# How to Improve Your Google Ranking: Myths and Reality

Ao-Jan Su[†], Y. Charlie Hu[‡], Aleksandar Kuzmanovic[†], and Cheng-Kok Koh[‡]
[†]Northwestern University, Evanston, IL, USA
Email:{ajsu,akuzma}@cs.northwestern.edu
[‡]Purdue University, West Lafayette, IN, USA
Email:{ychu,chengkok@purdue.edu}

***Abstract*—Search engines have greatly influenced the way people access information on the Internet as such engines provide the preferred entry point to billions of pages on the Web. Therefore, highly ranked web pages generally have higher visibility to people and pushing the ranking higher has become the top priority for webmasters. As a matter of fact, search engine optimization (SEO) has became a sizeable business that attempts to improve their clients' ranking. Still, the natural reluctance of search engine companies to reveal their internal mechanisms and the lack of ways to validate SEO's methods have created numerous myths and fallacies associated with ranking algorithms; Google's in particular.**

**In this paper, we focus on the Google ranking algorithm and design, implement, and evaluate a ranking system to systematically validate assumptions others have made about this popular ranking algorithm. We demonstrate that linear learning models, coupled with a recursive partitioning ranking scheme, are capable of reverse engineering Google's ranking algorithm with high accuracy. As an example, we manage to correctly predict 7 out of the top 10 pages for 78% of evaluated keywords. Moreover, for content-only ranking, our system can correctly predict 9 or more pages out of the top 10 ones for 77% of search terms. We show how our ranking system can be used to reveal the relative importance of ranking features in Google's ranking function, provide guidelines for SEOs and webmasters to optimize their web pages, validate or disapprove new ranking features, and evaluate search engine ranking results for possible ranking bias.**

## I. Introduction

Search engines have become generic knowledge retrieval platforms used by millions of Internet users on a daily bases. As such, they have become important vehicles that drive users towards web pages highly ranked at them (*e.g.*, [21], [24]). Consequently, finding ways to improve ranking at popular search engines is an important goal of all web sites that care about attracting clients.

Ways to improve a web page's search engine ranking are different. On one side, SPAM farms are a well-known approach to boost a web site's ranking. This is achieved by artificially inflating a site's popularity, *i.e.*, by increasing the number of fake links pointing to it. Luckily, ways to detect and contain such approaches appear to be quite successful [20], [22], [26]. On the other side, an entire new industry, *i.e.*, search engine optimization (SEO) (*e.g.*, [15], [19]), is booming. Such companies and experts claim to be capable of improving a web page's rank by understanding which page design choices and factors are valued by the ranking algorithms.

Unfortunately, the combination of the natural reluctance of search engine companies to reveal any detail about their internal mechanisms (which are kept top secret), the lack of any knowledge or independent validation of the SEO methodologies, and the ever-lasting interest on this topic, has opened the doors to various theories and claims, myths and folklore about which particular factor is influential *e.g.*, [1]–[3], [6], [14], [17], [18]. To the best of our knowledge, none of these claims are backed by any scientific evidence. At the same time, the problem of reverse-engineering a search engine's ranking algorithm, such as Google's, is widely considered a close-to-impossible task in the academic community due to its inherent complexity.

The key contribution of our paper is that we demonstrate that simple *linear* learning models, accompanied by a recursive partitioning ranking scheme, are capable of predicting a search engine's (Google's, in particular) ranking results with high accuracy. As an example, we show that when non-page-content factors are isolated, our ranking system manages to correctly predict 8 pages within the top 10 ones for 92% of explored keywords. In the more general scenarios, we manage to correctly predict 7 or more pages within the top 10 ones for 78% of explored keywords.

In this paper, we develope an automated ranking system that directly queries Google, collects search results, and feeds the results to its ranking engine for learning. Our ranking engine incorporates several learning algorithms, based on training both linear and polynomial models. Using our ranking system, we show that a linear model trained with linear programming and accompanied with recursive partitioning algorithm is able to closely approximate Google's ranking algorithm. In addition, we use our ranking system to analyze the importance of different ranking features to provide guidelines for SEOs and webmasters to improve Google's ranking of web pages. Finally, we present case studies on how our ranking system can facilitate in validating and disapproving potential new ranking features reported in the Internet. More specifically, we confirm that Google imposes negative bias toward blogs, and that HTML syntax errors have little to no impact to Google's ranking.

This paper is structured as follows. In Section II, we define the problem and outline the folklore. In Section III, we present the detailed design of our ranking system. We present our evaluation results of our ranking system in Section IV and several case studies in Section V. Finally, we conclude in Section VI.

## II. Problem Statement

### A. Goals

There have been numerous efforts that attempt to reveal the importance of ranking factors to a search engine [1], [3], [14], [18]. While some of them are guess-works by webmasters [1], [18], others are based on experience of search engine optimization (SEO) experts [3], [14]. While we recognize that guessing and experience might indeed be vehicles for revealing search engine internals, we strive for more systematic and scientific avenues to achieve this task. The goal of our study is to understand the important factors that affect the ranking of a web page as viewed by popular search engines. In doing so, we validate some folklore and popular beliefs advertised by webmasters and the SEO industry.

*1) Ranking Features:* In this work, we focus on Google's search engine, and aim to study the relative importance of web page features that potentially affect the ranking of a web page, as listed in Table I. For each web page (URL), we collect 17 ranking features. These ranking features can further be divided into 7 groups. The page group represents characteristics associated with the web page including page rank score (PR) and the age of the web page (AGE). The URL group represent features associated with the URL of the web page. Parameter HOST counts the number of occurrences of the keyword that appear in the hostname and PATH counts the number of occurrences of the keyword in the page segment of the URL.

