THE PSYCHOLOGY OF A WEB SEARCH ENGINE

by

Antoine I. Ogbonna

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Computing and Information Systems

in the

Computing and Information Systems

Program

YOUNGSTOWN STATE UNIVERSITY

August, 2011

THE PSYCHOLOGY OF A WEB SEARCH ENGINE

Antoine I. Ogbonna

Signature:

Antoine I. Ogbonna, Student — Date

Approvals:

Alina Lazar, Ph.D., Thesis Advisor — Date

John R. Sullins, Ph.D., Committee Member — Date

Stephen P. Klein, Ph.D., Committee Member — Date

Peter J. Kasvinsky, Dean of School of Graduate Studies & Research — Date

Abstract

With the complexity of the World Wide Web, including the difficulty to retrieve information, we attempt to construct Yoool, a simple yet efficient hyper-textual web search engine prototype that utilizes specialized algorithms, such as WordLink, WordMap, and A-Rank to minimize hyperlink manipulation while rendering objective, relevant, and satisfying results to the user.

ACKNOWLEDGMENTS

Before I begin, I want to thank the Lord Almighty for allowing me to make it this far in life and to help me make it further. I also want to thank my father, Nicholas C. B. Ogbonna, my mother, Caroline C. Ogbonna for believing in me even when I did not believe in myself. May the Lord Almighty grant you perfect eternal peaceful rest. Also I want to thank my fiancée, Ogechi (aka Chichi) for her love, support, despite sleepless nights, and for believing in me. To my entire family, both nuclear and external, I am forever indebted in your support.

In addition, I take this opportunity to thank my advisor, Dr. Alina Lazar, for her support, wisdom, encouragement, direction, and suggestions in completing my thesis. Without all these attributes from my thesis advisor, this paper would not have been possible. I am eternally grateful to her. To my committee members, Dr. John R. Sullins and Dr. Stephen P. Klein, I thank you for your editorship, wisdom, suggestions, and support in completion of this thesis. To the entire family of the Department of Computer Science and Information Systems, including Ms. Connie Frisby, I thank you.

To my friends, I thank you from the bottom of my heart for being there and supporting me, especially Francis K. Adjei, David M. Kitt, Matthew Novelli, John Mela, Aleisa Drivere, Benjamin King, Andrae Reed and the rest of whom I did not mention.

As my time at Youngstown State University concludes, rest assured that you will not be forgotten. May the Lord Almighty bless you and grant you your heart's desires.

Thank You!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the advancement of the World Wide Web (hereafter referred as www), retrieving information to satisfy the most information-hungry user becomes very tricky and difficult. With the amount of content being created on a daily basis, techniques developed to try to disseminate that information into what is meaningful and what is not becomes a challenging pursuit. In the beginning of the www, many web pages were not as complex or available as they are today, thus retrieving information was quite easy and security was not an issue, however with the exponential growth of web pages, as shown by Netcraft[1]in Figure 1.2, the probability that irrelevant and unnecessary information is returned is exponentially increased. In this paper, we attempt to create a web search engine that employs many techniques to help us in the information retrieval process and reduce hyperlink manipulation.

Many users are inexperienced in the use of web search engines to search for information on the www. So a web search engine is supposed to be intuitive, helpful, and return relevant and meaningful results. In order to combat this void, we introduce our system, Yoool[2], a multi-faceted, simple, efficient web search engine that employs keyword-based and semantic algorithms. The source of the name was inspired by the Swiss mathematician Leonhard Euler [3] (depending on how you pronounce

Figure 1.1: Leonhard Euler

[1]http://news.netcraft.com/wp-content/uploads/2011/07/wpid-site_count_history2.png

[2]We purposely chose to have 3 O's in our name

[3]http //www.ualberta.ca/dept/math/gauss/fcm/History_People/Euler_Leonhard/Euler_Leonhard.jpg

his name) [See Figure 1.1] and the irrational number

$$e \approx 2.71828182845904523536028747135266249\ldots$$

better known as Euler's Number to represent the continuous exponential growth of the www. He was also an inspirational figure in the mathematical field of Graph Theory which is tremendously represented in the www. In addition, we purposely chose the name, from a mathematical standpoint, to look like a finite sequence (ie. $i = 1, \ldots, y$ or $i = 1, \circ \circ \circ, y$), to represent our goal to always index the www completely, despite the many intricacies that are found in the www. Thus, this paper shall serve as a repertoire of historical background, including showcasing some of our ideas in the field of information retrieval.

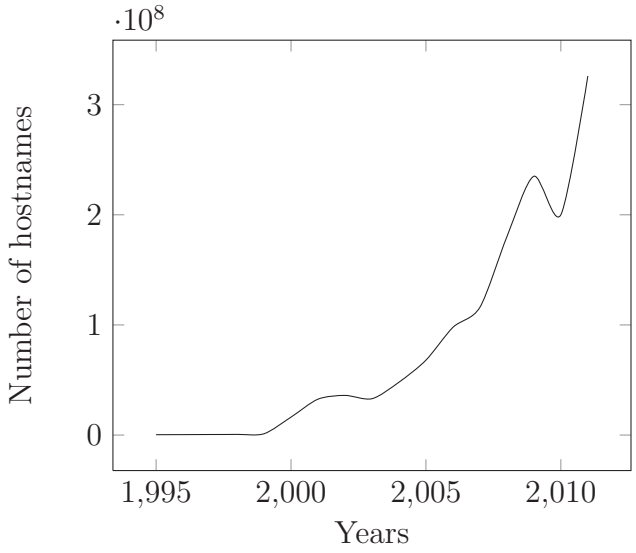


Figure 1.2: Exponential growth of World Wide Web

1.1 Motivation

Web search engines have come a long way in their index size, algorithms, and sophistication in understanding the www. However, the www cannot be compared to an academic literature as there is no structure, as explained by [1], due to the fact that no one really "owns" the Internet which hinders the ability to give the www some kind of structure. These days, many web search engines have become so commercialized that they lose focus on whose important, the user. In order to give a

historical background about web search engines, we list some below for the reader's understanding. If the reader is interested in reading about other web search engines, we invite them to peruse Wikipedia or other academic literature.

