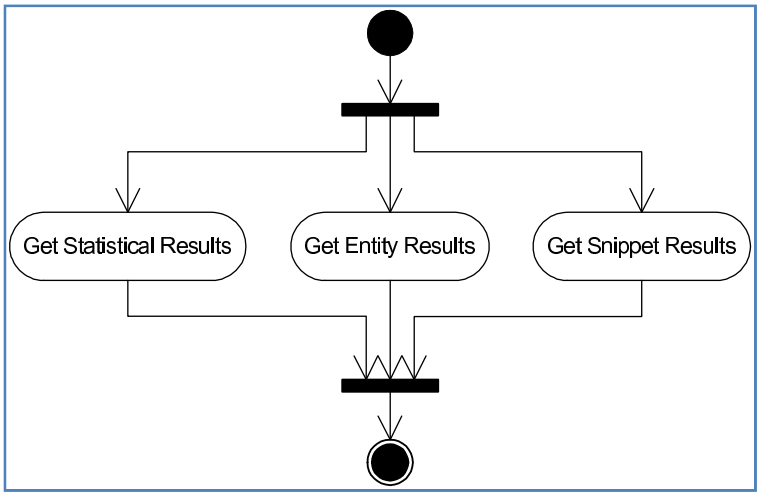

Figure 11: Results Generation Activity Diagram

As illustrated in Figure 11, the results generation component consists of three activities that run concurrently.

This is an important component overall because it determines which information gets displayed to the user. While the document set entity scores are important factors in the selection process this component contains several key innovations as discussed in detail in the following sections.

4.6.1 Get Statistical Results

The Calais Web Service assigns each document a category, as discussed earlier. Since this may reveal useful information about the relationship between the two subjects being researched, the document categories are aggregated and the top few are displayed by frequency. In addition, we provide general statistical data about the search such as the estimated result count for each document set.

4.6.2 Get Entity Results

The main results presented to the user come in the form of entities and snippets. The entities represent an abstraction of the data as a whole and provide a means to filter the snippets of text. Consequently, it is important to display only the most relevant entities and to ensure an even distribution of entities in

each of three general categories - person, place, and thing. To accomplish this, we perform distinct queries for each category as illustrated in Figure 12.

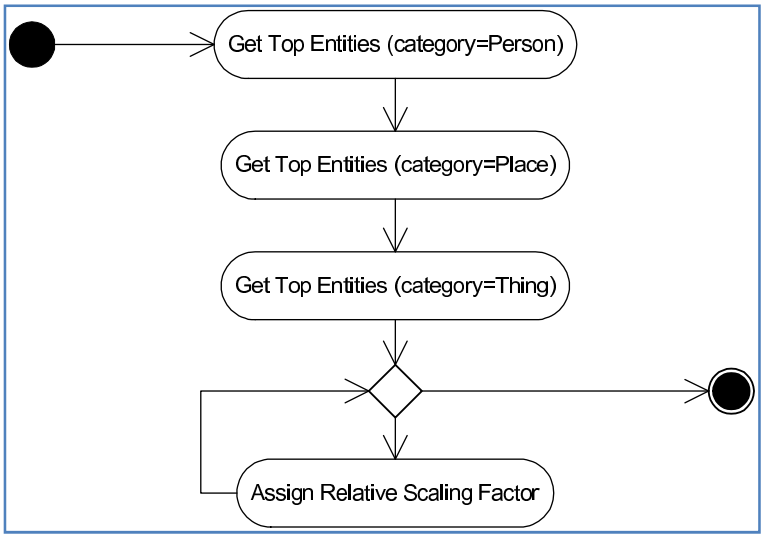

Figure 12: Get Entity Results Activity Diagram

To determine which entities should be selected, we rely heavily on the document set entity score, the calculation of which was discussed earlier. This score takes into account factors such as the entity's relevance and frequency of occurrence, as well as the source document's SERP rank.

Since the goal is to present the intersection of data, we only include entities that meet this criterion. Recalling that the three document sets are defined as follows:

1. documentSet1 = results from query1 = phrase1 AND phrase2
2. documentSet2 = results from query2 = phrase1 AND NOT phrase2
3. documentSet3 = results from query3 = phrase2 AND NOT phrase1

We therefore include all entities in documentSet1 along with any entities that exist in both documentSet2 and documentSet3.

Intuitively it makes sense to increase the scores of entities contained in all three document sets above those contained in just documentSet1. Similarly, the scores of entities contained in documentSet1 as well as documentSet2 or documentSet3 should be increased over those contained just within documentSet2 and documentSet3. To accomplish this we calculate a score that represents the relevance of an entity within a search as a whole as follows:

$$score = avg(e) * s_n$$

Where **e** is the document set entity score and $\mathbf{s_n}$ is a normalized term that accounts for the number of document sets the entity was found in. With this subset of entities thus defined, we then simply select those with the highest scores.

Finally, since we want to visually convey the relative importance of each entity, as determined by its score, we calculate a scaling factor. This is a discrete number that corresponds to a small, medium, large, or extra-large display size.