The domain group consists of features related to the domain of a web site. D_SIZE reports the number of web pages indexed by Google in the domain and D_AGE reports the age of the first page index by archive.org in the domain. Groups header, body, heading and link are features extracted from the content of the web page. TITLE counts the number of occurrences of the keyword in the title tag. M_KEY counts the number of occurrences of the keyword in the meta keyword tag and M_DES counts the number of occurrences of the keyword in the meta description tag. DENS is the keyword density of a web page which is calculated as the number of occurrences of the keyword divided by the number of words in the web page. H1 through H5 is the number of occurrences of the keyword in all the headings H1 to H5, respectively. ANCH counts the number of occurrences of the keyword in the anchor text of an outgoing link and IMG counts the number of occurrences of the keyword in an image tag.

Google claims to use more than 200 parameters in its ranking system. Necessarily, we explore only a subset of all possible features. Still, we demonstrate that ranking features listed in Table I are adequate for providing high ranking prediction accuracy. Moreover, we are capable of establishing important relationships among the explored features.

### B. State-of-the-Affairs (the Folklore)

The natural Google's reluctance to reveal any details about their internal mechanisms on one hand, and the great popularity and the impact it has in shaping users' browsing behavior on the other, has created a great interest and attempts to understand how its ranking algorithm works. Still, there is no consensus on the set of the most important features. Different people express quite different opinions, as we illustrate below.

We collect different opinions for Google's ranking features and summarize in Table II. The second column labeled by

| Group | Feature | Detail |
|---|---|---|
| Page | PR | pagerank score |
| | AGE | age of the web page |
| URL | HOST | keyword appear in hostname |
| | PATH | keyword in the path segment of url |
| Domain | D_SIZE | size of the web site's domain |
| | D_AGE | age of the web site's domain |
| Header | TITLE | keyword in the title tag of html header |
| | M_KEY | keyword in meta-keyword tag |
| | M_DES | keyword in meta-description tag |
| Body | DENS | keyword density |
| Heading | H1 | keyword in h1 tag |
| | H2 | keyword in h2 tag |
| | H3 | keyword in h3 tag |
| | H4 | keyword in h4 tag |
| | H5 | keyword in h5 tag |
| Link | ANCH | keyword in anchor text |
| | IMG | keyword in image tag |

TABLE I
Ranking Features

| # | SEOmoz'07 | Survey | Idv |
|---|---|---|---|
| 1 | Keyword use in title tag | Keywords in title | Keyword in URL |
| 2 | Anchor text of inbound link | Keywords in domain name | Keyword in domain name |
| 3 | Global link popularity of site | Anchor text of inbound links | Keyword in title tag |
| 4 | Age of site | Keywords in heading tags | Keyword in H1, H2 and H3 |
| 5 | Link popularity within the sites internal link structure | Keywords in URL | Page Rank |
| 6 | Topical relevance of inbound links to site | Anchor text from within the site | Anchor text of inbound link to you |
| 7 | Link popularity of site in topical community | Internal links | Site listed in DMOZ Directory |
| 8 | Keyword use in body text | Keywords in Alt attribute of images | Site listed in Yahoo Directory |
| 9 | Global link popularity of linking site | Relevance of external links | Rank Manipulation by Competitor Attack |
| 10 | Topical relationship of linking page | Keywords in body | Site Age |

TABLE II
Various Ranking Feature Opinions

*SEOmoz'07* [6] is a list of top 10 ranking factors created by surveying 37 SEO experts by SEOmoz [14] in 2007. This column represents observations from knowledgeable experts. The third column labeled by *Survey* [18] is a list of top 10 ranking features rated by a poll of Internet users interested in this topic. This column represents the perception of Google ranking algorithm from general Internet users. The fourth column labeled as *Idv* [17] is the top 10 ranking feature list posted by an Internet marketing expert on his personal web page. This column represents an individual investigator that studies this topic.

Due to the lack of systematical measuring and evaluating guidelines, it is not surprising to see a huge difference of ranking between the three lists. The SEO experts obviously favor ranking features associated with hyperlinks as they rated 7 out of the top 10 ranking features in this category. On the contrary, the other two opinions have only 3 and 1 top 10 ranking features associated with links, respectively. In addition, the web-site's age feature is in the top 4 features among SEOs, but is absent in the second list and is at the bottom of the third list. Moreover, some of the ranking features are counter-intuitive. For example, the 7th ranking feature in the *Idv* list suggests that a site listed in the Yahoo directory

can help Google's ranking. Furthermore, we are surprised to see only one list(*Idv*) includes the page rank in the top 10 lists, which is still widely believed as an important Google's ranking factor.[1]

From time to time, Internet users will see rumors that spread about Google's ranking algorithm. However, to the best of our knowledge, there does not exist a systematic approach to validate or disapprove these assumptions. This motivates us to perform research in this topic and build a system to facilitate the necessary evaluation process.

## III. Methodology

In this section, we discuss the design of our ranking system that analyzes and reverse-engineers Google's ranking algorithm. The architecture of our system is depicted in Figure 1. The two major components are the crawler and the ranking engine. The data collection is performed by the *crawler* which queries Google and receives the ranked search results. In addition, it downloads HTML web pages from their original web sites and queries domain information as described in Section III-A.