1.1.1 World Wide Web: The Early Days

WebCrawler

Coined as the "first Web search engine" to provide full text search, WebCrawler[4], live in 1994, had an early success. However, according to [1], WebCrawler was only able to index about 2 million web documents which pales in comparison to today's 16.7 billion web documents [5].

Yahoo!

Initially known as "Jerry and David's Guide to the World Wide Web", Yahoo![6] became one of the few search engines that are still in existence today, although, they were rather known as a web portal. However, due to fierce competition from other formidable companies, it lost tremendous market share, especially to Google as shown in Figure 1.3[7]. This could be due to the fact that Yahoo!'s early search engine was a pure keyword web search engine, that is, it only returned results based on the number of times the keyword appeared on a particular web page.

Google

Originally named BackRub, Google recognized some of the early flaws of the www and then devised a technique to combat some of those flaws. This technique, called PageRank[8], in [7], assigns a weight to every web page based on their outgoing and incoming hyperlinks. Their formula is explained later.

Despite the success of PageRank, manipulation of hyperlinks in web pages by web developers have increased a magnitude in order to get a high ranking despite the fact that their web page may not be relevant to the user's query. In addition, the increase of spam websites and paid hyperlinks inundated search results by ranking high, thus decreasing the user's experience. For example, a malicious web developer

[4]http://en.wikipedia.org/wiki/WebCrawler

[5]http://www.worldwidewebsize.com (circa May 2011)

[6]Registered trademark of Yahoo! Inc.

[7]http://searchenginewatch.com/article/2070937/comScore-April-2011-Search-Engine-Market-Share

[8]Registered trademark of Google, Inc.

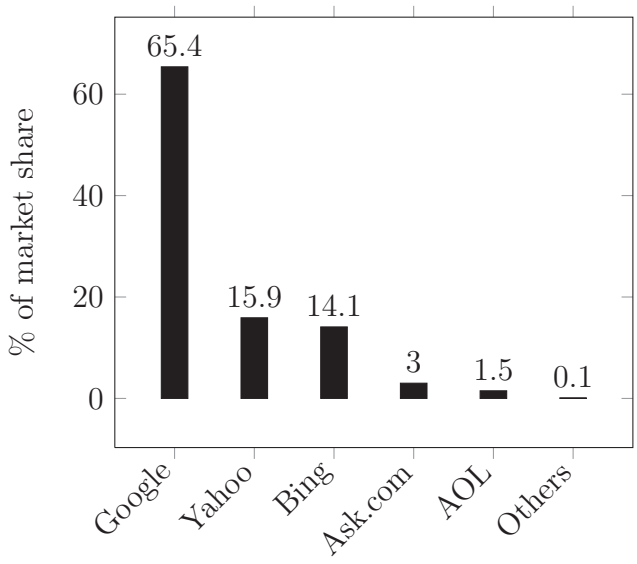


Figure 1.3: Market share of top search engines

may simply create numerous web pages that have their outgoing hyperlinks point to a specific page, hence increasing their importance. Combatting such tactics becomes challenging because of the complexity of the www. In this paper, we attempt to explain our approach in minimizing this manipulation thus maintaining or increasing the user's experience.

1.1.2 Yoool: Simple and Efficient

In order to combat the insane manipulation of hyperlinks by web developers, a web search engine system needs to employ sophisticated techniques. However, just having algorithms (ie matching keywords) is not enough. The search engine needs to attempt to "understand" the user's query in order to return relevant results. This technique of natural language processing is a challenging feat with many academic literature dedicated to it. In our web search engine, it attempts to not only match the keywords, but also tries to find relationships between words to understand the right result to return. That is, many words go together provided the system understand that there is an accepted syntax of the given language.

Example 1.1.1. Suppose a user types a query

[joe clinton white house]

With enough information, the system recognizes that the link weight between **joe** and **clinton** is close to nonexistent but the latter has a higher link weight with **white house** since there is a president (William Clinton) who resided in the White House. Thus the system would ignore the keyword **joe** and perform a search on the remainder, hence the modified query would be

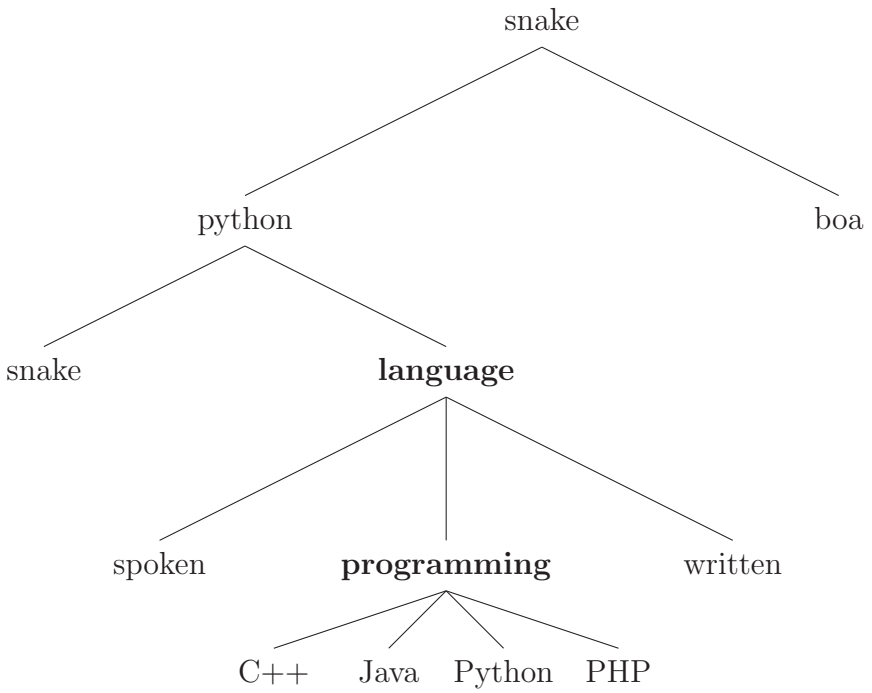
[~~**joe**~~ clinton white house]

As one can see this makes better sense.