4.6.3 Get Snippet Results

While the entities present an abstract view of the information, snippets are actual excerpts from the source documents and are likely to be the most valuable information from a researchers' perspective. Initially, we retrieve the snippets that make some reference to either, or ideally both, of the two subjects in question. We refer to the two subjects being researched as the target entities. The process used to extracting these snippets is outlined in Figure 13:

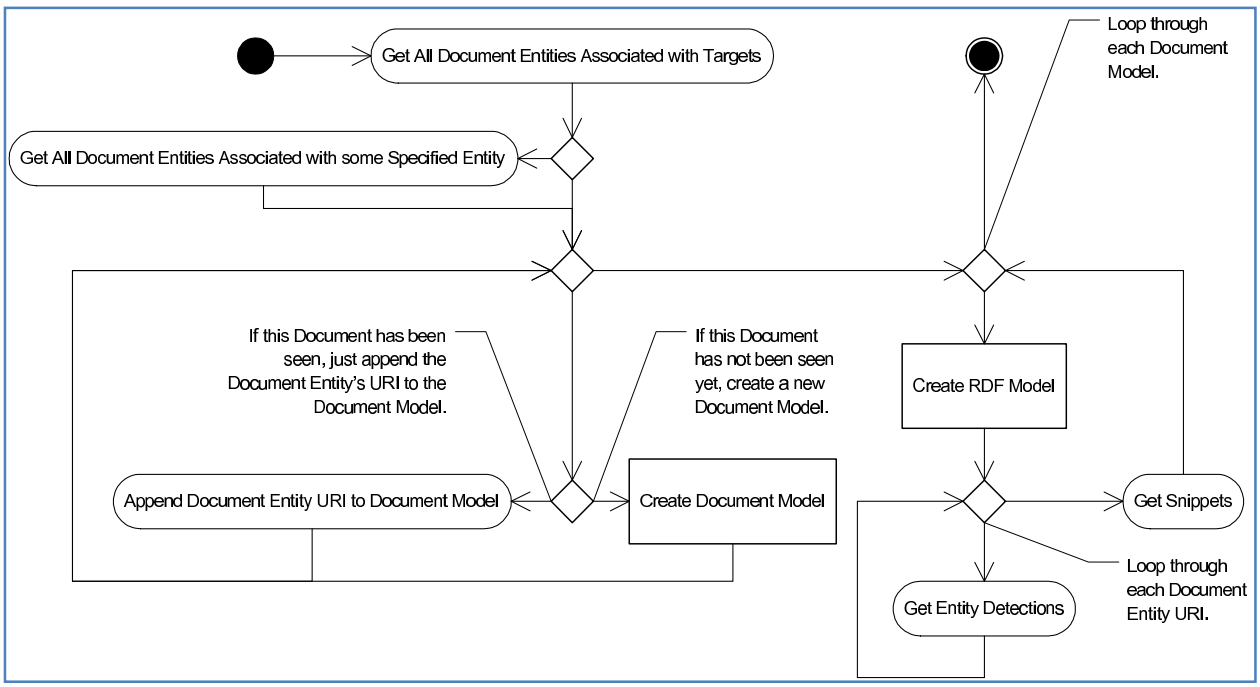

Figure 13: Snippet Extraction Activity Diagram

Recall that during the entity aggregation process multiple document level entities are aggregated at the document set level. The end result is that each document set entity refers to one or more document entities. Furthermore, each document set entity has been analyzed to determine if it matches one of the target entities. Snippet extraction begins with finding all document entities associated with document set entities that match either of the target entities. Each of these will be represented by a unique resource, identified by a unique Uniform Resource Identifier (URI), in some document's RDF.

From this collection of document entities, we construct a list of document models. A document model encapsulates a document along with all of the document entity URIs of interest. For each of these document models, we use Jena to create an RDF model which can be queried to retrieve all "detections", or instances, of a particular entity. The detections include the reference to the entity itself along with some portion of the containing text. In this manner we build a list of all detections of the target entities within a particular document.

Since the detections may overlap one another, we then stitch appropriate detections together to form snippets. The snippets are scored based on the scores of all entities contained by that snippet as well as the number of distinct entities referenced. A snippet referring to both target entities is thus determined to have more value than a snippet referring to just one.

Another case of this same process occurs when the user wishes to see snippets filtered by some non-target entity. This allows the researcher to explore information pertaining to the target entities in conjunction with some third entity. In this case the same process takes place with the addition of these document entities. The scoring also remains the same with a snippet referring to both target entities and the selected entity having most value and snippets referring to only the selected entity having the least value.

5 User Interface

5.1 Query Form

The user interface consists simply of a query form and a results visualization page. The user enters the names of two entities (people, places, things, etc.) along with any keywords or phrases that further qualify the subjects being searched. While these keywords are optional, they can greatly improve the accuracy of results. The only requirement is that two names are present.

