UNIVERSITY OF CINCINNATI

Date: May 9, 2007

I, Justin Fister,

hereby submit this work as part of the requirements for the degree of:

Master of Science

in:

Computer Science

It is entitled:

Correlation Analysis of On-Page Attributes and Search Engine Rankings

This work and its defense approved by:

Chair: Dr. Yizong Cheng

Dr. Kenneth Berman

Dr. Fred Annexstein

Correlation Analysis of On-Page Attributes and Search Engine Rankings

A thesis submitted to the Graduate School
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in the Department of
Electrical & Computer Engineering and Computer Science
in the College of Engineering

May, 2007

by

Justin Fister
B.A., Berea College, 1999

Committee Chair: Dr. Yizong Cheng

Abstract

Despite sophisticated search engine algorithms designed to eliminate irrelevant content, many Internet content providers with valuable information are unable to achieve visibility in the search engine results due to many factors including lack of information, misinformation, and search engine trade secrets. Furthermore, a survey of information on the topic yields questionable advice unsubstantiated by research and statistical analysis. Consequently, this thesis addresses some of the problems facing content providers by providing relevant statistics, as well as a simple research framework allowing content providers to easily extend this research and more fully understand the search engine ranking algorithms. Due to the large scope of the search engine ranking topic, this thesis focuses solely on examining the relationship between on-page attributes and search result ranking. Nevertheless, the research framework presented in this thesis can be altered to extend this research beyond the analysis of on-page attributes.

Acknowledgments

I express sincere thanks to Dr. Yizong Cheng for his guidance and encouragement during this process. I appreciate the freedom he gave in allowing me to choose a research area of interest to me, and for always being available for assistance and advice.

I would also like to thank Dr. Kenneth Berman and Dr. Fred Annexstein for their help in reviewing my thesis and participating in my defense.

Moreover, I would like to extend my gratitude to the ECECS Department and the University of Cincinnati for the excellent graduate-level education.

Finally, I want to thank my family for their encouragement and support during this entire process.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet's wealth of information makes it the most influential development of the 20th century, literally putting libraries upon libraries at the tip of our fingers. In decades past, information was transmitted through word of mouth, by phone, and through painstaking hours sifting through books in the library. The Internet has been a blessing not only to the consumers of information, but also to the publishers of that information. The number of web pages continues to grow substantially year after year as businesses move online, educators make resources available, and everyday users try their hand at blogging. Never has there been a time in which one's voice could be so easily heard across the world.

However, this blessing of informational indulgence has brought with it the plague of informational inundation. In August of 2005, Yahoo Inc. claimed to have nearly 20 billion web pages indexed in its search engine [25]. It is in this crowded landscape that website owners and content providers are faced with the new challenge of fighting to be heard and positioning themselves to be easily discovered. Fortunately, content providers have in their corner Internet search engines, which attempt to direct users to the best web pages for a given query. Currently, Internet search engines account for the majority of many sites' traffic,

sending visitors to hundreds of millions of web pages every day [20].

This thesis highlights important conceptual information regarding search engines, search engine marketing, and search engine optimization (SEO) for the benefit of website owners, SEO providers, and anyone publishing content on the Internet. Moreover, this thesis provides captured data and analysis of search engine results with the intention of discerning the relative importance of on-page attributes in the search engine ranking algorithm. This information can then be used by content providers to make their pages found in the large sea of web pages on the Internet. Finally, it is my hope that the tools and processes presented in this thesis for the capture and analysis of search result data can be extended and enhanced for further research in this area.

Chapter 2

Internet Search Engines

2.1 What is a Search Engine?

In this document, the term “search engine” refers to the website that visitors use to search for Internet documents, but also to the entire system used to “spider” the Internet, store and index web documents, and conduct searches. Although most search engines include many other related features and services including image search, news, shopping search, directories, and more; this document will only be focusing on the portion of a search engine that is used to search web pages. Furthermore, modern search engines are capable of indexing a variety of content types including Adobe Acrobat (pdf), Microsoft Word (doc), Microsoft Power Point (ppt), Microsoft Excel (xls), and more [7]; however, this document only covers the indexing and retrieval of HTML web documents.

2.2 Search Result Ranking

Amazingly, within seconds of entering in a search query, a search engine is capable of searching billions of documents and returning a list of relevant results.

The means by which search engines achieve such accuracy is a trade secret, but many aspects have become known and even more have come under speculation. For content providers understanding the ranking of results from search engines is of utmost importance, and the data presented in this thesis suggests further insights into the ranking process. There are many factors involved in ranking web pages for a given query, some of which are examined below.

On-Page Attributes

These are the qualities that are within the page itself. In the early days of the search engines, this was the only factor in ranking search results. The data collected in this thesis pertains mostly to on-page attributes. Examples of these include page title, URL, image ALT attribute text, image src attribute, link text, keyword frequency, keyword proximity, and other characteristics that can be found within the HTML itself.

Link Popularity or Page Rank

On-page attributes are crucial to rating the relevancy of a web document, but by themselves allow too much manipulation by unscrupulous content providers seeking to unfairly boost their rankings. Fortunately, the status quo of the search engine world would be shattered by the concept of Page Rank brought to life by the two founders of the Google search engine. Page Rank is an algorithm that assigns importance to a page based on the number and importance of incoming links to that page. In its essence, each web page casts votes for other pages by

linking to them [2].

End User Behavior

The role of end user behavior in the ranking of search results is speculative because end user information is held only by the search engines themselves. However, the value of such information and the ability with which it can be tracked makes its use very likely. An example of this type of information is click through data [19]. It follows logically that search results that are visited more often than results of a higher ranking should be given a boost in ranking.

Another example of end user behavior is traffic patterns. Why shouldn't a site's traffic be considered in the equation? Isn't a site more important if it gets more traffic? Isn't a link from a site with high traffic of more value than a link from a site with low traffic? The use of traffic in search rankings is purely speculation, yet the data could be easily be obtained from sources such as Alexa [1] or the Google Toolbar [9].

2.3 Survey of Search Engines

A greater understanding of search engines can be gained by reviewing the current search engine landscape.

Yahoo Search

Yahoo's current search engine was acquired from Inktomi in 2003. Inktomi was created in 1996, making it the oldest of today's major search engines [22]. Despite the fact that Yahoo is the world's most popular website [1], the popularity of the Yahoo search engine is still a distant second to Google [5].

MSN Search

Late in 2004, Microsoft released a beta version of its MSN Search service, which it developed from the ground up in an effort to dominate the search market [14]. User reviews were not the best, and the search engine languished at third place in the search engine race [5].

Ask.com

Ask Jeeves, as it was originally named, was launched in 1997 as a natural language search engine. In 2001, Ask purchased the Teoma search engine to replace the older underlying technology [23]. Although Ask receives only a small percentage of total searches, it remains the third largest search engine that maintains its own index [5].

Swiki

Swiki [6] is a newcomer into the search engine landscape and will, with all probability, not be more than a niche competitor. However, the concept used by Swiki is one that will most likely be utilized to an increasing degree by the

major search engines. Swiki uses feedback from end users to help rank search results. It's speculative whether the larger search engines will use such explicit user input, but it appears likely that end user behavior will be used in some form or another.

Google

Created in 1997 as part of a research project at Stanford University [22], Google has become the name brand in Internet searching. Their PageRank algorithm (discussed earlier) set them apart with the reputation of delivering the most relevant results of any search engine. The popularity of this search engine has made the term “google” a verb synonymous with conducting an Internet search. Despite heavy investment and research from competitors, Google continues to remain the most popular search engine [5]. Because of its enormous success and popularity, the search results data analyzed in this thesis comes from the Google search engine.

Chapter 3

Search Engine Optimization

3.1 SEO Intro and Prerequisites

In this document, the term Search Engine Optimization (SEO) will be used interchangeably with Search Engine Marketing to refer to any activity used to promote a web page's position in the search engine rankings. This section will provide background information on important steps for content providers to follow in regards to SEO, and the data analyzed in this thesis will shed some light on the relative importance of a subset of the information presented in this section.

Of course, prior to any talk of SEO, there must be a website publicly available for crawling by the search engines. It is assumed that this website consists of multiple static or dynamic pages represented as HTML.

3.2 Getting Found

Search engines have always had the option of submitting a URL to be crawled and indexed, and there is no harm in doing so. However, with link popularity being a key feature of all modern search engines, the best way to ensure that your web pages are indexed and remain in the search engine index is to obtain a link to your site from a site that is already currently indexed. This initial link is the first step in achieving any degree of visibility on the Internet, and it can be obtained in many ways including link exchanges, submission to a website directory, or paid advertising. Information on the means of gathering links will be presented in the Link Building section later in this chapter.

3.3 Targeting Keywords

Often, the difference between a page that gets a high search ranking and one that gets a low search ranking is the small amount of time spent thinking about the keywords that the page is targeting. Imagine a tale of two websites in which one is built using decent content with accurate and descriptive titles given to each page. The other website focuses on incredible content, but its editors fail to provide titles for the pages, so the Content Management System inserts a default title. Even though the second site has better content, search engines may not be able to determine this, but they *are* able to hone in on the keyword rich title of the first website. Information given in the latter parts of this thesis will discuss *where* these keywords will be used, but it is up to the content provider to first

determine *which* keywords are to be targeted, or marketed, to the search engine.