Example 1.1.2. Suppose another user types a query

[learn snake programming language]

The system would find a link weight for each among themselves but notices that **snake** does not go with the rest of the keywords but **snake** can be broken down.



Notice that the bolded words appear in the query so intuitively, the missing link must be **python** since the system notices that the keywords **language** and **programming** also appear in the path of the tree. And so the new modified query becomes

[learn ~~snake~~ **python** programming language]

In Example 1.1.1 and Example 1.1.2 show our idea of WordLink and WordMap (to be discussed later) where depending on the occurrence of this link, the system would calculate and assign a weight. Intuitively, one may think of this technique as a derivative of natural language processing since there is an 'order' to the keywords or similar, but slightly different, to a thesaurus. This method would determine the set of pages to perform our ranking thus may eliminate many irrelevant pages, thus minimizing web page manipulation from web developers. In addition, we feel that this method can be applied to any language in the world provided the system understands the syntax of the given language.

As humans, it is intuitive to understand and decipher the ambiguities of query words, however to a web search engine, it is quite difficult and challenging. In this paper, we introduce some initial steps into deciphering ambiguities in keywords.

1.1.3 Speed Effect

With the ever increasing speed of internet connections and its affordability, a difference of 1 second becomes noticeable to the user, thus degrading their experience. In addition, indexing and crawling the www has to be able to scale to the ever increasing number documents found on the www. Yoool employs many techniques to index the web documents such as caching, running many instances of the crawler, and compressing documents locally. This allows Yoool to scale very well on large sets of documents while returning accurate results in near real-time. We feel that if results are not returned quickly, users will be disappointed. Therefore, a search engine has to be able to keep up with the demand and stress of the www while at the same time find the answers to users' query.

Chapter 2

Preliminaries

In this section, we explain our architecture used in building our search engine.

2.1 Architecture

In order to construct our system, we opted to use open-source software as our choice of use to exemplifies our beliefs that information should be freely shared and not licensed.

2.1.1 Hardware

Because of budgetary constraints, the system was run a commodity server that was provided and purchased by the author and hosted in the datacenter of Youngstown State University. A quick snapshot of the server is shown in Figure 2.1. Table 2.1 illustrates its specifications. In addition to the server, a 1TB external hard drive is

Central Processing Unit	Intel Dual-Core Xeon 3.6GHz (x2)
Random Access Memory	6GB PC3200 ECC DDR2 SDRAM
Hard Disk Drive	36.4GB U320 SCSI 15K rpm in RAID1 (x2)
Network	10/100/1000Mbps Ethernet (x2)

Table 2.1: Server specifications

attached to be used as a repository of the web documents collected from the www. To illustrate our setup, see Figure 2.2.

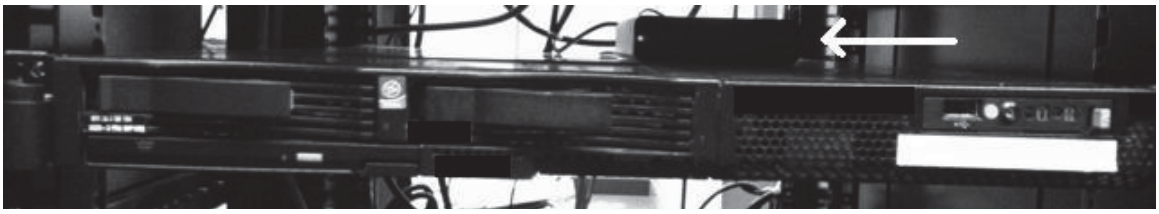


Figure 2.1: Server with a 1TB HDD (see arrow)

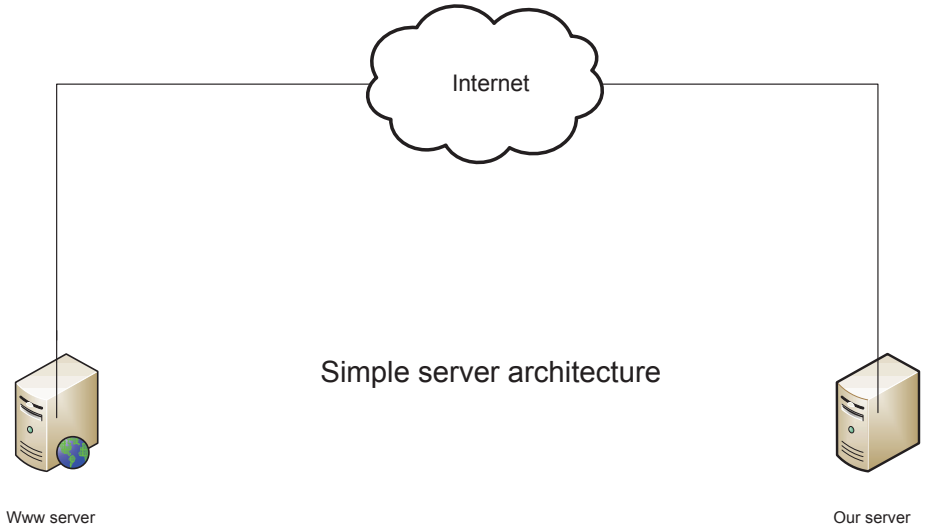


Figure 2.2: Server network architecture

2.1.2 Software

FreeBSD: A Unix-like operating software

FreeBSD[1] is an advanced Unix-like operating system with over 30 years of continuous development that descended from the BSD[2] family of operating systems. For this experiment, we chose this operating system because of its stability, security, and numerous documentations. If interested in learning how to use the operating system or its features, [5] and [8] are very helpful manuals. Some may argue that Linux should have being the chosen operating system but we opted to use FreeBSD instead because of its true closeness to Unix.