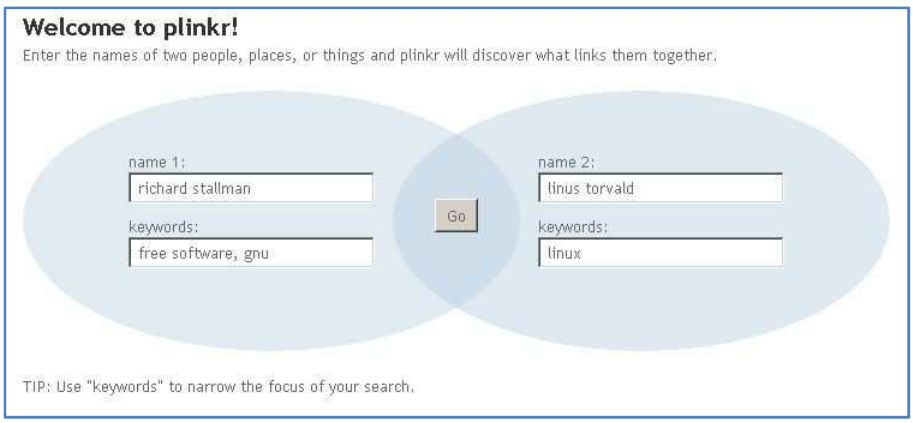

Figure 14: Query Form

Once validated, the query is submitted and processing begins.

5.2 Results Visualization Page

The user is brought to the results visualization page and some details of the search are immediately displayed including the query itself and the number of documents to be processed.

Figure 15: Results Visualization Page

Since processing can take some time to complete, results are displayed as they are received and the status of the search is continually updated.

5.2.1 Statistics

The statistics panel displays the estimated total documents available for each of the three queries. As discussed earlier, this may be helpful to the user as an indication of the overall quality of the search. A very large number of estimated documents for a particular subject may indicate that the search is too general and more keywords should be considered. Conversely a low number or a result of zero would indicate a lack of data for that particular query. This panel also lists the most relevant document categories which serve to describe the data as a whole.

5.2.2 Entity Cloud

The main panel of the results page is a "cloud", or weighted list of words, which lists the names of the most relevant entities discovered. These names are color coded to indicate how they have been generally categorized - person, place or thing - and the user has the option to show or hide any combination of the entities by category. The names are sized according to their relevance score with larger names indicating a higher relevance. Finally, the names are sorted from most to least relevant.

The entity cloud presents an abstraction of the information obtained – a list of entities the two subjects have in common - and as such is useful on its own. The entity cloud is also provides a way for the user to interact with the data. By clicking on a particular entity, the snippet results can be filtered to those containing references to that entity.

5.2.3 Snippets

The snippets panel lists pieces of text extracted from the documents along with a link to the actual source Web page. The initial listing shows snippets that were found to contain references to both subjects being searched. If none exist, the highest ranking snippets containing either subject are listed.

Snippets can be filtered by clicking on any entity listed in the entity cloud. This will display any snippets containing the selected entity as well as one or both subjects, when such snippets exist. If none do exist, snippets containing just the selected entity are displayed.

In either case, snippets are listed by score with those scoring highest at the top. All snippets will contain at least one entity, either a target or a selected filter, which is highlighted. In cases where the highlighted text is a reference to a named entity, such as with the pronoun "she", rolling the mouse over the text will reveal the actual name of the entity.

6 Implementation Details

6.1 Platform

Plinkr (http://www.plinkr.com) is a Web application based on the Ruby on Rails framework and written in JRuby and Java. The Ruby on Rails framework was chosen primarily for the way it facilitates rapid development and in particular for its object-relational mapping system which greatly simplifies database interaction. Additionally, there is a vast Ruby/JRuby library that makes working with Web technologies easier. The application leverages various existing technologies, in particular the Jena, the open source semantic Web framework, which made Java a requirement. JRuby was chosen over Ruby for its ability to seamlessly integrate Java applications. Finally, MySQL was chosen as the relational database management system.

6.2 Model

Ruby on Rails implements the Model-View-Controller architectural pattern. The following class diagram documents the applications' model.