Second, since multiple features can affect the ranking of web pages in complicated ways, the ranking engine extracts features under study from raw web pages and performs learning to train several ranking models to approximate the ranking results by Google. In this part, we make several contributions: (1) We confirm that Google's ranking function is not a simple linear function of all the features, by showing a nonlinear model can outperform, *i.e.,* approximates Google's ranking better than, a simple linear model. However, a nonlinear model is difficult for humans to digest. (2) We present a simple recursive ranking procedure based on a simple linear model and show that it can achieve comparable accuracies to the nonlinear model. The theoretical underpinning for such a procedure is that recursive application of a linear model (function) can effectively approximate a non-linear function. In addition, the linear model converges more efficiently and outputs more human readable results.

### A. The Crawler

The crawler submits queries to Google search engine and obtains top 100 web pages (URL) for each keyword. Without losing generality, we limited our queries to HTML files to avoid web pages generated dynamically by server side scripts such as CGI or PHP. In addition, we focus on web pages composed in English in our experiments. The Google API syntax we use for the above two features is as_filetype: html&lr=lang_en. Moreover, to obtain the date Google indexed the web pages, we submit our queries with an additional parameter qdr:y10. By doing so, the date Google indexed the web page will be returned in the search result page for us to extract the age of the page ranking feature. Finally, for each web page, the crawler does the following:

1) Downloads the web page from the original web site.
2) Queries the URL's page rank score by Google toolbar's API [11].
3) Obtains the age of a page (the date Google indexed the web page) by parsing the search result page.

[1]It is possible that the importance of the Page Rank feature was implicitly assumed. Nevertheless, we were unable to find any such explicit statement associated with the given lists.

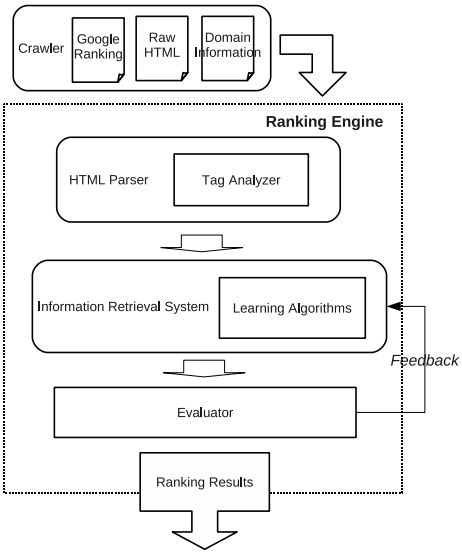



Fig. 1. System Architecture

4) Obtains the size (the total number of pages) of the domain by querying Google with site:[domain].
5) Queries archive.org and fetches the age of the web site (the date when the first web page was created on this web site).

### B. The Ranking Engine

There are three components in the ranking engine. The HTML parser [12] converts web pages into the document object model (DOM) for the tag analyzer to exam the number of keywords that appear in different HTML tags such as anchor text. The ranking engine trains the ranking model by combining features obtained from the web page contents, page rank scores, and domain information. After the model is created, the ranking engine evaluates the testing sets by applying the model. The evaluator then analyzes the results and provides feedback to the ranking engine which is used to adjust parameters in the learning algorithms such as error threshold. In the following sections, we describe the two ranking models we experimented in this paper – Linear programming and SVM. We use ranking features listed in Table I to train our ranking models.

*1) Linear Programming Ranking Model:* In this Section, we describe our linear programming ranking model. Given a set of documents $I = (i_1, i_2, ..., i_n)$, pre-defined Google ranking $G = (1, 2, ..., n)$, and a ranking algorithm $A$, the goal is to find a set of weights $W = (w_1, w_2, ..., w_m)$ that makes the ranking algorithm re-produce Google ranking with minimum errors. The objective function of the linear programming algorithm attempts to minimize errors (the sum of penalties) of the ranking of a document set. Equation (1) defines the objective function which is a pairwise comparison between two documents in a given data set.

$$\Phi(W) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_i \cdot |i - j| \cdot D(i, j) \qquad (1)$$

In Equation (1), $c_i$ is a factor that weights the importance of the $i_th$ document (*e.g.,* a top 5th page is more important than a top 50th page). $|i - j|$ is the distance (ranking difference)

between the $i_{th}$ and the $j_{th}$ page. Finally, $D(i,j)$ is a decision function we define as

$$D(i,j) = \begin{cases} 0 & \text{if } f(A,W,i) >= f(A,W,j), \\ 1 & \text{if } f(A,W,i) < f(A,W,j). \end{cases} \quad (2)$$

where $f(A,W,i)$ is the score produced by algorithm $A$ with a set of weights $W$ for the $i_{th}$ page in the given data set. Page X is ranked higher than page Y if it receives a higher score than page Y. The decision function denotes that if the ranking of the two pages preserves the order as Google's ranking, the penalty is zero. Otherwise, the penalty will be counted in the error function which is denoted in Equation (1).

Since we cannot import conditional functions (*e.g.,* $D(i,j)$) into a linear programming solver, we transform the decision function into the following form:

$$f(A,W,i) + D_{ij}F_{\max} \geq f(A,W,j), \quad (3)$$

where $F_{\max}$ is the maximum value to which $f(A,W,\cdot)$ would evaluate, and $D_{ij} \in \{0,1\}$. When $D_{ij} = 0$, the preceding inequality is satisfied only if

$$f(A,W,i) \geq f(A,W,j). \quad (4)$$

When $D_{ij} = 1$, the inequality is always satisifed. Therefore, we have effectively converted the original minimization problem into the problem of minimizing the $D_{ij}$. Hence, we can now replace $D(i,j)$ in the objective function with $D_{ij}$. Finally, the score function $f(A,W,i)$ can be represented by a dot product of ranking parameters $X = (x_1, x_2, ..., x_m)$ and weights $W = (w_1, w_2, ..., w_m)$ denoted as $f(A,W,i) = f_A(w_i \cdot x_i)$. For example, the parameter $x_i$ can be the number of keywords that occur in the `title` tag and $w_i$ is the weight associated with $x_i$.