[1]Registered trademark of The FreeBSD Foundation (http://www.freebsd.org)
[2]Berkeley Software Distribution from the University of California, Berkeley

Python: An object-oriented programming language

> "Python[3] is a programming language that lets you work more quickly and integrate your systems more effectively. You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs"[4]

. We opted to use Python instead of other programming languages due to the fact that it was a language suited for our application and it had a quick deployment time. Also it has a vast rich library of modules that are at our disposal including documentation for reference. Many high end technology companies use Python for most of their applications. Although Python is quite CPU intensive, depending on the application, it is a tradeoff we accounted for in order to deploy our software quickly. For this experiment, we used version 2.7.2. In the next section, we discuss the library that we used for the project. If the reader is interested in learning the language, we invite the reader to read [4]

BeautifulSoup

BeautifulSoup[5] is a Python HTML/XML parser module. Since our project includes crawling the web for web pages, we made the use of this module in order to parse out relevant words to assist us in our algorithm of ranking the web pages. With this module we are able to use the source code that is downloaded with the assistance of urllib2, a built-in Python library. With the inexperience level of many web developers being so high, rendering poorly written web pages with BeautifulSoup is quite possible. Version 3.0.0 is used for this project. For illustration, see Example 2.1.1

Example 2.1.1. Take for example the sample code below, BeautifulSoup can parse any tags where the text between the tags are needed and returns them in an array. This process allows us to control or select what is parsed.

```
1    <!--Omitted code-->
2    <div>
3         text0 text1 text2 text3
4         <a href="http://www.somewebsite.com">Click</a>
5         text4 text5 text6 text7
6     </div>
7    <!--Omitted code-->
```

[3]Trademark of the Python Software Foundation
[4]Source: http://www.python.org
[5]http://www.crummy.com/software/BeautifulSoup

If we apply the BeautifulSoup module, we get the following

```
1    <!--Omitted code-->
2    <div>
3    text0 text1 text2 text3
4    <a href = "http://www.somewebsite.com">Click</a>
5    text4 text5 text6 text7
6    </div>
7    <!--Omitted code-->
```

If we wanted to parse out links from the code, we get

```
['http://www.somewebsite.com']
```

or if we wanted to parse just the visible text that appears when viewed on a browser, we get

```
['text0','text1','text2','text3','text4','text5','text6','text7']
```

We see that the power of the module makes our job easier otherwise we would have to write a whole module which in itself is tedious. Any aspiring enthusiast, we recommend BeautifulSoup for your web project.

MySQL: Relational Database

MySQL[6] is our choice of relational database for our use. We chose MySQL because it was free and emphasizes on our idea of openness. We currently are using version 5.5.9. Although MySQL is able to scale to billions of rows, in our experiment, its performance was enough for the task. Our goal is to build fast, scalable database for our experiment, including building a new filesystem, dubbed QuickFS that would provide redundance, fast search, and error recovery.

[6]MySQL is registered trademark of Oracle Corporation

Chapter 3

Yoool: An overview

Building a web search engine is no easy task. In fact, it can be quite daunting to say the least. Because of the complexity of the www, catching all errors is a near impossible feat as many people coming online do not know the structure of the www or its implementation. In addition, much care has to be taken for people who run their website at home since generally, their internet speed may not be as fast as a corporation who may have a fiber optic internet connection.

3.1 Crawling

Crawling (downloading the source code of a webpage) a website takes time. It takes sophisticated software and hardware to be able implement such. In addition, many considerations need to be taken especially the network architecture of the Internet and the interval at which a web site can be crawled. If many instances of the crawling software is implemented, a network congestion may occur. Upon crawling the website, many processing is done on the page. We list them below.

3.1.1 WordLink

Understanding and processing natural language is a tedious, difficult process to do. Many academic literature have been dedicated to natural language processing. However in our case, we take a different approach. In a written language such the English language, there is an accepted syntax to compose sentences.

Example 3.1.1. Suppose the following statement was written,

[I go school]

It would be slightly accepted, however a proper way to compose the statement in the widely accepted syntax would be

[I am going to school]

So if the statement is broken apart, we see there is an inferred relationship between the words, that is, each word has a link to every other word where the link to the adjacent word is strongest and decreases in strength by a factor of $\frac{1}{x}$ where $x \geq 1$ is the number of hops from the chosen word position. Thus we would have

$$\begin{array}{rcll} \text{I} & \longrightarrow & \text{am} & [1] \\ \text{I} & \longrightarrow & \text{going} & [\frac{1}{2}] \\ \text{I} & \longrightarrow & \text{to} & [\frac{1}{3}] \\ \text{I} & \longrightarrow & \text{school} & [\frac{1}{4}] \\ \text{am} & \longrightarrow & \text{going} & [1] \\ \text{am} & \longrightarrow & \text{to} & [\frac{1}{2}] \\ \text{am} & \longrightarrow & \text{school} & [\frac{1}{3}] \\ & \vdots & & \\ \text{to} & \longrightarrow & \text{school} & [1] \end{array}$$

Now we repeat the same process starting from the right most word. So a statement that has the words [I am] or [am I] would have a link weight of $\frac{1+1}{2} = 1$ since syntactically those words go hand in hand which is a widely accepted syntax. As we can see, this process uses a combinatorial approach. That is, if we do the process from left to right and vice versa, the number of links is

$$\begin{aligned} 2 \cdot \binom{5}{2} &= 2 \cdot \frac{5!}{2!(5-2)!} \\ &= \frac{2 \cdot 120}{12} \\ &= \frac{240}{12} \\ &= 20 \end{aligned}$$

Based on this relationship, we see that a statement such as

[The car is going on the road to school]

the system would see a link in some of the words in the previous statement which are

bolded

[The car is **going** on the road **to school**]

Heurestically, the system recognizes the links and assigns a weight based on its database of those links.

Remark 3.1.1. So in general the number of links that are found in a document is given by

$$2 \cdot \binom{n}{2} = \frac{2(n!)}{r!(n-r)!}$$

This is simply an amended formula of a combination on a set which can be found in [2].

In order to apply the same idea to web documents, consider the following.

Definition 3.1.1. Let $W = \{w_1, w_2, \ldots, w_n\}$ be the set of words in a nonempty document D. Then for each $w_i, w_j \in W$, provided $i \neq j$, we define Θ to be the link between w_i and w_j. That is

$$w_i \quad \overset{\Theta}{\longleftrightarrow} \quad w_j,$$

for $i = 1, \ldots, n$, $j = 1, \ldots, n-1$.