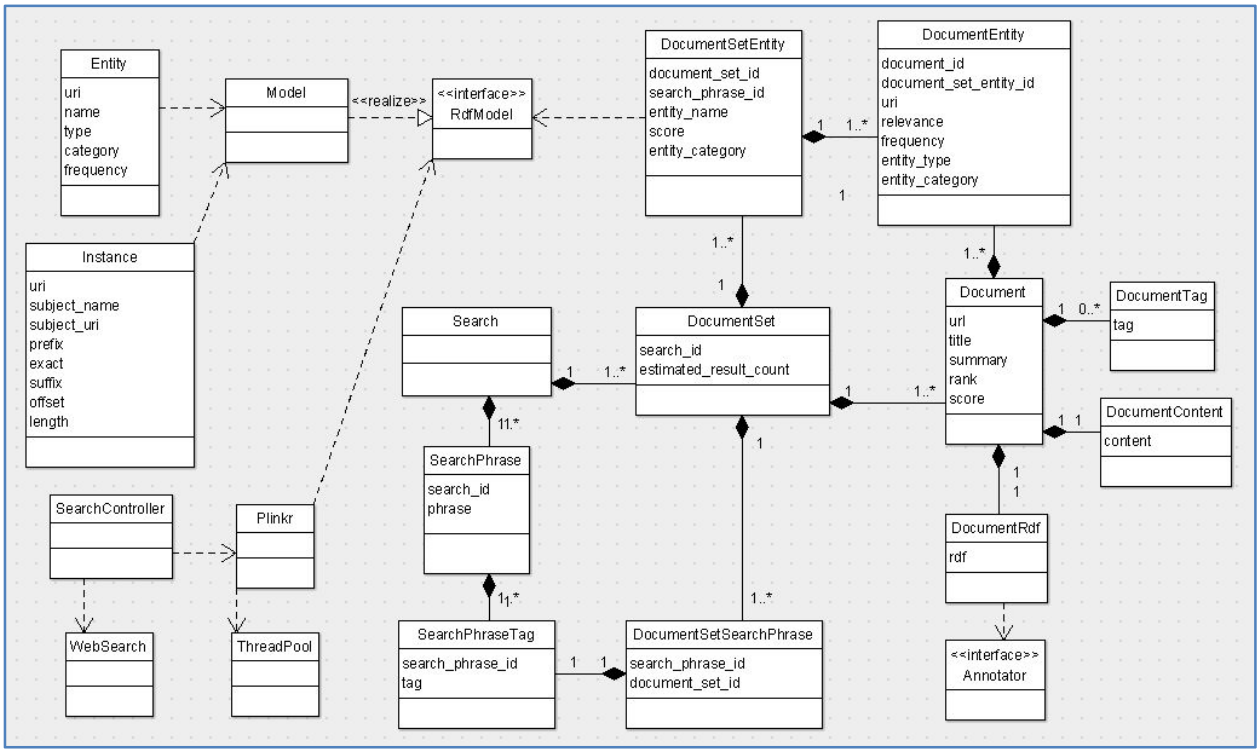

Figure 16: Class Diagram

6.3 Entity Relationship Diagram

The following Entity Relationship Diagram (ERD) documents the structure of the database.