In addition to the objective function, we set constrains to our linear programming model. For each pair of pages (i, j) where i < j (i is ranked higher than j by Google), we have a constrain:

$$f(A,W,j) - f(A,W,i) \leq \tau \quad (5)$$

where $\tau$ is the maximum allowed error which is set to a predefined constant $\tau$. The constant $\tau$ is adjusted by the feedback from the evaluator to refine the ranking results. For example, when linear programing solver cannot find a feasible solution, we relax the maximum allowed error $\tau$. The linear programming solver we use in our experiments is `ILOG CPLEX` [9]. In addition, we apply a recursive partitioning algorithm as we describe in Section III-B3 below.

*2) Support Vector Machines Ranking Model:* Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and learning ranking functions [25]. In a SVM, data points are viewed as `n-dimensional` vectors (n equals to the number of ranking features in our case). A SVM constructs a hyperplane or a set of hyperplanes in a high-dimensional space, which is used as a classifier to separate data points. In our experiments, we use the `SVM-rank` [16], [23] implementation with linear and polynomial kernels to train the ranking functions. The ranking features we used in our SVM experiments are the same as the linear programming model as discussed in Section II. The parameter `c` in `SVM-rank` controls the trade-off between training error and margin. The ranking engine adjusts the value of parameter `c` according to the feedback provided by the evaluator in order to find the best value for prediction accuracy. Finally, we perform a recursive partitioning algorithm as we describe in the following section.

*3) Recursive Partitioning Ranking Algorithm:* It is common that a search engine keeps several layers of indices in practice. For example, the first layer of indices may serve as a cache and it is able to answer queries for `top 20` pages. When the first layer query fails, it is then sent to subsequent indices. Additionally, the search engine's internal ranking algorithm can be non-linear. To capture such a non-linear and/or non-equational behavior for search engine's ranking function, we developed a recursive partitioning algorithm to approximate this ranking behavior. We apply this algorithm to both our linear and SVM models. We evaluate the power of this recursive partitioning ranking algorithm in Section IV-C. First, we describe our recursive partitioning algorithm with pseudo-code shown below.

**Algorithm 1** Recursive Partitioning Ranking Algorithm

```
1: procedure PARTITION(S, X) S: a set of pages, X: top X
2:     Rank(S) Train or apply ranking models
3:     while |S| > 2.5 * X do
4:         N = Max(2 * X, |S|/2)
5:         S ← Top(S, N) Return top N pages
6:         return Partition(S, X)
7:     end while
8:     return Top(S, X) Return top X pages
9: end procedure
```

In algorithm 1, $S$ denotes a set of pages in a dataset and $X$ denotes the target top X pages to be evaluated (*e.g.,* top 10 pages). Algorithm 1 can be explained by giving an example of how to train recursive ranking models for selecting top 10 pages ($X = 10$) out of 100 web pages ($|S| = 100$). In the first round of recursion, the learning algorithm produces a set of weights by training all 100 pages (in line #2 of algorithm 1). Next, line #4 calculates $N = 50$. In line #5, the function returns the top 50 pages out of 100 using the model learned previously. Next, the algorithm moves on to the second recursion with a set of 50 pages in $S$. Similarly, in the second recursion, a new set of weights for ranking is learned and the variable $N$ in line #4 becomes 25. The partitioning algorithm further extracts top 25 pages from the 50 pages (in line #5) and proceeds to the third round. In the third round, an additional new set of weights is learned in line #2 of the algorithm 1. The condition statement in line #3 is not met in this round. Therefore, the algorithm escapes the recursive while loop. Algorithm 1 then proceeds to line #8 and return the top 10 pages by applying the ranking model learned in the third round.

The process of evaluating the testing sets using our recursive partitioning ranking algorithm is similar to the steps we described above. The input of this evaluation process contains a set of pages to evaluate $S$, a variable $X$, and ranking models learned in the previous training procedure. For example, to evaluate top 10 pages out of 100 pages in a dataset, the recursive partitioning algorithm first obtains top 50 pages using the weights learned in the first round of the training process. The top 50 pages are sent to the second round and the algorithm extracts top 25 pages using the set of weights

learned in the second round. Finally, the third round evaluates top 10 pages out of the 25 pages using the third set of weights learned in the third round of the training process.

## IV. Evaluation

In this section, we evaluate the accuracy of our ranking system in predicting Google's ranking results. We first explain our experimental methodology. Next, we evaluate our system's overall ranking accuracy. Then, we evaluate the importance of the recursive partitioning ranking algorithm and demonstrate its effectiveness in dealing with the underlying non-linearities. Finally, we evaluate the relative importance of various web page features.

### A. Experimental Setup

Using our crawler, we collect search results from Google for 60 keywords in 4 categories during May 2009, shown in Table III. The four categories are Linux commands, chemical elements, as well as music and astronomy terms. We select these terms and the corresponding keywords to keep our experiments simple and at the same time not lose generality. In particular, the goal is to avoid plural, similar words that Google might take into account, *etc.* Later, in Section V-A, we demonstrate that our approach is applicable to other popular keywords as well.