So, how does one know what keywords should be chosen? Certainly, the first place to start is to think of keywords that someone would enter into a search engine to find your website or web page. For example, if you are developing a website or section of web pages on Spanish Medical Terminology, then you could make a list of possible searches. The list might look like this:

- medical Spanish
- Spanish medical terms
- Spanish medical terminology
- Spanish medical dictionary
- Spanish for health professionals
- Spanish medical encyclopedia

Once a list has been made, then it's necessary to analyze the resources and scope of the website being promoted through SEO. Newer, or cash-starved websites will probably not be able to target high traffic search terms because it is likely that well-established websites with deep pockets will already be competing for these terms. Once a candid assessment of the situation has been made, then the list of possible key phrases can be analyzed further.

There are several tools available for analyzing search term / key phrase popularity, but the Overture Keyword Selector Tool [24] is probably the most widely used. By entering in a keyword, or keywords, the tool will generate a list

of terms that match or contain the given keyword(s), as well as the number of times the search term(s) were queried in the previous month.

Figure 3-3-1. Overture Keyword Selector Tool Results

Keyword Selector Tool

Not sure what search terms to bid on?
Enter a term related to your site and we will show you:

- Related searches that include your term
- Estimated number of times that term was searched on last month

Get suggestions for: (may take up to 30 seconds)

medical spanish

Note: All suggested search terms are subject to our standard editorial review process.

Searches done in November 2006	
Count	Search Term
3036	medical spanish
2140	medical spanish translation
2021	medical english spanish translation
1068	spanish english medical dictionary online
1037	spanish english medical dictionary
435	spanish medical dictionary
424	clinically interview medical medical relevant spanish spanish textbook
266	learn medical spanish
263	spanish for medical personnel
250	english spanish spanish english medical dictionary

With this information, content providers can choose which search term(s) are suitable to use when promoting their web page(s).

3.4 On-Page Attributes

For some niche topics that target a very specific, low-traffic phrase, on-page attributes may be the only necessary SEO concern of the content provider. In many such cases, careful usage of the targeted keyword(s) in the page itself will be sufficient to gain the desired search result rankings. This is the ideal case because on-page factors are completely under the control of the content provider.

Having created a page of content, SEO for on-page attributes answers the question, "What changes can I make to this page that are likely to boost the search result ranking of this page?". The answer to that question usually depends on who you ask because so much of SEO is speculative. Fortunately, the latter portion of this thesis will address that question by analyzing thousands of search results to determine the relative importance of over a dozen on-page factors.

3.5 Link Building

Often, conducting SEO using on-page attributes and targeted keywords is not enough to gain visibility in the search engine results. Many search terms are

highly competitive and the stakes can be very high. Due to the increased emphasis on the number and quality of links, it is important that content providers needing greater visibility in the search results participate in "link building", or the process of acquiring inbound links.

There are literally hundreds of possible ways to approach the task of generating inbound links, so only the most common of these will be presented.

Link Exchange

The most common of all link building methods is the link exchange, in which Website A places a link to Website B in exchange for Website B linking to Website A. Potential issues with the link exchange may arise when one website has significantly greater PageRank than the other. Fraudulent activities include one partner removing the other's link without notice, creating a link that is not visible by the search engines, and putting links on a separate domain.

Purchasing Links

Because the purchase of links for promoting one's search engine ranking is explicitly banned, this method of link building is used at considerable risk. Nevertheless, the sale of links is inevitable given the inherent value of links, and the ability to choose which sites will link to you is certainly a tempting proposition for SEO providers and content providers.

Directory Submission

One of the best ways to get quality inbound links is by submitting your site to the Open Directory Project [15]. There are many directories of websites on the Internet, but the ODP is unique in that it is the de facto standard for website directories. Its directory is used to supply links to other directories including Google and Yahoo. So, a single submission to the ODP can often result in at least three high quality links.

Not All Links Created Equal

As content providers focus on gathering links it is important to note that not all links are equal. As mentioned previously, an inbound link from a high PageRank page is of more value than an inbound link from a low PageRank page. There is also some speculation that diversity in geographical location and domain type may also be an important factor.

Chapter 4

Research Design for the Analysis of Search Result Rankings

4.1 Research Goals

Although a comprehensive "reverse engineering" of the Google search ranking algorithm would be desirable, the breadth of such a project is far beyond the scope of this thesis. Instead, this thesis hopes to ascertain the relative importance of selected on-page attributes by measuring the association of a subset of on-page factors with the search rank of the page for a given query. A second goal of the project is to provide a framework for on-going research in this area, which consists of a collection of Ruby scripts for gathering and analyzing the data.

4.2 Case for Correlation

There are several methods of gathering and analyzing the data that could be used

to achieve the objectives of identifying the on-page attributes that are important in ranking Google search results. Because we are studying the relationship between several variables (on-page attributes) and a single other variable (search rank), multiple regression seems to be a possible choice. Similarly, feature extraction algorithms may be a possibility as well. However, there appears to be a dearth of multiple regression or feature selection algorithms suitable for dichotomous and ordinal data. Moreover, a desirable objective of this thesis is to provide a self-contained framework for collecting and analyzing the data, so the chosen algorithm would have to be non-proprietary and limited in complexity. Ultimately, a correlation analysis was chosen, as the correlation is the preferred statistical technique for measuring the relationship between two variables and is much simpler to implement and interpret than other statistical and machine learning techniques.

4.3 Description of On-Page Attributes

There are hundreds of possible attributes, or factors, that could be studied, so a subset of some of the more obvious ones were chosen. Each of these attributes are listed below with a short description. Attributes were measured as a dichotomous variable with a 0 or a 1 indicating that the keyword was not present or present, respectively. Keyword frequency was measured as a continuous variable, but converted to a dichotomous variable for comparison purposes.

Keyword Frequency (keyword_freq_d)

We've measured keyword frequency as the number of times the keyword appears in the page between word boundaries divided by the length of the page in characters. It is likely that Google calculates keyword frequency differently (perhaps removing html tags), but this method is simpler and should be a close estimate to a more complex method.

In Bold (in_b)

Dichotomous variable set to 1 if the keyword is contained within one or more bold tags (<b>), otherwise set to 0.

In H1 (in_h1)

The header 1 tag (<h1>) is the largest header tag in the HTML specification. This dichotomous variable is set to 1 if the keyword is contained within one or more header 1 tags, otherwise set to 0.

In H2 (in_h2)

Dichotomous variable set to 1 if the keyword is contained within one or more header 2 tags (<h2>), otherwise set to 0.

In H3 (in_h3)

Dichotomous variable set to 1 if the keyword is contained within one or more header 1 tags (<h3>), otherwise set to 0.

In Strong (in_strong)

The strong tag causes text to stand out similar to the bold tag. It is a

dichotomous variable set to 1 if the keyword is contained within one or more strong tags (), otherwise set to 0.

In Italics (in_i)

Dichotomous variable set to 1 if the keyword is contained within one or more italics tags (<i>), otherwise set to 0.

In Underline (in_u)

Dichotomous variable set to 1 if the keyword is contained within one or more underline tags (<u>), otherwise set to 0.

In Select (in_select)

The select tag is used to create drop downs and multi-select boxes. This dichotomous variable is set to 1 if the keyword is contained within one or more select tags (<select>), otherwise set to 0.

In Image Src (in_img_src)

The source of an image tag is technically the "src" attribute of the "img" HTML tag. It represents the URL of an image referenced in an HTML page. This dichotomous variable is set to 1 if the keyword is contained within the src attribute of one or more image tags, otherwise set to 0.

In Image Alt (in_img_alt)

The alt attribute of an image tag is used to specify alternate text which will be shown if the image is not able to be downloaded. This has been a long-time

favorite place to stuff important keywords for the search engines to find. This dichotomous variable is set to 1 if the keyword is contained within the alt attribute of one or more image tags (<img>), otherwise set to 0.

In Link Href (in_link_href)

This refers to the href attribute of an anchor tag (<a>), more commonly referred to as a link. The href attribute represents the URL of a link in which a browser will be directed when clicking on the link. This is a dichotomous variable set to 1 if the keyword is contained within the href attribute of one or more anchor tags, otherwise it is set to 0.

In Link Text (in_link_text)

The link text is the text found between the opening and closing anchor tags (<a>,). It is the portion of the link that is visible when the HTML is rendered by the web browser. This variable is also dichotomous, having the value of 1 if the keyword is enclosed within one or more anchor tags, otherwise set to 0.

In Input (in_input)

Input tags represent form elements such as a text box, button, file upload, radio button, check box, or hidden field. This variable indicates the presence of the keyword in the name, src, value, id, or class attributes of any input tags. If the keyword is present, then the value is set to 1, otherwise it is set to 0.

In Title (in_title)

Text within the title tag is typically displayed prominently at the top of most browser windows. It is also displayed for search results on most search engines. Consequently, it is suspected that the title is an important area for content providers to target with descriptive keywords. As with most of the other variables, this is a dichotomous variable set to 1 if the keyword is contained within the title tag, otherwise it is set to 0.