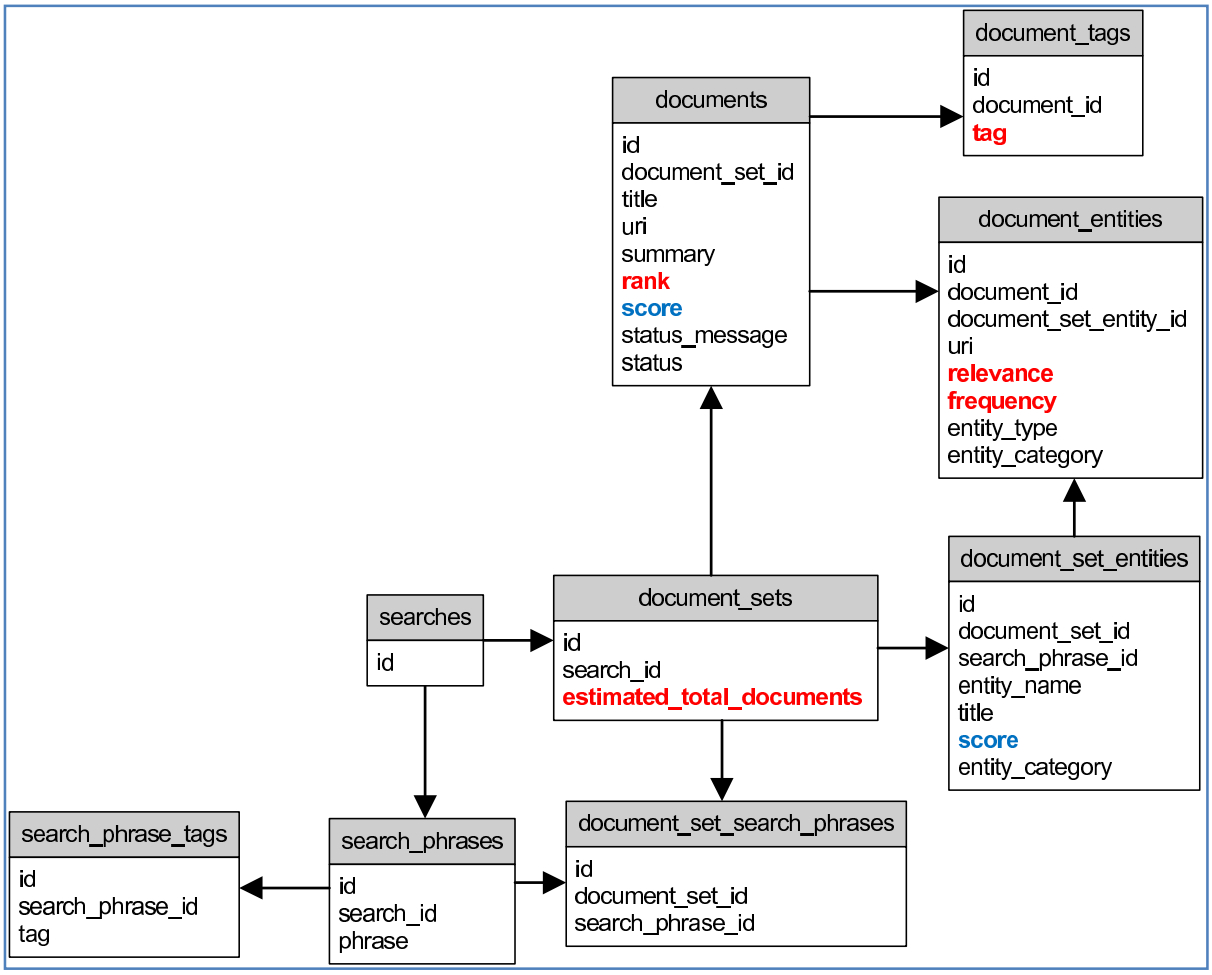

Figure 17: Entity Relationship Diagram

6.4 Adjustable Runtime Parameters

There are several runtime parameters that can be adjusted to fine-tune performance and consequently the quality of the user experience. One of these is the size of the various thread pools. Increasing the number of threads in a pool increases the number tasks that can simultaneously be processed but this also consumes more memory.

Another significant parameter is the maximum number of documents that will be processed. It would not be practical to analyze every document available, which could number in the millions. Furthermore, the Google API limits the number of results to 64 for any given query but even this relatively small

number of documents is somewhat beyond the means of Plinkr in its current implementation. It is assumed that the effectiveness of Google's PageRank ensures that the top few documents in each document set will provide the most relevant data and so we currently limit the number of documents per search to 12, for a total of 36, which typically takes under a minute to analyze. It is our hope that this number can be increased as development continues.

6.5 Deployment Details

Plinkr is deployed as a WAR file on the Apache Tomcat servlet container in conjunction with the Apache Web server. The application is run in on hosted virtual private server.

7 Conclusion

7.1 Evaluation

Evaluating the usefulness of Plinkr is a somewhat subjective task. Assuming that a search engine is the primary Web-based tool used today to research two subjects, we evaluate the success of Plinkr by comparing its results to those provided by a standard Google search. Figure 18 compares portions of the results pages on Plinkr and Google for a search of Bill Gates and Barack Obama.

Figure 18: Google and Plinkr Results Comparison

The top Google result is a link to a Gizmodo.com story about campaign donations. Included with that result is the snippet "William Gates has only made one presidential-candidate campaign donation this season, and it was to Barack Obama". This same snippet was found by Plinkr and included in the top results along with several other snippets from that same story. So from a snippets perspective, both searches may be considered to have equal utility. Since Plinkr does provide more information from that same story, the user might obtain what they were looking for without having to go to the source, or at least have a better sense of how useful going to that particular source page will be.

Google's results page offers only links to pages along with a single snippet from each page. Plinkr goes beyond this by providing an entity cloud that provides a high-level view of the various connections between Barack Obama and Bill Gates. For example, one highly ranked entity is Harvard University. This is potentially valuable information and nowhere in the first 10 pages of Google results does the word "Harvard" appear. Clicking the Harvard University entity in Plinkr produces the results displayed in Figure 19 below:

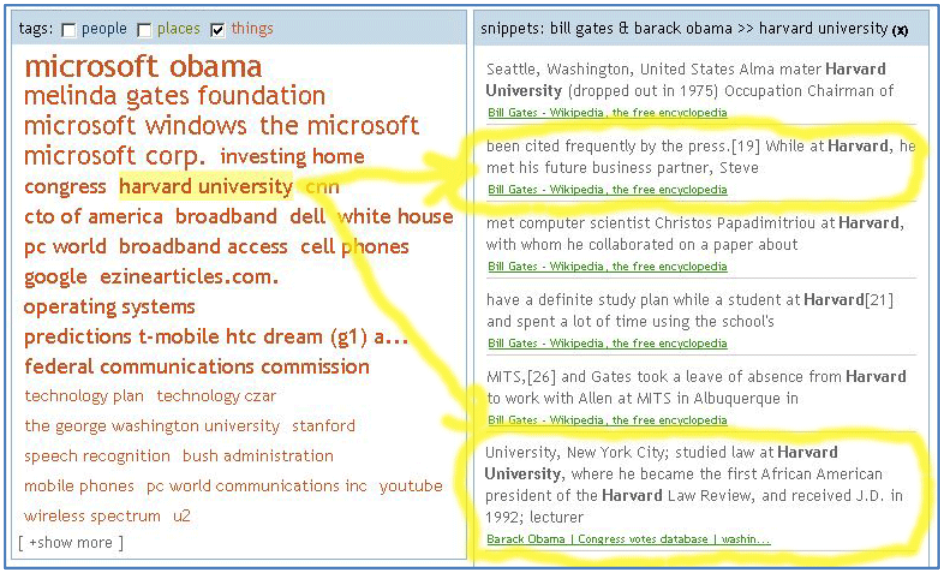

Figure 19: Plinkr Entity Cloud

While the resulting snippets do not explicit indicate that both Bill Gates and Barack Obama attended Harvard, they do suggest this possibility and clicking through to the source pages confirms as much. This aspect of Plinkr demonstrates its real potential.