In each experiment, we randomly select 15 keywords to form the *training set*. We then run these keywords through our ranking system and develop a ranking model. Then, we use the remaining 45 keywords, which we term the *testing set*, to evaluate the accuracy of our model.

| Type | Keywords |
|---|---|
| Linux commands | tcpdump, modprobe, egrep, chmod, dhclient, dmesg, netstat, nslookup, traceroute, rsync, crontab, iconv, telnet, vmstat, xtail |
| Chemicals | potassium, boron, chlorine, manganese, lithium, magnesium, phosphorus, sulfur, fluorine, iodine, helium, zinc, platinum, cobalt, uranium, |
| Music Terms | adagio, arpeggio, baroque, cadence, crescendo, diminuendo, fortissimo, legato, moderato, pianissimo, pizzicato, ritardando, rubato, sforzando, staccato, |
| Astronomy Terms | aphelion, apogee, chromosphere, ecliptic, equinox, photon, pulsar, supernova, zodiac, galaxy, polaris, neutron, perihelion, magnetosphere, perigee. |

TABLE III
Query Keywords

### B. Overall Ranking Accuracy

Here, we evaluate the effectiveness of our ranking system by comparing its ranking results to those of Google. In the experiments, we use linear programming (Section III-B1) as well as the two SVM learning algorithms (Section III-B2), *i.e.*, linear and polynomial. In all cases, we apply three rounds of recursion (Section III-B3).

Figure 2 shows the ranking results of our system under the LP model and the two SVM models, using the ranking features listed in Table I as input variables. In this figure, the x-axis represents the hit rate of the top 10 pages, *i.e.*, the percent of the top 10 pages ranked by Google that are also captured in the top 10 pages output by our system. The y-axis shows the percent of pages, distributed over all keywords from the test set, that satisfy the given hit rate. For example, the point (60,100) in the figure for the SVM polynomial algorithm means that for all 45 keywords from the test set, our algorithm managed to always (100% of time) correctly predict at least 6 pages out of the top ten ones reported by Google (hence, 60%). In general, the closer the curve is to the upper right corner, the better the result is.

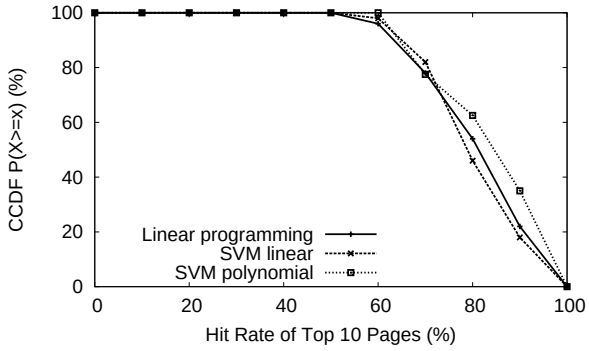



Fig. 2. Comparison of Ranking Models

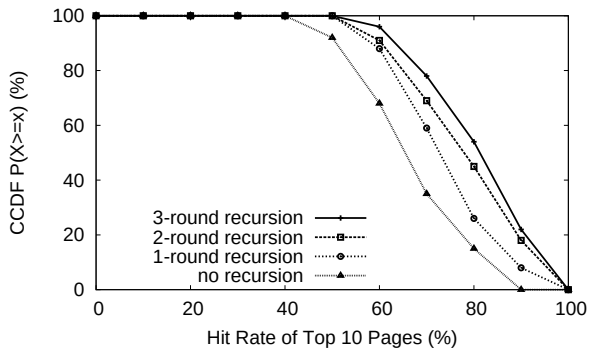
Figure 2 shows that the linear programming model achieves accuracy comparable to SVM-linear, and has only slightly lower accuracy compared to the SVM-polynomial model. In particular, for the LP model, 78% of the keywords experienced a hit rate greater than 70%, and 54% of the keywords experienced a hit rate greater than 80%. Despite the slight lag behind the polynomial model, the linear model is much more practical and convenient because it provides human-readable insights (feature weights in our case). Hence, we use linear programming models in the rest of our experiments. An exception is the next section, where our goal is to understand the role of recursive partitioning ranking. Hence, we explore this issue in the context of the non-linear algorithm as well.

### C. The Power of Recursive Partitioning Ranking

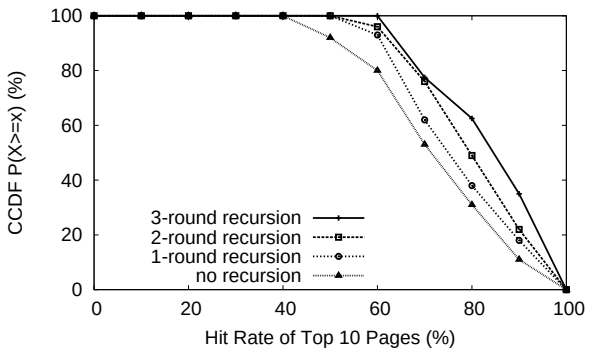
Several factors potentially introduce non-linear effects in Google's ranking results. First, it is more than likely that Google uses non-linear functions in their ranking algorithms. Second, it is preferable for a search engine to keep multiple indices (*e.g.*, one for the most accessed top 20 pages, another one for pages 20-40, *etc.*), instead of having a huge index. These indices can be ranked independently and then merged together. Hence, this can add another level of non-linearity that can further complicate the reverse-engineering problem.

Our goal here is to understand the role of the recursive partitioning and its ability to "smooth out" these non-linearities. Hence, we compare the performance of recursive and non-recursive ranking approaches. We study both linear and polynomial models. Our hypothesis is that recursion should be much more effective in the linear case, because the polynomial model should be able to track non-linearities more effectively. We validate this hypothesis below.

Figure 3 compares the results of the LP model and the SVM-polynomial model without recursive partitioning and with 1, 2, 3 rounds of recursive partitioning. We make the following observations. (1) Without recursive partitioning ranking, the LP model achieves lower prediction accuracy than the SVM-Polynomial model. For example, the former predicted 8 out of the top 10 pages by Google for 18% of the keywords, while the later predicted 8 out of the top 10 pages by Google for 32% of the keywords. (2) Recursive partitioning significantly improves the ranking accuracy for both models.