Suppose we have a large collection of documents, then there is a likelihood that there will be many words that are repeated and since there are so many ways one can compose a statement, this would increase the weight of the links that are repeated. So consider the following lemma that describes WordLink.

Lemma 3.1.1. *Let $D = \{d_1, d_2, \ldots, d_a\}$ be a set of nonempty documents. Let $W = \{w_1, w_2, \ldots, w_b\}$ be the set of words for each $d \in D$ and $L = \{\Theta_1, \Theta_2, \ldots, \Theta_c\}$ be the set of links for each $d \in D$ with the restriction that each word cannot point to itself. Then the WordLink, defined as $WL(l)$, is given by*

$$WL(l) \;=\; \sum_{i=1}^{m} \left(\sum_{j=1}^{n} l \right),$$

where $m = |D|$, $n = |L|$ and $l \in L$.

Proof. Let $D = \{d_1, d_2, \ldots, d_a\}$ be a set of nonempty documents. Let $W = \{w_1, w_2, \ldots, w_b\}$ be the set of words for each $d \in D$ and $L = \{\Theta_1, \Theta_2, \ldots, \Theta_c\}$ be the set of links for each $d \in D$ with the restriction that each $w \in W$ cannot link to itself. Let $m = |D|$ be the number of nonempty documents and $n = |L|$ be the number of links in each $d \in D$ provided $n \leq 2 \cdot \binom{k}{2}$. Then for each $d \in D$ we sum up all the links and compute their relative weight within each $d \in D$, it gives us

$$\sum_{j=1}^{n} l$$

but there could be more documents containing more of the same links, that is, $m \geq 1$. And so we have

$$\sum_{i=1}^{m} \left(\sum_{j=1}^{n} l \right)$$

Computing for the weight relative to the set of documents gives us

$$WL(l) \;=\; \sum_{i=1}^{m} \left(\sum_{j=1}^{n} l \right)$$

■

Example 3.1.2. Let the following be a set of 2 documents containing various words.

$$\begin{aligned} d_1 &= [\text{ I went to school today }] \\ d_2 &= [\text{ The car stopped rolling }] \end{aligned}$$

If we apply Lemma 3.1.1, we get the following set of WordLink weights where $WL(l)_1 \geq$

$WL(l)_2 \geq \ldots \geq WL(l)_p$:

$$\begin{aligned} L_1 &= \{1, 1, \ldots, 0.25\} \\ L_2 &= \{1, 1, \ldots, 0.33333333\} \end{aligned}$$

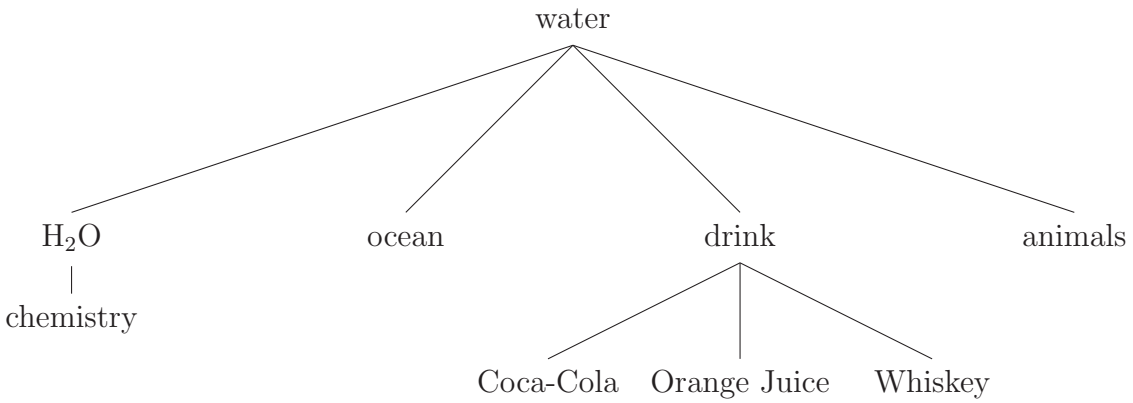
As we can see, this process is computationally heavy as the number of words increase but with the decreased cost of hardware, implementing a system of such high computing power is possible at a lower cost. However, it is an ongoing algorithm that could be refined in the future. The beauty of this algorithm is that it assumes that there is an accepted syntax, so uncommon phrases would have a low WordLink weight thus rendering it useless. In a way, it becomes like a neural network where one feeds the system a bunch of documents, and the system looks for patterns, dynamically changing their WordLink as more documents are fed. The idea of WordLink was greatly inspired by the works of [6]. We invite the reader to read more about it.

3.1.2 WordMap

We view the www as a big graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes or pages in G and $E = \{e_1, e_2, \ldots, e_m\}$ is the set of edges or hyperlinks in G. If we apply this same idea locally, that is, on a single document, we can see that the words on the document represent the nodes and their inferred position relative to their accepted syntax is the link between them. Now if we cluster these words based on their link occurrence, one can deduce that the similar meaning and appropriate syntax to use. Consider the following example.

Example 3.1.3. If we had a large set of documents, and computed their WordLink weight, we can find out their similarities with each other. For example, suppose the following set had the word **water** then a map is can be drawn based on the links of

the word.



Performing such computation on a large set of documents such as the www becomes quite intensive. The idea behind this was the fact that by having a 'map' of words that are formed from the www documents, one can determine which document has the appropriate information even though the exact query word may not be used in retrieving the documents. Intuitively, one views this map as a map of similar meanings of words based on their WordLink weights. Such algorithm would be ideally run at crawling time. However, due to time constraints, this algorithm was not implemented for the sake of this project.