In URL (in_url)

The URL is the subdomain, domain, path, and query string that uniquely identifies the page on the Internet. "In URL" is also a dichotomous variable set to 1 if the keyword is contained within the URL of the page, otherwise set to 0.

4.4 Rank-Biserial Correlation

The product-moment correlation coefficient (Pearson r) and its derivatives are the standard correlation techniques; however, the Pearson r is used when both variables are on an interval/ratio scale. In this case, the data pairs include the ordinal search ranking and a nominal dichotomous score of 0 or 1. Because of this, the appropriate correlation is the Rank-Biserial Correlation, which is used with ordinal vs dichotomous variables [4]. It has been in use for over 50 years showing equivalence to the Spearman r, which is used when both pairs are

ordinal. This coefficient denoted by rb, has the following formula:

$$rb = (2/n)(Y_0 - Y_1), \text{ where}$$

n is the number of ranked entries,
Y_1 is the mean rank of those scoring 1 on the dichotomy, and
Y_0 is the mean rank of those scoring 0 on the dichotomy.

The coefficient is calculated in the Ruby script file correlation.rb in Appendix H.

Although this study does not use the product-moment correlation coefficient, the correlation.rb file also contains a method for calculating this coefficient, as it may be used in future extensions and/or enhancements to the current study in which continuous data is measured.

4.5 Limitations and Concerns

This section discusses some of the limitations to the approach used in this thesis as well as other related-concerns. As with all statistics, the observation of extraneous factors can be just as important as the data itself.

Limited Expressiveness of Dichotomous Variables

Future research in this area may benefit from analyzing data on an interval/ratio scale rather than as dichotomous variables because of the inherent expressiveness of interval/ratio data. For example, rather than simply measuring

whether a keyword is present in an HTML tag, it may be more informative to measure the position of the keyword in the tag, or measure the proportion of the text within a tag that is the keyword (keyword length / tag text length). Moreover, another useful metric could be the keyword frequency within a tag. There are many possible variations, all of which would be useful for further research, yet beyond the scope of this thesis.

Truncated Range

Inaccurate correlation values can be the result of a truncated range, which occurs when the sample range is significantly smaller than the actual population range [17]. In the case of this study, the range may become truncated by Google in a couple of possible ways. The first is that, by default, a Google search filters results by weeding out similar pages as well as more than two pages coming from the same domain. Technically, this does truncate the range; however, I believe this form of truncation helps achieve more accurate correlation values. Without filtering, many of the results come from the same website, which frequently implies use of the same layouts, coding and styling guidelines, and even the same content management system. The above three factors may all cause high correlations between variables being studied, which will make it difficult to determine the true correlation between the variable and the search rank. For example, a website using a content management system may force all page titles to be displayed in an h1 tag as well. If this website has many pages in the search results, then the correlation between title and search rank may skew the correlation between the h1 tag and search rank (and vice versa). For this reason, filtering was used for the searches in this study.

The second way in which Google may truncate the range is that Google search only displays a maximum of 1000 search results for any query, and usually less than this. For example, a Google search for Java (restricted to html file type only) yields over 272 million results; however, even with filtering turned off, there are only 962 results shown. This severely truncated range is an issue that is dealt with in Section 5.2 when discussing the query selection process adhered to in this project. Through query selection, the affect of the truncated range can be minimized.

Nonlinearity

Another threat to the accuracy of a correlation is nonlinearity of the regression line. Not all relationships are linear, and a strong correlation can be masked by a nonlinear relationship [17]. Keyword frequency is the variable within this study that is most likely to be affected by nonlinearity, as there is speculation that very high keyword frequencies may be considered 'keyword spamming', and consequently, the page may be penalized.

Chapter 5

Data Collection and Analysis Process

As mentioned previously, the goal of this thesis is not only to provide SEO data for content providers, but also to develop a set of tools and processes for continuing and extending this research in the future. This chapter discusses the processes and tools used in collecting and analyzing the data. All code is developed using Ruby, a dynamic programming language known for developer productivity [16].