(a) Linear Programming




(b) SVM-Polynomial

Fig. 3. Power of Recursive Partitioning

The improvement reaches diminishing return after 3 rounds. The average hit rate after 3 rounds of recursion improves by 20% over the no recursion algorithm. This also suggests that Google's ranking function is not linear (at least of the features we studied.) (2) After 3 rounds of recursive partitioning, the LP model achieves similar accuracy as the SVM-polynomial model. Hence, it is sufficient to analyze the linear model which gives human-readable outputs as the weights of the linear model directly reflect the relative importune of different features. We next investigate the weights of the features in the linear model.

### D. Relative Importance of Features

We next analyze the relative importance of different ranking features towards contributing to the overall ranking of a page. Since in the LP model the ranking function is simply a linear combination of all the ranking features, the relative importance of them boils down to the relative values of the weights in the linear function.

Figure 4 shows the weights of the linear equation from the LP and SVM ranking models, sorted in the decreasing order, for the three recursion rounds. The first insight is that the dominant features (*i.e.*, the first 7 on x-axis) carry larger weights in the first two recursion rounds, than in the last. This effect is particularly pronounced in Figure 4(b), for the SVM-linear model. This suggests that these features are dominant in "pushing" web pages to the top 50 or 20 pages, yet in order to get into the top 10 ones, other factors (including those that are in general less valued) become relatively more important as well.

We observe that despite some disagreement in the two linear models, the features weights in the two models are highly correlated. In particular, the first 5 features are in the same order for both algorithms. Not surprisingly, page rank is the dominant factor in both cases. Nevertheless, HOST (keyword appearing in the hostname), TITLE (keyword appearing in the title tag of the HTML header), M_DES (keyword appearing in the meta-description tag), and PATH (keyword appearing in the path segment of the URL), are the other leading factors, respectively.

Figure 4 also shows a high correlation among the factors that have little to no impact (the bottom of the x-axis). In particular, AGE (the age of the web page) is at the bottom of the list; on the contrary, D_AGE, the *domain* age, is an important parameter valued by the ranking system. This is not a surprise since the domain age in general increases its reputation. Continuing with the factors at the bottom of the list, both algorithms show that M_KEY (keyword in the meta-keyword tag) has low impact; on the contrary, M_DES (keyword in the meta-description tag) has a higher impact, as we explained above. Further, D_SIZE (domain size) and IMG (keyword in the image tag) do not have much influence.

## V. Case Studies

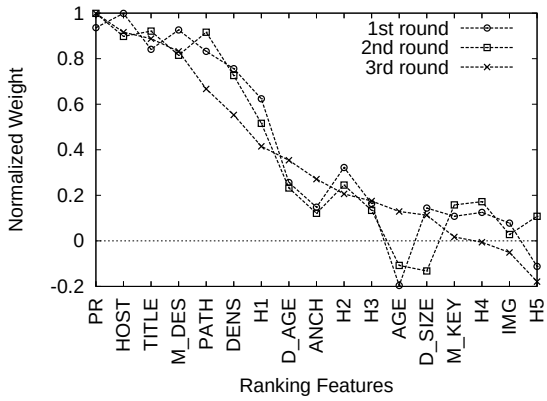
Here, we perform several case studies. In particular, we first focus on isolating the content score and explore the general effects of dealing with a smaller number of ranking features. Next, we evaluate the ability of our approach to add new ranking features in a methodical way; we explore whether a given category, *e.g.*, blogs, is a factor considered by the search engine ranking algorithm. Finally, we explore whether HTML syntax errors affect ranking results or not.

### A. Isolating Subsets of Ranking Features

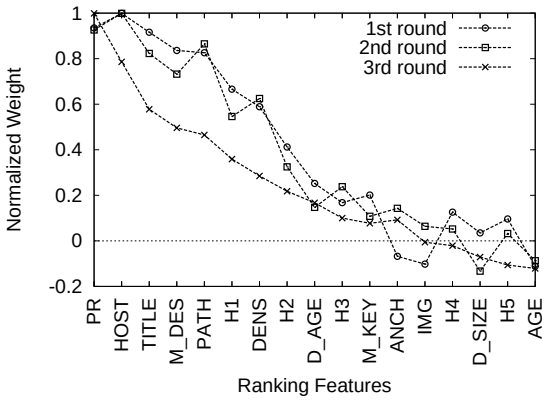
In this case study, we explore how well our ranking system works, *i.e.,* approximates Google, when focusing on a *subset* of parameters; in particular, all but Page features shown in Table I. These include the features that belong to the URL, Domain, Header, Body, Heading, and Link groups. To do so, we need to decouple the Page ranking features from others. We achieve this by crawling only "young" web pages (*i.e.,* past 24 hours) using Google search API `qdr:d`. Indeed, by looking only at 'young web pages', we manage to effectively remove the age factor. This is because all pages are of the same age. Moreover, we manage to remove the pagerank factor, because it takes much more than one day for a web page to obtain its page rank, as we also evaluate experimentally.

In order to find enough new web pages (at least 100 per keyword) that are generated in recent 24 hours, we necessarily abandon the keywords shown in Table III. This is simply because not enough new web pages that contain these keywords (and are indexed by Google) appear daily. Hence, we turn our attention to more popular keywords. In particular, we query the popular `Google Trends` website [5] and obtain a list of 60 popular keywords for our experiment. We then query the above advanced Google API (`qdr:d`), which gives us only new web pages, using the 60 keywords. If a given keyword cannot return 100 or more links, we abandon such a keyword. Finally, we randomly select 15 keywords as the training set, and the remaining 45 as the testing set.