7.2 Future Work

7.2.1 Performance

Performance remains a significant challenge. One way around this issue would be to provide results asynchronously by notifying the user when the results become available. This might appeal to some,

and would allow for the processing of larger document sets, but the average user expects results in real-time within seconds, not minutes. One approach to improving performance would involve pre-populating data for certain popular searches or individual subjects but this too has its limitation. Another approach would be to use a more distributed architecture and increase the degree of parallelism. It is also possible that continued refactoring of the existing code could achieve substantial gains. In any case, performance is likely an engineering problem that can be solved.

7.2.2 Results Quality

7.2.2.1 Entity Proximity

While performance is a significant issue in terms of the user experience, we are more immediately concerned with improving the quality of results. The current scoring of entities does not account for their proximity to the target entities within a document. It could be argued that entities within some distance of a target entity, or perhaps simply within the same sentence, have a higher relevance. Taking this into account when scoring entities would highlight entities that are statistically irrelevant but meaningful none the less.

7.2.2.2 Clustering

Another direction for future work would be to determine if the target entities could refer to more than one known entity and if so, prompt the user for more input. This could possibly be handled by clustering the documents and identifying metadata for each cluster. For example, a search for Paris Hilton might identify two clusters, one around the celebrity and another around the hotel.

7.2.2.3 Link Analysis

Plinkr currently only accounts for generic text data within a Web page. However, there is certainly information to be found in the HTML encoding of the page such as outbound links. An analysis of all

outbound links on all pages might serve to define new sources to explore or might reveal additional patterns.

7.2.2.4 Word Analysis

Plinkr has focused on entities as the primary vehicle for abstracting information. A simultaneous word analysis of the same documents could be used to identify statistically significant nouns or verbs. The WordNet lexical database could be used to this end.

7.2.2.5 Entity Profile

Finally, various known sources Web-based resources such as Wikipedia could be used to assemble a general profile about each subject being researched. While this might not directly reveal information about how the subjects are connected, it would provide another piece of contextual information that a researcher might find valuable.

7.3 Related Work

Despite the fact the Semantic Web is in its infancy, it is quite topical and numerous technologies based on its promise are rapidly being developed. While we are not aware of an application that has the same objective as Plinkr, there are many emerging applications Semantic Search.

7.3.1 Google

It has been reported that Google is developing semantic search technologies that will extend the existing keyword search algorithms (Perez). Very recently Google began displaying longer snippets of text on the results page that highlight the keywords in context.

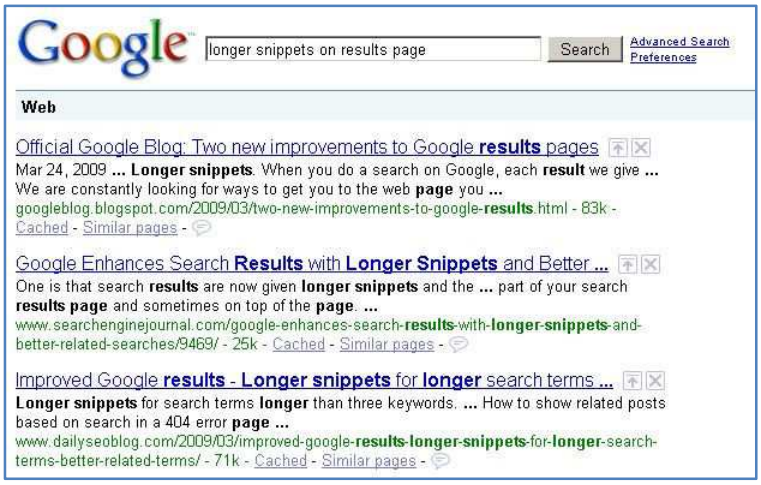

Figure 20: Screenshot from Google Results Page

Like Plinkr, this allows the user to obtain more information about the search without having to go to the source material.

7.3.2 Evri

Evri is a semantic search engine that is closely related to our research in the sense that Evri seeks to build a "map of connections between people, places, and things on the Web" (Evri: About Us). With Evri, a user can view a results page for a single subject from a pre-existing list of popular subjects.