5.1 Storing the Data

MySQL was chosen as the database for storing the data because it is free and well-documented. All data is stored in a single table named 'search_results', and each record represents a ranked search result and information related to it. The table structure is shown below.

```
TABLE `search_results` (
  `id` int(11) NOT NULL auto_increment,
  `query` varchar(100) NOT NULL default '',
  `total_results` int(11) NOT NULL default '0',
  `search_rank` int(11) NOT NULL default '0',
  `url` varchar(200) NOT NULL default '',
  `title` varchar(250) NOT NULL default '',
  `snippet` text NOT NULL,
  `created_at` datetime NOT NULL default '0000-00-00 00:00:00',
  `updated_at` datetime NOT NULL default '0000-00-00 00:00:00',
  `response_code` varchar(45) NOT NULL default '',
  `response_body` longtext NOT NULL,
  `keyword_freq` float NOT NULL default '-1',
  `in_b` int(11) NOT NULL default '-1',
  `in_h1` int(11) NOT NULL default '-1',
  `in_h2` int(11) NOT NULL default '-1',
  `in_h3` int(11) NOT NULL default '-1',
  `in_strong` int(11) NOT NULL default '-1',
  `in_i` int(11) NOT NULL default '-1',
  `in_u` int(11) NOT NULL default '-1',
  `in_select` int(11) NOT NULL default '-1',
  `in_img_src` int(11) NOT NULL default '-1',
  `in_img_alt` int(11) NOT NULL default '-1',
  `in_link_href` int(11) NOT NULL default '-1',
  `in_link_text` int(11) NOT NULL default '-1',
  `in_input` int(11) NOT NULL default '-1',
  `in_url` int(11) NOT NULL default '-1',
  `in_title` int(11) NOT NULL default '-1',
  `content_type` varchar(45) NOT NULL default '-',
  `new_rank` int(11) NOT NULL default '-1',
  `keyword_freq_d` int(11) NOT NULL default '-1',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Figure 5-1-1. Database Schema for Search Result Data

The ActiveRecord package is used heavily in this project, so it will be a key dependency when utilizing these data processing scripts.

5.2 Conducting the Search

The first step in gathering data is to conduct a query and record the search results, yet according to the Google Terms of Service, “You may not send automated queries of any sort to Google's system without express permission in advance from Google” [8]. Fortunately, Google provides a SOAP API for legally querying the search engine. This project uses a custom Ruby script titled do_search.rb (see Appendix B) for conducting a Google query and storing search result data into the database. It requires the Ruby-Google wrapper as a dependency [12].

Query Selection

An important issue in regards to the search is query selection, after all, this is essentially the means in which the sample is collected. Conventional sampling wisdom tells us that the word used for each query should be randomly chosen from all possible words; however, that approach will yield inaccurate results due to the truncated range problem discussed in Section 4.5. When first gathering test data for this project, it was quickly realized that the majority of words, when queried as a single term, produce search results that will be severely truncated. The example given earlier is the search for 'Java' on Google that yields 272 million results, but only 962 results are displayed. In that particular case, only about 1 out of every 280,000 results was included in the search results. Consequently, over 80% of the search result titles contained the word 'Java', which in turn caused a low correlation value between the title and the search rank. To limit truncation, queries were selected from a list of uncommon words

found at [21]. Because this study examines variables related to images (In Image Src and In Image Alt), some queries were selected that were more likely to contain images. For these queries, words were selected from a list of uncommon animals [13].

Despite the importance of limiting truncation, care must be taken to select search terms that return sufficient results. From the test data used, it appears that more than 300 search results (after filtering and re-ranking) are necessary so that correlation coefficients can be determined for most attributes. Even with more than 300 search results, an attribute will sometimes have too few counts for one value in the dichotomy. For example, the **in_img_alt** attribute may have all values set to 0 and none set to 1. In this case, the correlation can not be calculated. Similarly, if there is just a single count for a value in the dichotomy, the coefficient can be skewed, as it is determined solely by the rank of that one value. For these reasons, only queries returning more than 300 results (after filtering and re-ranking) were used.

Filtering

By default, a Google search is conducted with filtering on, which eliminates similar results and prevents more than two results from the same domain ending up in the search results. Filtering should not be confused with the automatic truncation to 1000 results or less, which occurs regardless of whether filtering is turned on. As mentioned in Section 4.5, filtering mode was turned on while collecting search result data to prevent an increase in correlations between variables, and to decrease the effect that one domain's pages could have on the

data.

Another form of filtering was used in that non-HTML file types were filtered out of the queries by use of special query directives. Specifically, each query was appended with " -filetype:pdf -filetype:doc -filetype:ps -filetype:xls -filetype:txt -filetype:ppt -filetype:rtf" to prevent non-HTML file types from being returned in the search results. This is important because this thesis is concerned only with on-page attributes found in HTML.

5.3 Gathering HTML Pages

As this is a study regarding on-page factors, the HTML of the pages returned from the search results must be gathered so that data can be extracted from each page. For complete accuracy, it is important that the page appear just as it did when crawled by the Google. Fortunately, the Google SOAP API provides the ability to retrieve the page from its cache, as it appeared when last indexed by Google. As the search script queries Google and iterates over the results, the cached page is requested for each result and stored in the result's record for later processing.

5.4 Re-Ranking Search Results

As the search result data is collected there is a very small percentage of results (<

1%) that are not able to be saved. This is sometimes due to the cached page not being available, or an error in a Ruby parsing library. Whatever the case may be, this creates gaps in the ranking sequence, which would produce an inaccurate correlation. Consequently, the script re_rank.rb (Appendix C) gives new ranks to the search results by first removing invalid records and then ranking the results according to their original search rank.

5.5 Extracting On-Page Data

At this step in the data collection process, each newly ranked search result has been stored in a database record along with the full HTML of the page. The Ruby script do_gather_data.rb (Appendix D) iterates over each of the newly ranked records for the given query and calculates the on-page data for each search result page. For example, the script parses the <title> tag in the HTML and checks for the presence of the query term. If it is present in the title tag, then the in_title column is set to 1, otherwise it is set to 0. This process is done for each attribute on each page for the given query until each of the records have been processed. At that point, correlation analysis can be conducted between each attribute and the page's search rank for the given query.

5.6 Correlation Analysis

After the data is collected, the Rank-Biserial correlation coefficient [4] is

calculated for each on-page attribute and the search rank. The Ruby script do_correlation.rb (Appendix F) is used to iterate over the data for a given query and output the correlation coefficient for each attribute / rank pair. Besides being able to calculate the Rank-Biserial correlation coefficient, correlation.rb (Appendix H) contains code to calculate the Pearson Product-Moment correlation coefficient in case future extensions to this research would measure continuous data (rather than dichotomous).

5.7 Process Summary

The table below summarizes the data collection and analysis process and highlights the corresponding source code.

Database ORM Layer	search_result.rb (Appendix A)
Conduct the Search	do_search.rb (Appendix B)
Gather the HTML	do_search.rb (Appendix B)
Re-Rank Results	re_rank.rb (Appendix C)
Extract On-Page Data	do_gather_data.rb (Appendix D), page_attributes.rb (Appendix E)
Correlation Analysis	do_correlation.rb (Appendix F), do_correlation_by_attribute.rb (Appendix G), correlation.rb (Appendix H)

Table 5-7-1. Source Code Summary for Data Collection and Analysis

Chapter 6

Correlation Analysis

Data was gathered for 10 search queries and a rank-biserial correlation was used to analyze the relationship between each of the dichotomous attributes and the search rank. Because Keyword Frequency is not dichotomous, it was assigned high and low dichotomous values. A value of high (1) was given if the value was above the mean and a value of low (0) was given if the value was equal to or less than the mean.

6.1 Results by Query

The table below is an example of a result table for an individual query. This table lists the attributes/search rank correlations for one of the queries, as well as the frequency counts for each of the values in the dichotomy. Result tables for the remaining queries can be found in Appendix I.

Attribute	Correlation	0/Low	1/High
in_select	0.9583	336	1
in_title	0.9517	318	19
in_input	0.9325	330	7
in_h1	0.8720	322	15
in_h2	0.8664	333	4
in_url	0.8178	321	16
keyword_freq_d	0.6936	291	46
in_strong	0.6707	328	9
in_i	0.5818	330	7
in_b	0.4915	313	24
in_img_alt	0.3411	330	7
in_h3	0.1049	329	8
in_img_src	-0.0747	332	5
in_link_href	-0.0829	273	64
in_link_text	-0.1957	250	87
in_u	-0.7284	335	2

Table 6-1-1. Results for Query 'adhibit'

6.2 Results by Attribute

The table below list the queries and corresponding attribute/search rank correlation for the attribute 'keyword_freq_d' (keyword frequency as a dichotomous variable).

Query	Correlation	0/Low	1/High
adhibit	0.6936	291	46
agrestic	0.4690	467	82
appetency	0.5896	464	63
kouprey	0.7540	399	134
mystagogue	0.7694	512	83
nouthetic	0.6937	546	102
numbat	0.3976	328	314
pacarana	0.6815	330	54
paradoxology	0.6886	337	32
phascogale	0.6364	355	117
Avg. Correlation:	**0.6373**		

Table 6-2-1. Results for Attribute 'keyword_freq_d'

To conserve space, the result tables for the other attributes have been moved to Appendix J.

6.3 Results Summary

One of the challenges of this research is that many of the attributes have very small frequency counts for one side of the dichotomous variable. For example, the attribute **in_img_alt** has just a single search result with a value of 1 for the query 'paradoxology'. This means that only 1 of the 369 search results for paradoxology contained the keyword in the ALT attribute of the img HTML tag. Unfortunately, having so few results on one side of a dichotomous variable can skew the correlation's accuracy. To diminish the effects of this, the average correlation for each attribute across all queries was recorded. The following table summarizes these values in sorted order.

Attribute	**Mean Correlation**
in_title	0.8118
in_url	0.7565
in_h1	0.7135
in_input	0.6785
in_select	0.6494
keyword_freq_d	0.6373
in_h2	0.5509
in_h3	0.5031
in_img_src	0.4799
in_strong	0.4633
in_b	0.3956
in_img_alt	0.3715
in_i	0.3466
in_link_href	0.2602
in_link_text	0.1263
in_u	-0.0969

Table 6-3-1. Summary of Mean Correlation per Attribute

Interpretation of this data as it pertains to content providers is discussed in the next chapter.

Chapter 7

Conclusions and Future Work

Special care must be taken in the interpretation of correlational data. Although many of the attributes showed a strong correlation with search rank, it is necessary to also consider that the association could be caused by a third un-tested (or unknown) variable that the attribute and the search rank are associated with.

7.1 Conclusions for Content Providers

Now that the data has been collected and statistically analyzed, it is time for the rewarding part of this research – practical application of the results. How can content providers use this information to increase their visibility? How should this data be interpreted? First, it is important to note that all but one or two attributes (**in_u**, **in_link_text**), seem to have little or no relationship with search rank. Also, many of the attributes have extremely high correlations with search rank, which indicates that the attributes must be correlated highly with each other. The need for partial correlations is discussed further below in section 7.2.2; however, even without partial correlations, content providers can still err

on the side of caution and simply include keywords in several of the attributes, especially the ones that are the most highly correlated with search rank.

The results do seem to suggest that content providers should include the keyword in a wide variety of tags. For example, the correlation that exists with tags that are not as common (e.g., input and select) seems to suggest that Google may be favoring results that include a broad usage of the tags throughout the page. The fact that so many of the attributes have a positive correlation with search rank also lends credibility to this approach. Furthermore, a broad usage of the search term throughout the page will increase the keyword frequency, which is also positively correlated with search rank. Perhaps future research in this area could examine this approach empirically by creating a variable that represents how many different HTML tags the keyword was found in. The correlation of this variable with search rank could then be measured.

For content providers deciding what Content Management System (CMS) to use, or how to structure their web application, these results indicate the need to control what keywords can go into a URL. Given the high correlation between the **in_url** attribute and search rank, it is highly probable that the inability to insert keywords in a URL will put the content provider at a disadvantage. The same could be said about some of the other attributes (e.g., **in_title**), but nearly every CMS already allows control over the other tags, so it is not worth mentioning.

Another valuable conclusion to be drawn from the results of this research is that

content providers should place emphasis on the on-page factors first and foremost. The title alone has a correlation of 0.81 with search rank, which means the proportion of variance in common is 0.66 (r^2). That means that only 34% of the variance of the search rank can be accounted for elsewhere. That does not leave a lot of room for off-page factors like PageRank. Of course, 34 percent is still a non-trivial amount, but this research puts the off-page attributes in perspective. Content providers should not lose sight of the importance of on-page attributes when focusing on off-page factors.

Finally, it's also important to keep in mind that this thesis represents a starting point and a framework for delving deeper into this topic. The source code in the Appendices of this thesis provide all that is necessary to extend this topic in new directions. Some of the possibilities for future work are discussed below.

7.2 Future Work

Because search engines are increasingly using broader factors in determining the ranking of a search result, this research is the “tip of the iceberg” in furnishing content providers with useful information regarding search engines. Listed below are various ways in which this research could be extended or modified.

7.2.1 Analyze Additional Attributes

Further analysis could be done on various attributes (both continuous and dichotomous) and their relation to search rank. All possible attributes are too

numerous to list, but some of the most promising are included below.

Number of Links on a Page

Even the Google website advises that webmasters keep the number of links per page to less than 100 [10]. The fact that they keep track of this makes this a good variable to test.

Total Length of a Page

It has been suggested that the total length of the HTML in a page could be a factor in ranking [3].

Total Length of the Body Text (non-HTML portions)

Some suggest that the number of words in a page are an important attribute of a page [11].

Keyword Frequency Within each HTML tag

There is a correlation between increased keyword frequency for a page and its ranking, but what about keyword frequency within a tag? Is it helpful to put the same keyword more than once in the title of a page?

Position of Keyword Within Each Tag

Many claim that the keyword should be as close to the beginning of a tag that contains it as possible [11], [3]. These claims could easily be investigated with the set of tools used in this thesis.

Length of Text in a Tag Containing Keyword

Some have suggested that the number of characters within tags may be taken into consideration by the search engines as well [11]. This would probably be best analyzed using a non-linear measure of association.

Date Last Updated / Indexed

The major search engines keep track of when a page was last updated and many speculate that pages updated more often are ranked higher. If the last update of pages can't be gathered, then perhaps the date of when the page was last indexed could be used as an estimate.

In Noscript Tags

It is believed that text between the <noscript> </noscript> tags is indexed and may boost a page's ranking.

In CSS / Javascript Filenames

The filename of an external CSS or Javascript file may be a good place to insert keywords [3].

In Javascript Code

Although most search engine crawlers do not retrieve external javascript files, it is possible that they would take into consideration keywords found within Javascript code or variable names.

In CSS Attributes

The use of CSS is very widespread and many tags include CSS class and id attributes. The values of these attributes may be monitored by the search engines for keywords.

Letter Case

Spelling a word in all capital letters can certainly make it stand out. Although, there is little or no evidence suggesting the use of this, it seems to be a logical on-page attribute for search engines to use.

Number of Backlinks

PageRank has become an important factor in search ranking and the number and importance of backlinks is its primary component; however, it would be useful to see just how important the number of backlinks is to the search rank.

Number of Backlinks Containing Keyword in Link Text

Link text of backlinks has long been known as a factor in search ranking, but the extent of this has not been quantified.

Number of Backlinks Containing Keyword in Title

Examining the link text of backlinks is common practice, but it would be just as easy for search engines to store the title of the linking page.

Geodiversity of Backlinks

Search engines must not only sort documents by relevance, but they must

also attempt to thwart individuals attempting to grow their backlinks through collusion with other website owners. Links coming from a wide variety of geographical locations is an indication that the links were probably not added by the same person or group.

7.2.2 Partial and Multiple Correlations

One of the difficulties in interpreting the results of this studies lies with the fact that many of the variables may be highly correlated with each other. For example, it is common for websites to use the title of a page in header text at the top of the page as well. This creates a high correlation between the in_title variable and the in_h1, in_h2, and in_h3 variables. To solve this problem, partial correlations can be calculated, which represent the correlation between two variables if the third were held constant [18]. Depending on the type of data measured, future studies may also consider using multiple correlation and/or feature selection techniques.

7.2.3 Analyzing Multiple Keywords

To narrow the scope of this thesis, research was done using single-word queries; however, it would be useful to perform similar analysis on queries with multiple terms. Instead of using dichotomous variables for attributes, perhaps the attributes could be continuous with the value being the number of terms from the query that appear within the particular tag. For example, a search for “excellent Ruby developers” (without quotes), may include a title that includes the term

Ruby and developers, but not the word excellent. In that case, the variable **in_title** would have a value of 2, because 2 of the 3 terms were present in the title tag.

7.2.4 Inclusion of Other Search Engines

Google is the industry leader, but they are just half the market. Yahoo commands significant market share as well. Moreover, Microsoft is a distant third in the search engine race, but it's difficult to count them out considering their deep pockets and ability to leverage the Windows OS. The Internet can change very quickly, thus it would be beneficial to content providers to perform similar research on other search engines, and compare the results to those of this study.

Bibliography

[1] Alexa Internet, Inc. "Alexa Web Search," *http://www.alexa.com*, 2007.

[2] Sergey Brin and Larry Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proceedings of the Seventh International Conference on World Wide Web, 7*, pp. 107-117, 1998.

[3] Tara Calishain and Rael Dornfest. *Google Hacks*, O'Reilly & Associates, 2003.

[4] Edward E. Cureton. "Rank-Biserial Correlation," *Psychometrika,* vol. 21, no. 3, pp. 287-290, September 1956.

[5] ECT News Network, Inc. "Google Still the Leader," *http://www.ecommercetimes.com/story/54362.html*, November 21, 2006.

[6] Eurekster, Inc. "Eurekster Swiki Home," *http://swicki.eurekster.com/*, 2007.

[7] Google Inc. "Frequently Asked Questions – File Types," *http://www.google.com/help/faq_filetypes.html*, 2007.

[8] Google Inc. "Google Privacy Center: Terms of Use," *http://www.google.com/terms_of_service.html*, 2007.

[9] Google Inc. "Google Toolbar," *http://toolbar.google.com*, 2007.

[10] Google Inc. "Webmaster Help Center – Webmaster Guidelines," *http://www.google.com/support/webmasters/bin/answer.py?answer=35769*,

2007.

[11] Peter Kent. *Search Engine Optimization for Dummies*, 2nd Edition, Wiley Publishing, Inc., 2006.

[12] Ian Macdonald. "Ruby-Google 0.6.0," *http://raa.ruby-lang.org/project/ruby-google/*, February 8, 2006.

[13] Paul Massicot. "Animal Index by Species Name," *http://www.animalinfo.org/spec_ind.htm*, October 28, 2003.

[14] Microsoft Corporation. "MSN Historical Timeline," *http://www.microsoft.com/presspass/press/2002/nov02/11-08MSN8GlobalTimeLine.mspx*, June 2005.

[15] Netscape Communications Corporation. "Open Directory Project," *http://dmoz.org/*, 2007.

[16] Ruby Users Group. "Ruby Programming Language," *http://www.ruby-lang.org/*, 2007.

[17] Chris Spatz. *Basic Statistics: Tales of Distributions*, Brooks/Cole Publishing Company, 1997.

[18] Murray R. Spiegel. *Schaum's Outline of Theory and Problems of Statistics*, Schaum Publishing Company, 1961.

[19] Danny Sullivan. "How Search Engines Rank Web Pages," *http://searchenginewatch.com/showPage.html?page=2167961*, July 31, 2003.

[20] Danny Sullivan. "Searches Per Day," *http://searchenginewatch.com/showPage.html?page=2156461*, April 20, 2006.

[21] Thomas Love Peacock Society. "Uncommon Words in the Works of Peacock," *http://www.thomaslovepeacock.net/words.html*, 2001.

[22] Lee Underwood. "A Brief History of Search Engines," *http://www.webreference.com/authoring/search_history/*, August 18, 2004.

[23] Aaron Wall. "History of Search Engines: From 1945 to Google 2006," *http://www.search-marketing.info/search-engine-history/*, November 19, 2006.

[24] Yahoo! Inc. "Keyword Selector Tool," *http://inventory.overture.com/d/searchinventory/suggestion/*, 2007.

[25] Yahoo! Inc. "Yahoo! Search Blog: Our Blog is Growing Up – And So Has Our Index," *http://www.ysearchblog.com/archives/000172.html*, August 8, 2005.

Appendix A
search_result.rb