Figure 5 shows the ranking results using the LP model and three iterations of recursive partitioning and ranking.




(a) Linear Programming




(b) SVM-Linear

Fig. 4. Weights in Different Rounds

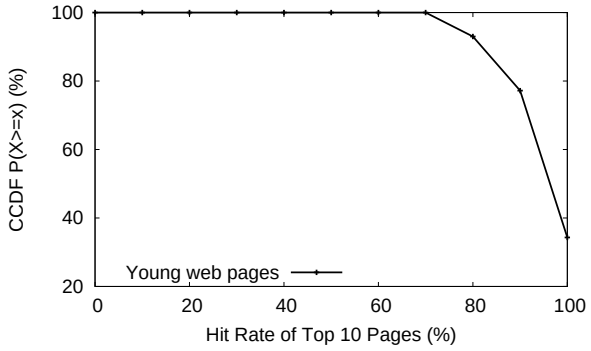



Fig. 5. Ranking Results for "Young" Pages

The x-axis shows our ranking system's hit rate of top 10 pages returned by Google. We observe that the hit rate is about 80% for 92% of keywords, and about 90% for 77% of keywords. These results are much better than those in Figure 2, suggesting that our ranking system is more accurate, *i.e.,* approximates Google's ranking results much better, when only content related features affect the ranking. This case study suggests when the parameters are more specific, *i.e.,* fewer and closely related, our ranking engine performs better.

### *B. Negative Bias Toward Blogs*

In this case study, we show that our methodology can be used to validate new ranking features. In this particular case, we introduce a new ranking feature that represents a web page *category*, *e.g.*, news, music, blogs, *etc*. Our goal is to discover if there exists positive or negative bias toward some categories. For example, there are rumors on the Internet saying that blogs rank lower than regular web pages by Google. We validate this belief using our ranking system.

To characterize web sites into different categories, we apply a simple keyword approach. For example, in case of the Blog category, we looked at keyword "blog" in the URL, title tag, and Google's snippet, in a case insensitive manner. If the keyword appears in any of the three places (regardless of how many times it appears), we categorized the page as a blog. We apply a similar approach to all other categories using different keywords.

Next, we add a new feature to our ranking algorithm, named CAT (which stands for category) and explore each of the categories *in isolation*. For example, when we explore the news category, we test both potential positive and negative bias towards the given category; (by assigning a positive or negative unit value to the CAT feature parameter in the linear model for pages from the given category); at the same time, all other non-news-related web pages that are returned by the search engine have the CAT feature turned off. We then run our ranking system with all features (including CAT) using the LP model and three iterations of recursive partitioning and ranking.

For all explored categories, except for Blogs, the assigned category feature (either positive or negative) has *no* influence whatsoever on the ranking prediction (the result was the same as when we used the original 17 features). Figure 6 compares the ranking results with and without testing the negative bias hypothesis for the Blogs feature. The figure clearly shows that the negative hypothesis indeed shows true for Blogs. Indeed, the prediction results improve when we adopt the negative bias hypothesis for Blogs. The weight for the negative Blog bias is as high as 0.2, which puts it in the top 10 features explored in our paper.

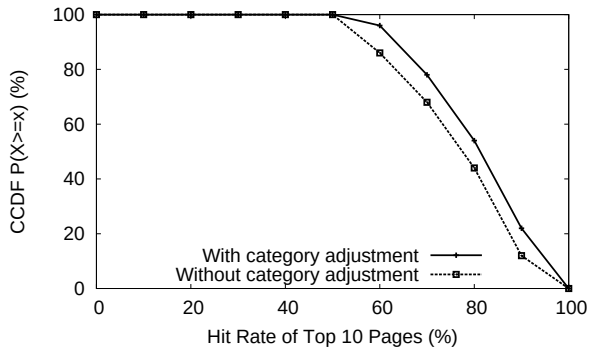



Fig. 6. Category Analysis

For example, in our `iptables` dataset, a blog article titled *'Iptables dependency: why we got there and how we got out'* [10] has high scores in all of our top 3 ranking factors. Specifically, it has a high page rank score of 6, and the target search term appears in the title and URL. However, this blog is ranked 62th out of 100 by Google and it would have been ranked 22th without the negative bias toward blogs. Another example is a blog titled *'Rsync Version 3 Alpha Out - O'Reilly*

*ONLamp Blog'* [13] in our `rsync` dataset has everything it needs to be ranked as a top 10 web page. It has a good page rank score of 4 and the keyword appears in the title, URL and meta description tag. In addition, it also has a good keyword density in its content. However, this blog is ranked 32th by Google while it should have been ranked 8th without the bias.

Thus, we show that our system is capable of validating new conjectures about Google's ranking algorithm and that Google imposes negative bias toward blogs.

### C. HTML syntax errors do not matter

In this final case study, we explore the impact of HTML syntax errors on Google's ranking algorithm. Some SEO experts hypothesised that Google estimate the `quality` of a web page which includes the correctness of HTML syntax [4], [8]. However, we demonstrate that this is not the case by adding this new feature into our ranking system.