3.2 Searching

Text mining in itself is both an art and a science especially if it involves a large set of documents such as the www. Many academic literature regarding text mining have been published. Most notable one was [1] and [3]. However their methods come up short on some key areas. They are discussed below

3.2.1 HITS Algorithm

Jon M. Kleinberg's is best known as the one who understood the link structure of the www. In his paper [3], he describes the notion of *Hubs* and *Authorities* in his now famous HITS Algorithm, in which he uses eigenvector methods to perform searches on the web. That is, he hypothesized that the link structure of the web provided a lot of information in terms of finding which page pointed to what page and which pages received pointers to themselves. Ideally, this approach would be fantastic however, these days it would suffer tremendously because many users do not know how to use

the www for research and there are many hyperlinks that point to the same page that they are on, rendering those links useless. His approach idealizes a properly structured www as in a connected graph[1] but alas it is not so.

3.2.2 PageRank

Arguably similar to Kleinberg's approach, [7] proposes a ranking weight for each web page on the www. Its idea was for each hyperlink on a web page that points to another page constitutes a 'vote' on the importance of that page. It calls this importance PageRank[2]. According to [7] it defines PageRank given in Definition 3.2.1.

Definition 3.2.1. Let $E(u)$ be some vector over the Web pages that corresponds to a source of rank. Then, the PageRank of a set of Web pages is an assignment, $R^{'}$, to the Web pages which satisfies

$$R^{'}(u) \;=\; c\sum_{v\in B_u}\frac{R^{'}(v)}{N_v} + cE(u)$$

such that c is maximized and $||R^{'}||_1 = 1$ ($||R^{'}||_1$ denotes the L_1 norm of $R^{'}$).

In simple terms, PageRank calculates the weight or importance of web pages based on the number of hyperlinks that points to it. Such tactic would work if all web developers would adhere honestly to the creation of their website unfortunately there are malicious web developers who would 'game' the system for financial gains to be on the list of top 10 hyperlinks. For example, if a malicious web page has many hyperlinks pointing at it, it would rank high even though their page may be irrelevant thus degrading the user's experience. In addition, calculating the PageRank is done at crawling time which would require heavy computing hardware. Also the algorithm does not take into account the personalization of each user even though they may use the same keywords but have different interpretation.

3.2.3 A-Rank: Be first on the list

In order to understand our algorithm, we visualize the web as both an undirected and directed graph $G = (V, E)$. Now each hyperlink constitutes a knowledge link. In other words, if page a has a link to page b then a is "transferring" knowledge and a is

[1] See [6] for definition of a connected graph

[2] Named after Lawrence Page, co-founder of Google, Inc.

"receiving" from b and if b points to another page c then a 'knows' about it through b. Then c is very important since both b and a knows about it. A good analogy would be to analyze the way routers work on the Internet. That is each router knows about almost every node through their links. We approach this same idea to the www but slightly different. Now if a user performs a query, the relevant pages are fetched and call that set of pages Z where $Z \subset G$. In order to determine the importance or weight of each web page in Z, we randomly arrange each web page as a $n \times n$ matrix. Call that matrix $A_{i,j}$ where i and j are the rows and columns of A. The square matrix would only contain 0 or 1 where 1 determines a link from i to j. Suppose a query q fetched 4 web pages, then A would like the following:

$$A_{i,j} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{bmatrix}$$

Now we assume the fact that page a cannot point to itself, so $A_{i,i} = 0$ where $i = j$. Thus $A_{i,j}$ becomes

$$A_{i,j} = \begin{bmatrix} 0 & x_{1,2} & x_{1,3} & x_{1,3} \\ x_{2,1} & 0 & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & 0 & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & 0 \end{bmatrix}$$

As mentioned earlier, we visualize the www as an undirected graph and thus each hyperlink is a link of knowledge from one page to another. Now the idea is to find out which page has the most knowledge about a particular query. To be able to answer this dilemma, we employ the use of Graph Theory knowledge to help us understand the process that is going on. The following definitions can be found in [6].

Definition 3.2.2. Let G be a graph. We say that G is a **connected graph** if for each pair of nodes v_x and v_y, there is a path that joins v_x and v_y.

Definition 3.2.3. Let G be a connected graph and $v \in V(G)$ be a node of G. We say $e(v)$ is the **eccentricity** of v if it is the distance to a node, say $u \in V(G)$, that is farthest from v. Thus

$$e(v) = \max\{d(u, v) : u \in V(G)\}$$

Applying Definition 3.2.3 to Z, we can see that through the links, it allows us to find which page has the most knowledge. One can think of it as a popularity measure. That is, the more people know about you, the more important you are. For example, the page `http://www.yahoo.com`, which belongs to Yahoo!, Inc. is a very important page because so many people know about it even though they don't explicitly link to it. This is where our view of the www changes to a directed graph. So for each page in Z, we find the most efficient route to the farthest page provided there are no loops; that is a page cannot loop back on itself stopping the finding of the path. The numeric value for each in Z corresponds to the number of hops it takes to the farthest page. Thus we augment matrix $A_{i,j}$. To better illustrate, see Figure 3.1, where in the