```
# ORM layer for data

require 'rubygems'
require_gem 'activerecord'

ActiveRecord::Base.establish_connection(
   :adapter => "mysql",
   :host => "localhost",
   :database => "thesis",
   :port => 5000,
   :username => 'thesis',
   :password => "abc123")

class SearchResult < ActiveRecord::Base
end
```

Appendix B
do_search.rb

```
require 'google'
require 'base64'
require File.dirname(__FILE__) + '/../search_result'

#########################################################
# Performs a Google search and stores the results in the
# search_results table.  To fully populate the search_results table
# other scripts including grab_html.rb will need to be run
# - also grabs cached page from Google when available
# - prevents duplicates from being added
#########################################################

KEY = 'kTsudvdQFHJ4CblCiM3P6oaufysV9/uL'
google = Google::Search.new(KEY)
query = ARGV.shift
total_results = ARGV.shift.to_i
if !query || !total_results
  p "please enter a query and the total # of expected " +
        "search results  (e.g., java 1000)"
  exit
end
query_suffix = ' -filetype:pdf -filetype:doc -filetype:ps -filetype:xls
-filetype:txt -filetype:ppt -filetype:rtf'


step = 10
start_results = []
0.step((total_results - step), step) {|i| start_results << i }

already_added = Hash.new # for lookup of urls to prevent duplicates
start_results.each do |start_result|
  i = 0
  q = nil
  retry_count = 0
  filter = true

  begin
    q = google.search(query + query_suffix, start_result, 10, filter)
```

```
  rescue => e
    p e
    if retry_count > 5
      next
    else
      retry_count += 1
      sleep 3
      p "retry......."
      retry
    end
  end

  p "results: #{q.resultElements.length}"
  q.resultElements.each do |result|
    begin
      i += 1
      printf "\nResult # %d\n", i + start_result
      print "url = #{result.url}\n"
      if already_added.has_key?(result.url)
        p "** already added (duplicate)"
      else
        already_added[result.url]=result.url
        cached_retry_cnt=0
        begin
          cached = Base64.decode64(google.cache(result.url))
          # remove Google cache header
          cached.gsub!(/^(.*?)<\/td><\/tr><\/table><\/td><\/tr><\/table>\s*<h
r>/im, '')
        rescue => b64e
          p b64e
          p "retry cached..."
          cached_retry_cnt+=1
          retry if cached_retry_cnt < 6
        end

        sr = SearchResult.new
        sr.query = query
        sr.total_results = q.estimatedTotalResultsCount
        sr.search_rank = i + start_result
        sr.url = result.url
        sr.title = result.title
        sr.snippet = result.snippet
        if cached && cached.length > 50
          sr.response_code = 200
          sr.response_body = cached
```