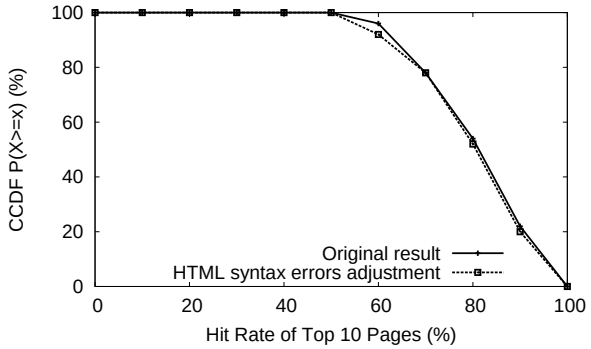



Fig. 7. HTML Syntax Analysis

In this experiment, we use the HTML tidy library program [7] to analyze and count HTML errors of each web page in our data set. This new feature is then added to our ranking system to train new ranking models. In the training set, the number of web pages with one or more syntax errors is 275 out of 1,500 pages (18.33%). In the testing set, the number of pages with syntax errors is 972 out of 4,500 (21.60%).

Figure 7 compares the results from the original and the new ranking models. The figure shows that the performance of the new model is very close to the original one. In addition, the weights of the new ranking feature in each of the 3 rounds are very close to zero (0.068, 0.056 and 0.033, respectively). This indicates that HTML syntax errors have very little to no impact on a web page's Google ranking.

## VI. Conclusions

In this paper, we study the problem of reverse engineering Google's ranking algorithm. Even though Google's internal ranking function can be very complex, we demonstrate that it is possible to approximate Google's organic search results by adopting a linear model trained with a linear programming optimizer along with a recursive partitioning scheme. We performed large-scale experiments using over 6000 web pages and showed that our ranking system is capable of predicting Google's ranking results with high accuracy. Specifically, our system is able to correctly predict 7 out of the top 10 pages for 78% of evaluated keywords.

Using our ranking system, we revealed the relative importance of ranking features in Google's ranking function. In particular, page rank is the dominant factor, followed by the search keyword appearing in the hostname, in the title tag of the HTML header, in the meta-description tag, in the path segment of the URL, as the other leading factors. Such revelation provides guidelines for SEOs and webmasters to optimize their web pages and obtain a better position in Google's search result pages. Moreover, we showed how to use our system to validate or disapprove new ranking features and to evaluate search engine ranking results for possible ranking bias. In particular, we used our ranking system to confirm the rumors in the Internet that the Google's ranking is biased against blogs.

## References


[1] "A breakdown of Google's ranking factors," http://www.pronetadvertising.com/articles/a-breakdown-of-googles-ranking-factors.html.

[2] "Google Algorithm's Top Ranking Factors," http://www.accuracast.com/seo-weekly/ranking-factors.php.

[3] "Google Ranking Factors," http://www.vaughns-1-pagers.com/internet/google-ranking-factors.htm.

[4] "Google SEO Test Google Prefers Valid HTML & CSS," http://www.hobo-web.co.uk/seo-blog/index.php/official-google-prefers-valid-html-css/.

[5] "Google Trends," http://www.google.com/trends.

[6] "Google's Top Search Engine Ranking Factors," http://lornali.com/online-marketing/seo/googles-top-search-engine-ranking-factors.

[7] "HTML Tidy Library Project," http://tidy.sourceforge.net/.

[8] "HTML Validation: the hidden key to SEO," http://www.rsspieces.com/html-validation-the-hidden-key-to-seo.

[9] "ILOG CPLEX: High-performance software for mathematical programming and optimization," http://www.ilog.com/products/cplex/.

[10] "Iptables dependency: why we got there and how we got out," http://www.zimbrablog.com/blog/archives/2005/11/iptables-dependency-why-we-got-there-and-how- we-got-out.html.

[11] "PageRank on Google Toolbar," http://www.google.com/support/toolbar/bin/answer.py?hl=en&answer=79837.

[12] "PHP Simple HTML DOM Parser," http://simplehtmldom.sourceforge.net.

[13] "Rsync Version 3 Alpha Out - O'Reilly ONLamp Blog," http://www.oreillynet.com/onlamp/blog/2007/10/rsync_version_3_alpha_out.html.

[14] "Search Engine Ranking Factors," http://www.seomoz.org/article/search-ranking-factors.

[15] "SeoPros.org," http://www.seopros.org/.

[16] "SVM-rank Support Vector Machine," http://www.cs.cornell.edu/People/tj/svm_Flight/svm_rank.html.

[17] "Top 10 Most Important Google Ranking Factors," http://blogs.myspace.com/index.cfm?fuseaction=blog.view&friendId=21196&blogId=493022330.

[18] "Top Google Ranking Factors," http://www.squidoo.com/topGoogleRankingFactors.

[19] "TOPSEOs.com," http://www.topseos.com/.

[20] A. A. Benczúr, K. Csalogány, T. Sarlós, and M. Uher, "SpamRank–Fully Automatic Link Spam Detection," in *Proc. AIRWeb*, 2005, pp. 25–38.

[21] J. Cho and S. Roy, "Impact of search engines on page popularity," in *Proc. WWW '04: Proceedings of the 13th international conference on World Wide Web*. New York, NY, USA: ACM, 2004, pp. 20–29.

[22] Z. Gyongyi, P. Berkhin, H. Garcia-Molina, and J. Pedersen, "Link spam detection based on mass estimation," in *Proc. VLDB '06: Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006, pp. 439–450.

[23] T. Joachims, "Making Large–Scale SVM Learning Practical," in *Proc. Advances in Kernel Methods — Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 169–184.

[24] M. Moran and B. Hunt, *Search Engine Marketing, Inc.: Driving Search Traffic to Your Company's Web Site*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[25] N. V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, New York., 2000.

[26] B. Wu and B. D. Davison, "Identifying link farm spam pages," in *Proc. WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*. New York, NY, USA: ACM, 2005, pp. 820–829.