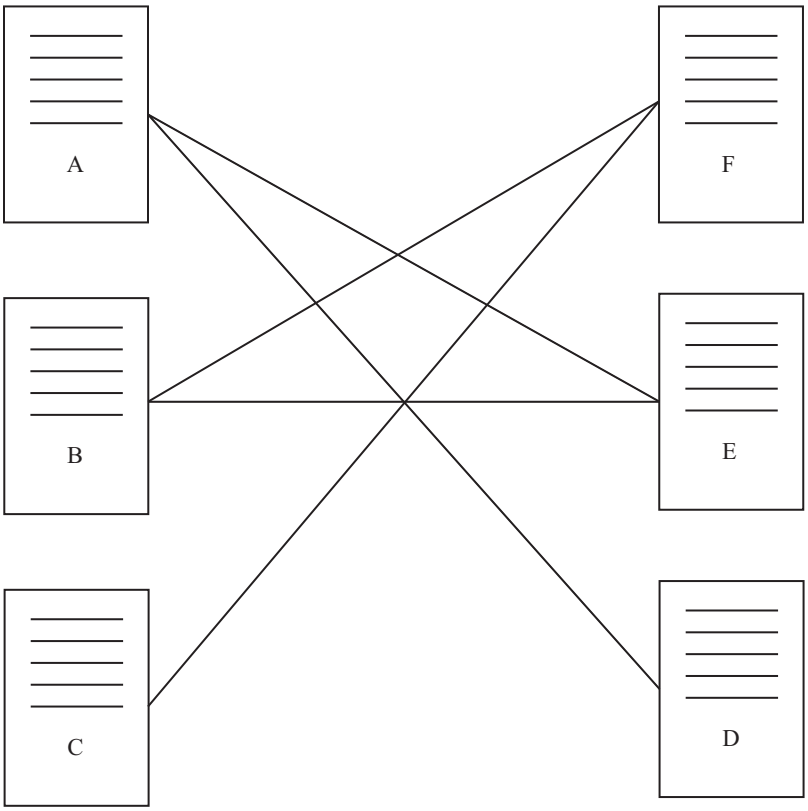


Figure 3.1: Simple collection of web pages

picture, page C and D would easily be quite important because they have the longest path, which in this case to each other. However to balance the knowledge, we need to know how much each webpage knows that contributes to the popularity relative

to Z. This process allows us to form a matrix equation.

$$\begin{aligned} Ax &= b \\ &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

where b is the $n \times 1$ matrix collection of their eccentricities as defined in Definition 3.2.3. Performing the computation is just a matter of using Reduced-Row Echelon Form to solve for $\mathbf{x}$. Now to determine the A-Rank of $\mathbf{x}$, that is each page in Z we perform some computation using the following equation.

$$AR(x_i) = \left| x_i \sum_{i=1}^{|Z|} \frac{1}{x_{i+1}} \right|$$

Example 3.2.1. Suppose a query q is given and the pages fetched are the ones given in Figure 3.1. For ease of understanding, let $x_1 = A$, $x_2 = B$, $x_3 = C$, $x_4 = D$, $x_5 = E$, $x_6 = F$. Then $\mathbf{x}$= $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ are the pages. And so our matrix would be

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 5 \\ 5 \\ 3 \\ 4 \end{bmatrix}$$

Rearranging to solve for $\mathbf{x}$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ 3 \\ 5 \\ 5 \\ 3 \\ 4 \end{bmatrix}$$

Solving

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 5 \\ -2 \\ 6 \\ 6 \\ -2 \\ 5 \end{bmatrix}$$

Now to compute the A-Rank for each page relative to Z, we get

$$\begin{aligned} x_1 &= 2.33333333 \\ x_2 &= 0.53333333 \\ x_3 &= 2.6 \\ x_4 &= 2.6 \\ x_5 &= 0.53333333 \\ x_6 &= 2.333333333 \end{aligned}$$

As one can see, this corresponds quite well with our hypothesis that pages C and D are quite important even though they may not have a lot of hyperlinks connecting them to others.

The beauty of this technique is that it scales quite well since we are simply solving a linear system of equations.

Definition 3.2.4. Let D be the set webpages fetched from a query. Let A be an $n \times n$ matrix of the set of links connected within each page. Let b be the set of eccentricities that corresponds to the each $a \in A$. Then the A-Rank[3] for each $d \in D$, denoted $AR(d)$, is the ranking of the page obtained through the following equation.

$$AR(d_i) = \left| d_i \sum_{i=1}^{|Z|} \frac{1}{d_{i+1}} \right|$$

[3]Named after the first initial of the author and corresponds to the first letter of the English alphabet since every web page wants to be first

Chapter 4

Project Results

In this section, we present our data and the aesthetics we plan on implementing on the website. Currently, the web address can be found at `http://www.yoool.net`. We hope to be able to acquire the .com version of the address as it is presently owned by an asian company.

4.1 Data collected

4.1.1 MySQL current data

In addition to the data collected, we present a schematic that outlines how our software crawls the www to fetch documents. It is show in Figure 4.1. Comparing

Hyperlinks	190120
Download pages	4637
Distinct words	82257
Links between pages	191734

Table 4.1: MySQL database numbers

the download pages, we average a page has ≈41 hyperlinks in it. All these data was collected within a 48-hr period. We ran 1 instance of the crawler, however we could have run many instances but consideration had to be taken care since the project was on an education network. Many of the web pages download came from various countries around the world. They are illustrated in In addition, we encountered many 404 (Page Not found) pages (about 1618 links were bad). Upon closeup, we see that our URLResolve function did not properly account for the double dots in a relative hyperlink.