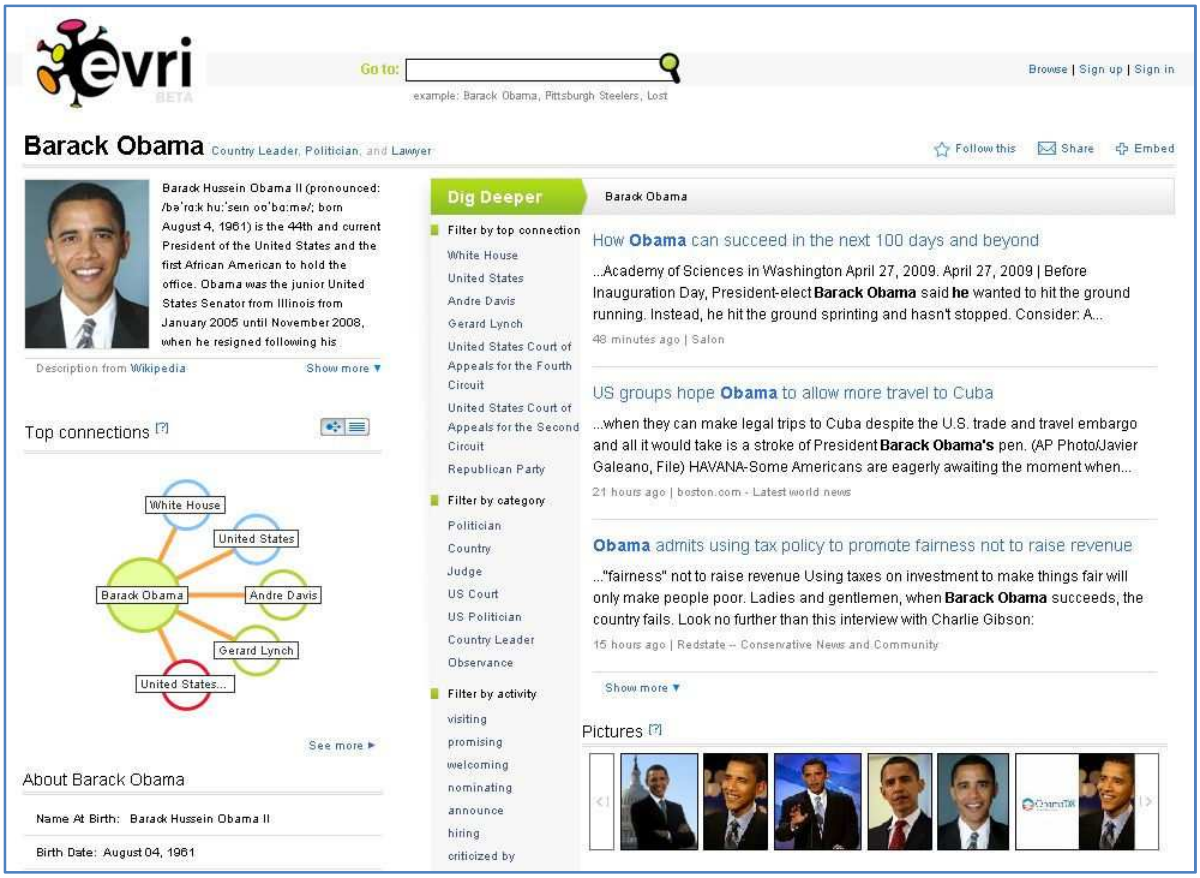

Figure 21: Screenshot from Evri results Page

While quite limited in terms of what can be researched, Evri does present relevant snippets of information along with the ability to filter these snippets by category, activity, etc. Evri also presents highly relevant "connections" which are essentially related entities that can be further explored.

7.3.3 Hakia

Hakia is another semantic search engine that focuses more on natural language queries. Hakia presents results that are similar to Google but seeks to find meaning in the user's query rather than simply perform a keyword search.

Figure 22: Screenshot from Hakia Results Page

8 Appendix

8.1 Appendix A: Google Search API

The following is a sample of a JSON formatted response returned by the Google Search

API:


```
{"responseData": {
 "results": [
  {
   "GsearchResultClass": "GwebSearch",
   "unescapedUrl": "http://en.wikipedia.org/wiki/Paris_Hilton",
   "url": "http://en.wikipedia.org/wiki/Paris_Hilton",
   "visibleUrl": "en.wikipedia.org",
   "cacheUrl":
"http://www.google.com/search?q\u003dcache:TwrPfhd22hYJ:en.wikipedia.org",
   "title": "\u003cb\u003eParis Hilton\u003c/b\u003e - Wikipedia, the free
encyclopedia",
   "titleNoFormatting": "Paris Hilton - Wikipedia, the free encyclopedia",
   "content": "\[1\] In 2006, she released her debut album..."
  },
  {
   "GsearchResultClass": "GwebSearch",
   "unescapedUrl": "http://www.imdb.com/name/nm0385296/",
   "url": "http://www.imdb.com/name/nm0385296/",
   "visibleUrl": "www.imdb.com",
   "cacheUrl":
"http://www.google.com/search?q\u003dcache:1i34KkqnsooJ:www.imdb.com",
   "title": "\u003cb\u003eParis Hilton\u003c/b\u003e",
   "titleNoFormatting": "Paris Hilton",
   "content": "Self: Zoolander. Socialite \u003cb\u003eParis
Hilton\u003c/b\u003e..."
  },
  ...
 ],
 "cursor": {
  "pages": [
   { "start": "0", "label": 1 },
   { "start": "4", "label": 2 },
   { "start": "8", "label": 3 },
   { "start": "12","label": 4 }
  ],
  "estimatedResultCount": "59600000",
  "currentPageIndex": 0,
  "moreResultsUrl":
"http://www.google.com/search?oe\u003dutf8\u0026ie\u003dutf8..."
 }
}
, "responseDetails": null, "responseStatus": 200}
```