```
        end
        if !sr.save
          print "SAVE FAILED!!!!!!!!!!!!!!!!!!!!!!!!"
        else
          p 'saved search result'
        end
      end  # end !already_added block

    rescue => e
      p e
    end
  end # end each result
  p '-----------------------------------------------'
  sleep 3

end # end for each search
```

Appendix C
re_rank.rb

```
require File.dirname(__FILE__) + '/../search_result'
############################################################
# This script re-ranks the data by setting the new_rank field
# for valid data.  Entries w/o a new_rank will be skipped in the analysis
# Why this is necessary:
# - this thesis covers only HTML pages, so pdf,
#   xls, and non-HTML file types must be removed from the analysis
# - Rank-Biserial Correlation Coefficient does not
#   allow "holes" in the ranks, so data must be re-ranked
############################################################

query = ARGV.shift
if !query
  p "please enter a query"
  exit
end
select_str = "select id,new_rank FROM search_results WHERE query LIKE
'#{query}' AND response_code = 200 AND char_length(response_body)>200  ORDER
BY search_rank"
results = SearchResult.find_by_sql(select_str)
rank = 1
results.each do |result|
  begin
    p result.id
    result.new_rank = rank
    throw Exception.new("ERROR saving search results #{result.id}") if !
result.save
    rank += 1
  rescue => e
    p e
    retry
  end
end
```

Appendix D
do_gather_data.rb

```
require File.dirname(__FILE__) + '/../search_result'
############################################################
# After the search results have been generated and the HTML gathered,
# this can be run to extract on-screen data and store in db
#
# Eventually, this script will:
# 1) conduct a search
# 2) gather all search results in the db
# 3) grab the HTML for each search result and store in db
# 4) calculate on-screen features for each result
############################################################

query = ARGV.shift
if !query
  p "please enter a query"
  exit
end
conditions = "query LIKE '#{query}' AND new_rank > 0"
SearchResult.find(:all, :conditions => conditions).each do |sr|
  begin
    pa = PageAttributes.new
    pa.parse(sr.response_body, sr.query)
    sr.url_pos = sr.url.index(sr.query) || 500
    sr.in_url = sr.url.index(sr.query) ? 1 : 0
    sr.title_pos = pa.title_pos || 500
    sr.in_title = pa.title_pos ? 1 : 0
    sr.keyword_freq = pa.keyword_freq * 10.0
    tags = ['b', 'h1', 'h2', 'h3', 'strong', 'i', 'u', 'select', 'img_src',
'img_alt', 'link_href', 'link_text', 'input']
    tags.each {|t| eval('sr.in_' + t + ' = pa.in_' + t+ '?(sr.response_body,
sr.query) ? 1 : 0')}
    sr.save
    p "Saved #{sr.id}"
  rescue => e
    p e
  end
end
```

Appendix E
page_attributes.rb

```
##########################################
# This class is used to extract the on-page
# attributes for a given
# HTML page as it relates to a given keyword
# - used by do_gather_data.rb
##########################################

class PageAttributes

  attr_reader :title_pos, :body_pos, :keyword_freq

  def initialize()
    %w(b strong h1 h2 h3 i u select).each do |tag|
      meth = %Q{def in_#{tag}?(html, keyword)
              return is_in_tag?('#{tag}', html, keyword)
            end}
      self.instance_eval(meth)
    end
    reset
  end

  def parse(html, keyword)
    reset
    safe do
      @title_pos = nil
      if html =~ /<title>(.*?)<\/title>/mi
        @title_pos = $1.index(/#{keyword}/mi) if $1
      end
    end
    @body_pos = nil
    safe do
      if html =~ /<body[^>]*>(.*)$/mi
        @body_pos = $1.index(/#{keyword}/mi) if $1
      end
    end

    safe { @key_in_bold = in_b?(html, keyword)}
    safe { @keyword_freq = calc_keyword_freq(keyword, html)}
  end
```

```
# simple wrapping of code to prevent tedious
# checking for nil objects and other errors
def safe
  begin
    yield
  rescue => e
    p e.to_s
  end
end

def reset
  @title_pos = nil
  @keyword_freq = nil
  @body_pos = nil
end

def calc_keyword_freq(keyword, text)
  count = 0
  safe { text.scan(/\b#{keyword}\b/i) { |w| count += 1; } }
  return 0.0 if count==0
  return count.to_f/text.length.to_f
end

def in_img_src?(html, keyword)
  is_in_tag_attribute?('img', 'src', html, keyword)
end

def in_img_alt?(html, keyword)
  is_in_tag_attribute?('img', 'alt', html, keyword)
end

def in_link_href?(html, keyword)
  is_in_tag_attribute?('a', 'href', html, keyword)
end

def in_link_text?(html, keyword)
  is_in_tag?('a', html, keyword)
end

def in_input?(html, keyword)
  name = is_in_tag_attribute?('input', 'name', html, keyword)
  src = is_in_tag_attribute?('input', 'src', html, keyword)
  value = is_in_tag_attribute?('input', 'value', html, keyword)
  the_id = is_in_tag_attribute?('input', 'id', html, keyword)
```

```
    the_class = is_in_tag_attribute?('input', 'class', html, keyword)
    return name || src || value || the_id || the_class
  end

  private
  def is_in_tag?(tag, html, keyword)
    # this doesn't work - why not???
    #return true if html =~
/<#{tag}[^>]*?>.*?(?!<\/#{tag}>).*?#{keyword}.*?<\/#{tag}>/mi
    html.scan(/<#{tag}(?: [^>]*?)*>(.*?)<\/#{tag}>/mi) do |match|
      return true if $1 =~ /#{keyword}/mi
    end
    return false
  end

  def is_in_tag_attribute?(tag, attr, html, keyword)
    html.scan(/<#{tag}[^>]*\/?>/mi) do |match|
      #p 'match: ' + match
      return true if match =~ /#{attr}="([^">])*#{keyword}([^">])*/mi
    end
    return false
  end

end
```

Appendix F
do_correlation.rb

```
##########################################
# Correlational analysis with results separated
# by query.  External Ruport reporting library
# is used, if it is found on the system.
# NOTE: to use Ruport, the following line had to be commented out
# from within the ruport source code:
#   r.gsub!(/\A.{#{width+1},}/) { |m| m[0,width-2] + ">>" }
# ( ruport/format/text.rb:73 )
##########################################

require File.dirname(__FILE__) + '/correlation'
require File.dirname(__FILE__) + '/../search_result'

ruport = true
begin
  require 'ruport'
rescue => er
  p er
  ruport=false
end


queries = [:adhibit, :agrestic, :appetency, :kouprey, :mystagogue,
:nouthetic, :numbat, :pacarana, :paradoxology, :phascogale]

attrs = [:keyword_freq_d, :in_title, :in_url, :in_b, :in_h1,
  :in_h2, :in_h3, :in_strong, :in_i, :in_u, :in_select,
  :in_img_src, :in_img_alt, :in_link_href, :in_link_text, :in_input]


#query = ARGV.shift
queries.each do |query|
  data_sets = {}
  attrs.each {|n| data_sets[n] = Array.new}
  begin
    conditions = "query LIKE '#{query}' AND new_rank>0"
    SearchResult.find(:all, :conditions => conditions).each do |result|
      # just use rank for rank-biserial correlation
      score = result.new_rank
```

```
      attrs.each{ |n| data_sets[n] << [result.send(n.to_s), score]}
    end
  rescue => e
    print e
  end


  print "\n\nQuery: #{query}\n"
  if !ruport
    print "Attribute, Correlation, 0/Low, 1/High\n"
  end
  ruport_data = []

  attrs.each do |n|
    #puts "Generating correlation coefficient for (x,y) => (#{n.to_s},
score)"
    next if !n
    if n == :page_rank || n==:keyword_freq || n==:title_pos || n==:url_pos
      puts Correlation.raw_score(data_sets[n])
    else
      results = Correlation.rank_biserial(data_sets[n])
      correlation = sprintf("%1.4f", results[0]) || 'N/A'
      n0 = results[1]
      n1 = results[2]
      if ruport
        ruport_data << [n.to_s, correlation.to_s, n0.to_s, n1.to_s] # convert
to_s or ruport will crash
      else
        print "#{n}, #{correlation}, #{n0}, #{n1}\n"
      end
    end
  end

  # print ruport table
  if ruport
    ruport_table = Ruport::Data::Table.new(:data => ruport_data,
       :column_names => ["Attribute", "Correlation", "0/Low", "1/High"])
    print ruport_table.sort_rows_by {|r|
      next -5.0 if r["Correlation"].eql?("NaN")
       (1.0 - r["Correlation"].to_f)
    }.to_s
  end

end
```

Appendix G
do_correlation_by_attribute.rb

```
##########################################
# Correlational analysis with results separated
# by attribute.  Also, calculates mean
# rank-biserial correlation for each attribute.
# External Ruport reporting library
# is used, if it is found on the system.
# NOTE: to use Ruport, the following line had to be commented out
# from within the ruport source code:
#   r.gsub!(/\A.{#{width+1},}/) { |m| m[0,width-2] + ">>" }
# ( ruport/format/text.rb:73 )
##########################################

require File.dirname(__FILE__) + '/correlation'
require File.dirname(__FILE__) + '/../search_result'

ruport = true
begin
  require 'ruport'
rescue => er
  p er
  ruport=false
end

queries = [:adhibit, :agrestic, :appetency, :kouprey, :mystagogue,
:nouthetic, :numbat, :pacarana, :paradoxology, :phascogale]
attrs = [:page_rank_d, :keyword_freq_d, :in_title, :in_url, :in_b, :in_h1,
:in_h2, :in_h3, :in_strong,
:in_i, :in_u, :in_select, :in_img_src, :in_img_alt, :in_link_href,
:in_link_text, :in_input]
mean_corrs = {}

attrs.each do |attr|
  #attr = (ARGV.shift || 'in_url').to_sym

  columns = ["Query", "Correlation", "Low/0", "High/1"]
  print "\n\nAttribute:  #{attr}\n"
  if !ruport
    print "Query, Correlation, Low/0, High/1\n"
  end
```