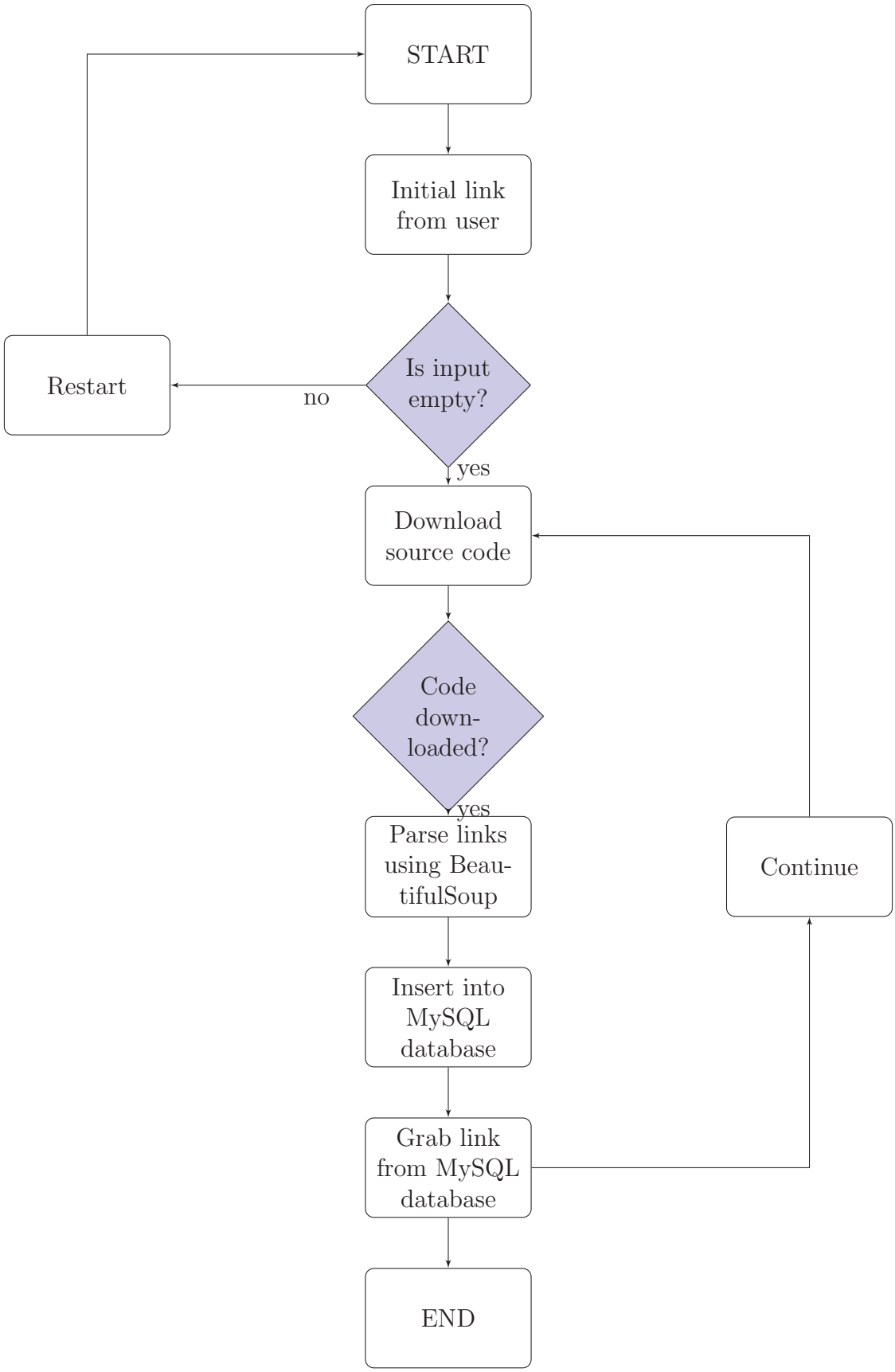


Figure 4.1: Flowchart of YooolBot crawler

United States	55%
Canada	25%
India	20%
China	10%
Norway	<0.1%
Pakistan	<0.1%

Table 4.2: Geographic location of web pages downloaded

4.2 Query results

4.2.1 Query

In all, our approach to building this search engine is simplicity. In fact, our homepage is simple because the user, who is the most important in all of this, is not inundated with nonsense. For our homepage, we present simply a textbox for the user their query. There are no links, no images, because the user is not interested in those. We would like to minimize the effort that the user would exert by quickly returning results in a pleasant manner using AJAX (Asynchronous JavaScript And XML), that is, return relevant results but do not bombard the user with unscrupulous results. In the homepage, we return 3 results at a time, because we feel that is an appropriate number so as not to overwhelm the user. In the results page, we propose returning 10 results with a dynamic window that provides quick and efficient curated results (for the sake of time, this was not implemented at time of writing). The whole frontend of the system was written in PHP with a connection to the MySQL database. Although our choice would have been to write the program in C++ but PHP is easier to code. For the illustration of the homepage, see Figure 4.2. Now when perform a search query [barack obama], our system returns the following that is shown in Figure 4.3 which we believe returns satisfying results despite the limited webpages that were downloaded.

4.3 Future explorations

4.3.1 Project continuity

Despite the many challenges that were faced during the building of Yoool, we feel that we just scratching the surface. The next challenge would be to implement WordLink and WordMap. In addition, we would like to add many features to the search engine.

Figure 4.2: Yoool homepage (circa 2011)

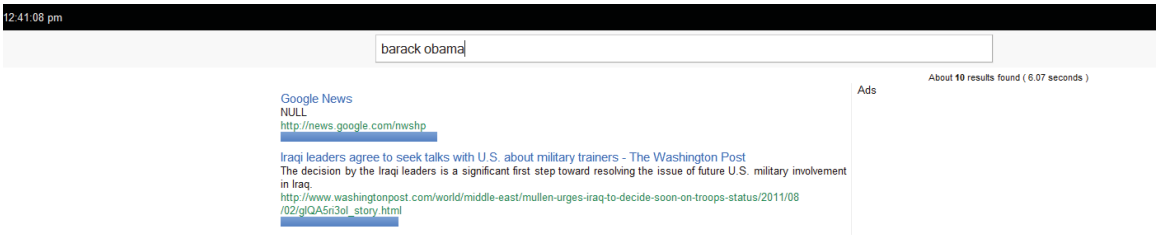


Figure 4.3: Yoool search results page (circa 2011)

Our ultimate goal for Yoool is to be able to understand the query and return the appropriate result whether it is an image, a map, pictures etc, all from one location.

4.3.2 Financial pursuit

The undertaking of this project is immense. However, it would be best not to be deterred due to financial gain. Yoool would never be influenced by monetary gains, that is, one cannot pay to be at the top of the results page. In our mind, the user is always the center of attention and thus should remain so. Any investor/volunteer that believes in our pursuit is welcomed.

References

[1] Sergey Brin and Lawrence Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, Computer Networks and ISDN Systems **30** (1998).

[2] Richard A. Brualdi, *Introductory Combinatorics*, 3rd ed., Prentice-Hall, Inc., 1999.

[3] Jon M. Kleinberg, *Authoritative sources in a hyperlinked environment*, J. ACM **46** (1999), 604–632.

[4] Mark Lutz, *Learning Python*, 3rd ed., O'Reilly Media, Inc., 2008.

[5] Christopher Negus and Francois Caen, *BSD® UNIX® Toolbox*, Wiley Publishing, Inc., 2008.

[6] Antoine I. Ogbonna, *Eccentricity Sequence of 2*, Master's thesis, Youngstown State University, Youngstown, Ohio, 2010.

[7] Lawrence Page, *The PageRank Citation Ranking: Bringing Order to the Web*, 1998.

[8] David I. Schwartz, *Introduction to UNIX®*, 2nd ed., Pearson Education, Inc., 2006.

Index