8.2 Appendix B: Calais Web Service

8.2.1 Entity Types

The Calais Web Service currently supports the extraction of the following types of entities:

Anniversary, City, Company, Continent, Country, Currency, Email Address, Entertainment Award Event, Facility, Fax Number, Holiday, Industry Term, Market Index, Medical Condition, Medical Treatment, Movie, Music Album, Music Group, Natural Disaster, Natural Feature, Operating System, Organization, Person, Phone Number, Product, Programming Language, Province Or State, Published Medium, Radio Program, Radio Station, Region, Sports Event, Sports Game, Sports League, Technology, TV Show, TV Station, URL

8.2.2 Entity Type Categories

We map the Calais entity types to a more general category using the following rules:

Category	Entity Types
Person	Person
Place	City, Continent , Country, Province Or State, Region
Thing	Anniversary, Company, Currency, Email Address, Entertainment Award Event, Facility, Fax Number, Holiday, Industry Term, Market Index, Medical Condition, Medical Treatment, Movie, Music Album, Music Group, Natural Disaster, Natural Feature, Operating System, Organization, Phone Number, Product, Programming Language, Published Medium, Radio Program, Radio Station, Sports Event, Sports Game, Sports League, Technology, TV Show, TV Station, URL

Table 3: Categories of Entity Types

8.2.3 Sample RDF

Person Entity:

```
<rdf:Description rdf:about="http://d.opencalais.com/pershash-1/bb5919f6-
f008-3ae9-a3aa-a7981d9f95d0">
<rdf:type rdf:resource="http://s.opencalais.com/1/type/em/e/Person"/>
<c:name>John Scott</c:name>
<c:persontype>N/A</c:persontype>
<c:nationality>N/A</c:nationality>
</rdf:Description>
```

Instance:

```
<rdf:Description rdf:about="http://d.opencalais.com/dochash-1/3ce040fb-
7373-37b3-a2a6-528488c74b14/Instance/33">
<rdf:type rdf:resource="http://s.opencalais.com/1/type/sys/InstanceInfo"/>
<c:docId rdf:resource="http://d.opencalais.com/dochash-1/3ce040fb-7373-
37b3-a2a6-528488c74b14"/>
<c:subject rdf:resource="http://d.opencalais.com/pershash-1/bb5919f6-f008-
3ae9-a3aa-a7981d9f95d0"/>
<!--Person: John Scott-->
<c:detection>[My name is ]John Scott[, Co-Owner Tonic ]</c:detection>
<c:prefix>version="1.0"?> My name is </c:prefix>
<c:exact>John Scott</c:exact>
<c:suffix>, Co-Owner Tonic </c:suffix>
<c:offset>119</c:offset>
<c:length>20</c:length>
</rdf:Description>
```

Relevance:

```
<rdf:Description rdf:about="http://d.opencalais.com/dochash-1/3ce040fb-
7373-37b3-a2a6-528488c74b14/Relevance/20">
<rdf:type rdf:resource="http://s.opencalais.com/1/type/sys/RelevanceInfo"/>
<c:docId rdf:resource="http://d.opencalais.com/dochash-1/3ce040fb-7373-
37b3-a2a6-528488c74b14"/>
<c:subject rdf:resource="http://d.opencalais.com/pershash-1/bb5919f6-f008-
3ae9-a3aa-a7981d9f95d0"/>
<c:relevance>0.480</c:relevance>
</rdf:Description>
```

8.2.4 Document Categories

The Calais Web Service currently supports the following document categories:

Business Finance, Entertainment Culture, Environment, Health Medical Pharma, Hospitality Recreation, Law Crime, Politics, Sports, Technology Internet, Weather, Other

9 Bibliography

Brin, Sergey and Lawrence Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." World-Wide Web Conference. Brisbane, 1998.

Broder, Andrei. "A taxonomy of web search." ACM SIGIR Forum 2002: 3-10.

Calais: Frequently Asked Questions. 20 04 2009 <http://www.opencalais.com/faq>.

Ekeklint, Susanne. "Semantic Tagging." 2001.

Evri: About Us. 26 04 2009 <http://www.evri.com/about.html>.

Google: Google AJAX Search API. 20 04 2009 <http://code.google.com/apis/ajaxsearch/web.html>.

Guha, R., Rob McCool and Eric Miller. "Semantic Search." Proceedings of the 12th international conference on World Wide Web. Budapest, Hungary: ACM New York, NY, USA, 2003.

Jena: A Semantic Web Framework for Java. 20 04 2009 <http://jena.sourceforge.net/>.

Perez, Juan Carlos. PC World: Google Rolls out Semantic Search Capabilities. 26 04 2009 <http://www.pcworld.com/businesscenter/article/161869/google_rolls_out_semantic_search_capabilities.html>.

W3C: Semantic Web Activity. 21 04 2009 <http://www.w3.org/2001/sw/>.

Weglarz, Geoffrey. "Two Worlds of Data – Unstructured and Structured." Information Management Magazine September 2004.