```
  ruport_data = []
  corr_sum = 0 # sum of all correlations (excluding NaN)
  corr_cnt = 0 # count of all correlations (excluding NaN)

  queries.each do |query|
    begin
      data_set = []
      conditions = "query LIKE '#{query}' AND new_rank>0"
      sql = "SELECT #{attr},new_rank FROM search_results WHERE #{conditions}"
      #SearchResult.find(:all, :conditions => conditions).each do |result|
      SearchResult.find_by_sql(sql).each do |result|  # more efficient
        score = result.new_rank
        data_set << [result.send(attr.to_s), score]
      end

      # calculate correlation
      if attr == :page_rank || attr==:keyword_freq || attr==:title_pos ||
attr==:url_pos
        puts Correlation.raw_score(data_set)
      else
        results = Correlation.rank_biserial(data_set)
        if !results[0].nan?
          corr_sum += results[0]
          corr_cnt += 1
        end
        correlation = sprintf("%1.4f", results[0]) || 'N/A'
        n0 = results[1]
        n1 = results[2]
        if ruport
          ruport_data << [query, correlation.to_s, n0.to_s, n1.to_s] #
convert to_s or ruport will crash
        else
          print "#{query}, #{correlation}, #{n0}, #{n1}\n"
        end
      end

    rescue => e
      print e
    end
  end

  # print table
  if ruport
    avg_corr = sprintf("%1.4f", (corr_sum.to_f/corr_cnt.to_f))
    mean_corrs[attr] = avg_corr
```

```
    ruport_data << ['', '', '', '']
    ruport_data << ['Avg. Correlation:', avg_corr, '', '']
    ruport_table = Ruport::Data::Table.new(:data => ruport_data,
                                           :column_names => columns)
    print ruport_table.to_s

  end

end

# print table of mean correlations by attr
ruport_data = []
mean_corrs.each do |attr,corr|
  ruport_data << [attr, corr]
end
ruport_table = Ruport::Data::Table.new(:data => ruport_data,
                                       :column_names => ["Attribute", "Mean
Correlation"])
print "\n\n"

print ruport_table.sort_rows_by {|r|
  next -5.0 if r["Mean Correlation"].eql?("NaN")
   (1.0 - r["Mean Correlation"].to_f)
}.to_s
```

Appendix H
correlation.rb

```
##########################################
# Actual methods used to calculate the correlation.
# Rank-biserial and standard pearson r (raw score formula)
# are included.
##########################################



class Correlation

  # Rank-Biserial Correlation Coefficient is used
  # for correlating an ordinal variable
  # and a dichotomous variable, which is true for most of the data.
  def self.rank_biserial(data)
    # r = 2*(Y1-Y0)/n
    #   n = number of data pairs
    #   Y0 = mean rank for values in which x = 0
    #   Y1 = mean rank for values in which x = 1
    n = data.length
    y0_sum = 0
    n0 = 0
    y1_sum = 0
    n1 = 0
    data.each do |data_pair|
      if (data_pair[0]==0)
        y0_sum += data_pair[1]
        n0 += 1
      elsif (data_pair[0]==1)
        y1_sum += data_pair[1]
        n1 += 1
      end
    end

    y0_mean = y0_sum.to_f/n0.to_f
    y1_mean = y1_sum.to_f/n1.to_f
    r = (2.0/n.to_f) * (y0_mean.to_f - y1_mean.to_f)
    return r, n0, n1
  end
```

```
  # Raw Score formula to calculate the Correlation Coefficient (Pearson r)
  # data must be a 2 dimensional array of data [ [x,y],[x,y],[x,y], ... ]
  def self.raw_score(data)
    sum_cross = sum_x = sum_y = sum_x_sqr = sum_y_sqr = 0
    data.each do |data_pair|
      # calculate sum of cross products
      sum_cross += (data_pair[0] * data_pair[1])
      # calculate sum of x
      sum_x += data_pair[0]
      # calculate sum of y
      sum_y += data_pair[1]
      # calculate sum of x squared
      sum_x_sqr += data_pair[0]*data_pair[0]
      # calculate sum of y squared
      sum_y_sqr += data_pair[1]*data_pair[1]
    end
    n = data.length
    denominator = (n*sum_cross) - (sum_x * sum_y)
    divisor = Math.sqrt( (n*sum_x_sqr - sum_x*sum_x)*(n*sum_y_sqr -
sum_y*sum_y) )
    denominator.to_f/divisor.to_f
  end


end
```

Appendix I: Results by Query

Attribute	Correlation	0/Low	1/High
in_select	0.9194	546	3
in_title	0.8100	492	57
in_h1	0.7388	526	23
in_url	0.7149	513	36
in_h3	0.6686	535	14
in_h2	0.6603	535	14
in_strong	0.5947	535	14
in_input	0.5043	528	21
in_link_text	0.4814	471	78
keyword_freq_d	0.4690	467	82
in_i	0.4583	531	18
in_img_alt	0.3949	535	14
in_link_href	0.3871	490	59
in_b	0.3132	501	48
in_img_src	0.0622	547	2
in_u	0.0596	545	4

Table I-1. Results for Query 'agrestic'

Attribute	Correlation	0/Low	1/High
in_select	0.9886	525	2
in_h3	0.9696	526	1
in_strong	0.8810	521	6
in_img_src	0.8667	522	5
in_input	0.8154	512	15
in_url	0.8053	494	33
in_img_alt	0.7763	523	4
in_title	0.7160	486	41
in_h1	0.6891	511	16
keyword_freq_d	0.5896	464	63
in_link_href	0.5406	486	41
in_b	0.4542	490	37
in_link_text	0.4272	476	51
in_i	0.2153	516	11
in_h2	-0.1730	523	4
in_u	-0.9886	526	1

Table I-2. Results for Query 'appetency'

Attribute	Correlation	0/Low	1/High
in_title	0.9009	472	61
in_url	0.8516	489	44
in_h2	0.7967	518	15
in_h1	0.7755	503	30
keyword_freq_d	0.7540	399	134
in_select	0.7283	530	3
in_img_src	0.6475	507	26
in_input	0.6204	519	14
in_img_alt	0.6198	511	22
in_strong	0.6165	522	11
in_u	0.5951	531	2
in_h3	0.4991	512	21
in_link_href	0.4796	432	101
in_link_text	0.4317	414	119
in_b	0.4311	463	70
in_i	0.2904	508	25

Table I-3. Results for Query 'kouprey'

Attribute	Correlation	0/Low	1/High
in_title	0.8615	517	78
in_select	0.8469	591	4
in_url	0.8096	556	39
in_h1	0.7911	573	22
in_input	0.7720	568	27
keyword_freq_d	0.7694	512	83
in_img_src	0.7487	580	15
in_link_href	0.6740	516	79
in_u	0.5826	588	7
in_b	0.4598	517	78
in_link_text	0.4234	496	99
in_h2	0.3979	579	16
in_strong	0.3668	573	22
in_img_alt	0.3456	559	36
in_h3	0.3269	586	9
in_i	0.2439	566	29

Table I-4. Results for Query 'mystagogue'

Attribute	Correlation	0/Low	1/High
in_select	0.9567	647	1
in_url	0.8771	626	22
in_img_src	0.8541	642	6
in_title	0.8008	598	50
in_input	0.7975	632	16
in_h1	0.7316	629	19
keyword_freq_d	0.6937	546	102
in_h3	0.6309	637	11
in_h2	0.6027	635	13
in_link_href	0.3749	581	67
in_i	0.3443	621	27
in_img_alt	0.3181	618	30
in_strong	0.2686	624	24
in_b	0.2486	565	83
in_link_text	0.1254	516	132
in_u	0.0854	639	9

Table I-5. Results for Query 'nouthetic'

Attribute	Correlation	0/Low	1/High
in_title	0.6606	461	181
in_url	0.5482	538	104
in_select	0.5037	635	7
in_h2	0.4565	605	37
in_h1	0.4497	558	84
keyword_freq_d	0.3976	328	314
in_h3	0.3176	615	27
in_input	0.2896	609	33
in_b	0.2486	503	139
in_link_href	0.2287	452	190
in_img_src	0.2137	532	110
in_strong	0.2134	593	49
in_img_alt	0.0951	538	104
in_link_text	0.0913	393	249
in_i	0.0648	604	38
in_u	-0.0019	633	9

Table I-6. Results for Query 'numbat'

Attribute	Correlation	0/Low	1/High
in_input	0.9267	382	2
in_title	0.8661	362	22
in_url	0.8221	372	12
in_h1	0.7571	364	20
in_h2	0.7185	375	9
keyword_freq_d	0.6815	330	54
in_img_alt	0.6000	380	4
in_img_src	0.5502	375	9
in_i	0.5430	377	7
in_b	0.4808	352	32
in_h3	0.4365	373	11
in_strong	0.2293	373	11
in_u	0.1008	379	5
in_link_href	-0.2235	307	77
in_link_text	-0.3525	288	96
in_select	-0.4803	381	3

Table I-7. Results for Query 'pacarana'

Attribute	Correlation	0/Low	1/High
in_url	0.8964	358	11
in_title	0.8751	346	23
in_h1	0.7070	359	10
keyword_freq_d	0.6886	337	32
in_h2	0.6764	362	7
in_input	0.6575	362	7
in_b	0.5227	353	16
in_h3	0.4834	361	8
in_strong	0.4674	359	10
in_img_src	0.4440	359	10
in_i	0.4141	361	8
in_select	0.1362	367	2
in_link_href	-0.1886	230	139
in_img_alt	-0.2935	368	1
in_link_text	-0.5140	124	245
in_u	-0.6867	366	3

Table I-8. Results for Query 'paradoxology'

Attribute	Correlation	0/Low	1/High
in_select	0.9362	470	2
in_title	0.6755	419	53
keyword_freq_d	0.6364	355	117
in_h1	0.6230	439	33
in_h3	0.5937	466	6
in_img_alt	0.5172	434	38
in_h2	0.5069	457	15
in_img_src	0.4864	448	24
in_input	0.4693	457	15
in_url	0.4219	453	19
in_link_href	0.4120	386	86
in_strong	0.3542	444	28
in_link_text	0.3445	362	110
in_i	0.3101	366	106
in_b	0.3058	411	61
in_u	0.0133	467	5

Table I-9. Results for Query 'phascogale'

(see query 'adhibit' in Table 6-1-1)

Appendix J
Results by Attribute

(see attribute 'keyword_freq_d' in Table 6-2-1)

Query	Correlation	Low/0	High/1
adhibit	0.9517	318	19
agrestic	0.8100	492	57
appetency	0.7160	486	41
kouprey	0.9009	472	61
mystagogue	0.8615	517	78
nouthetic	0.8008	598	50
numbat	0.6606	461	181
pacarana	0.8661	362	22
paradoxology	0.8751	346	23
phascogale	0.6755	419	53
Avg. Correlation:	0.8118		

Table J-1. Results for Attribute 'in_title'

Query	Correlation	Low/0	High/1
adhibit	0.8178	321	16
agrestic	0.7149	513	36
appetency	0.8053	494	33
kouprey	0.8516	489	44
mystagogue	0.8096	556	39
nouthetic	0.8771	626	22
numbat	0.5482	538	104
pacarana	0.8221	372	12
paradoxology	0.8964	358	11
phascogale	0.4219	453	19
Avg. Correlation:	0.7565		

Table J-2. Results for Attribute 'in_url'

Query	Correlation	Low/0	High/1
adhibit	0.4915	313	24
agrestic	0.3132	501	48
appetency	0.4542	490	37
kouprey	0.4311	463	70
mystagogue	0.4598	517	78
nouthetic	0.2486	565	83
numbat	0.2486	503	139
pacarana	0.4808	352	32
paradoxology	0.5227	353	16
phascogale	0.3058	411	61
Avg. Correlation:	0.3956		

Table J-3. Results for Attribute 'in_b'

Query	Correlation	Low/0	High/1
adhibit	0.8720	322	15
agrestic	0.7388	526	23
appetency	0.6891	511	16
kouprey	0.7755	503	30
mystagogue	0.7911	573	22
nouthetic	0.7316	629	19
numbat	0.4497	558	84
pacarana	0.7571	364	20
paradoxology	0.7070	359	10
phascogale	0.6230	439	33
Avg. Correlation:	0.7135		

Table J-4. Results for Attribute 'in_h1'

Query	Correlation	Low/0	High/1
adhibit	0.8664	333	4
agrestic	0.6603	535	14
appetency	-0.1730	523	4
kouprey	0.7967	518	15
mystagogue	0.3979	579	16
nouthetic	0.6027	635	13
numbat	0.4565	605	37
pacarana	0.7185	375	9
paradoxology	0.6764	362	7
phascogale	0.5069	457	15
Avg. Correlation:	0.5509		

Table J-5. Results for Attribute 'in_h2'

Query	Correlation	Low/0	High/1
adhibit	0.1049	329	8
agrestic	0.6686	535	14
appetency	0.9696	526	1
kouprey	0.4991	512	21
mystagogue	0.3269	586	9
nouthetic	0.6309	637	11
numbat	0.3176	615	27
pacarana	0.4365	373	11
paradoxology	0.4834	361	8
phascogale	0.5937	466	6
Avg. Correlation:	0.5031		

Table J-6. Results for Attribute 'in_h3'

Query	Correlation	Low/0	High/1
adhibit	0.6707	328	9
agrestic	0.5947	535	14
appetency	0.8810	521	6
kouprey	0.6165	522	11
mystagogue	0.3668	573	22
nouthetic	0.2686	624	24
numbat	0.2134	593	49
pacarana	0.2293	373	11
paradoxology	0.4674	359	10
phascogale	0.3542	444	28
Avg. Correlation:	0.4663		

Table J-7. Results for Attribute 'in_strong'

Query	Correlation	Low/0	High/1
adhibit	0.5818	330	7
agrestic	0.4583	531	18
appetency	0.2153	516	11
kouprey	0.2904	508	25
mystagogue	0.2439	566	29
nouthetic	0.3443	621	27
numbat	0.0648	604	38
pacarana	0.5430	377	7
paradoxology	0.4141	361	8
phascogale	0.3101	366	106
Avg. Correlation:	0.3466		

Table J-8. Results for Attribute 'in_i'

Query	Correlation	Low/0	High/1
adhibit	-0.7284	335	2
agrestic	0.0596	545	4
appetency	-0.9886	526	1
kouprey	0.5951	531	2
mystagogue	0.5826	588	7
nouthetic	0.0854	639	9
numbat	-0.0019	633	9
pacarana	0.1008	379	5
paradoxology	-0.6867	366	3
phascogale	0.0133	467	5
Avg. Correlation:	-0.0969		

Table J-9. Results for Attribute 'in_u'

Query	Correlation	Low/0	High/1
adhibit	0.9583	336	1
agrestic	0.9194	546	3
appetency	0.9886	525	2
kouprey	0.7283	530	3
mystagogue	0.8469	591	4
nouthetic	0.9567	647	1
numbat	0.5037	635	7
pacarana	-0.4803	381	3
paradoxology	0.1362	367	2
phascogale	0.9362	470	2
Avg. Correlation:	0.6494		

Table J-10. Results for Attribute 'in_select'

Query	Correlation	Low/0	High/1
adhibit	-0.0747	332	5
agrestic	0.0622	547	2
appetency	0.8667	522	5
kouprey	0.6475	507	26
mystagogue	0.7487	580	15
nouthetic	0.8541	642	6
numbat	0.2137	532	110
pacarana	0.5502	375	9
paradoxology	0.4440	359	10
phascogale	0.4864	448	24
Avg. Correlation:	0.4799		

Table J-11. Results for Attribute 'in_img_src'

Query	Correlation	Low/0	High/1
adhibit	0.3411	330	7
agrestic	0.3949	535	14
appetency	0.7763	523	4
kouprey	0.6198	511	22
mystagogue	0.3456	559	36
nouthetic	0.3181	618	30
numbat	0.0951	538	104
pacarana	0.6000	380	4
paradoxology	-0.2935	368	1
phascogale	0.5172	434	38
Avg. Correlation:	0.3715		

Table J-12. Results for Attribute 'in_img_alt'

Query	Correlation	Low/0	High/1
adhibit	-0.0829	273	64
agrestic	0.3871	490	59
appetency	0.5406	486	41
kouprey	0.4796	432	101
mystagogue	0.6740	516	79
nouthetic	0.3749	581	67
numbat	0.2287	452	190
pacarana	-0.2235	307	77
paradoxology	-0.1886	230	139
phascogale	0.4120	386	86
Avg. Correlation:	0.2602		

Table J-13. Results for Attribute 'in_link_href'

Query	Correlation	Low/0	High/1
adhibit	-0.1957	250	87
agrestic	0.4814	471	78
appetency	0.4272	476	51
kouprey	0.4317	414	119
mystagogue	0.4234	496	99
nouthetic	0.1254	516	132
numbat	0.0913	393	249
pacarana	-0.3525	288	96
paradoxology	-0.5140	124	245
phascogale	0.3445	362	110
Avg. Correlation:	0.1263		

Table J-14. Results for Attribute 'in_link_text'

Query	Correlation	Low/0	High/1
adhibit	0.9325	330	7
agrestic	0.5043	528	21
appetency	0.8154	512	15
kouprey	0.6204	519	14
mystagogue	0.7720	568	27
nouthetic	0.7975	632	16
numbat	0.2896	609	33
pacarana	0.9267	382	2
paradoxology	0.6575	362	7
phascogale	0.4693	457	15
Avg. Correlation:	0.6785		

Table J-15. Results for Attribute 'in_input'