Beyond PageRank: Machine Learning for Static Ranking

Matthew Richardson
Microsoft Research
One Microsoft Way
Redmond, WA 98052
+1 (425) 722-3325
mattri@microsoft.com

Amit Prakash
MSN
One Microsoft Way
Redmond, WA 98052
+1 (425) 705-6015
amitp@microsoft.com

Eric Brill
Microsoft Research
One Microsoft Way
Redmond, WA 98052
+1 (425) 705-4992
brill@microsoft.com

ABSTRACT
Since the publication of Brin and Page's paper on PageRank, many in the Web community have depended on PageRank for the static (query-independent) ordering of Web pages. We show that we can significantly outperform PageRank using features that are independent of the link structure of the Web. We gain a further boost in accuracy by using data on the frequency at which users visit Web pages. We use RankNet, a ranking machine learning algorithm, to combine these and other static features based on anchor text and domain characteristics. The resulting model achieves a static ranking pairwise accuracy of 67.3% (vs. 56.7% for PageRank or 50% for random).

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning. H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval.

General Terms
Algorithms, Measurement, Performance, Experimentation.

Keywords
Static ranking, search engines, PageRank, RankNet, relevance

1. INTRODUCTION
Over the past decade, the Web has grown exponentially in size. Unfortunately, this growth has not been isolated to good-quality pages. The number of incorrect, spamming, and malicious (e.g., phishing) sites has also grown rapidly. The sheer number of both good and bad pages on the Web has led to an increasing reliance on search engines for the discovery of useful information. Users rely on search engines not only to return pages related to their search query, but also to separate the good from the bad, and order results so that the best pages are suggested first.

To date, most work on Web page ranking has focused on improving the ordering of the results returned to the user (query-dependent ranking, or *dynamic ranking*). However, having a good query-independent ranking (*static ranking*) is also crucially important for a search engine. A good static ranking algorithm provides numerous benefits:

WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-323-9/06/0005.

- **Relevance**: The static rank of a page provides a general indicator to the overall quality of the page. This is a useful input to the dynamic ranking algorithm.
- **Efficiency**: Typically, the search engine's index is ordered by static rank. By traversing the index from high-quality to low-quality pages, the dynamic ranker may abort the search when it determines that no later page will have as high of a dynamic rank as those already found. The more accurate the static rank, the better this early-stopping ability, and hence the quicker the search engine may respond to queries.
- **Crawl Priority**: The Web grows and changes as quickly as search engines can crawl it. Search engines need a way to prioritize their crawl—to determine which pages to re-crawl, how frequently, and how often to seek out new pages. Among other factors, the static rank of a page is used to determine this prioritization. A better static rank thus provides the engine with a higher quality, more up-to-date index.

Google is often regarded as the first commercially successful search engine. Their ranking was originally based on the PageRank algorithm [5][27]. Due to this (and possibly due to Google's promotion of PageRank to the public), PageRank is widely regarded as the best method for the static ranking of Web pages.

Though PageRank has historically been thought to perform quite well, there has yet been little academic evidence to support this claim. Even worse, there has recently been work showing that PageRank may not perform any better than other simple measures on certain tasks. Upstill et al. have found that for the task of finding home pages, the number of pages linking to a page and the type of URL were as, or more, effective than PageRank [32]. They found similar results for the task of finding high quality companies [31]. PageRank has also been used in systems for TREC's "very large collection" and "Web track" competitions, but with much less success than had been expected [17]. Finally, Amento et al. [1] found that simple features, such as the number of pages on a site, performed as well as PageRank.

Despite these, the general belief remains among many, both academic and in the public, that PageRank is an essential factor for a good static rank. Failing this, it is still assumed that using the link structure is crucial, in the form of the number of inlinks or the amount of anchor text.

In this paper, we show there are a number of simple url- or page-based features that significantly outperform PageRank (for the purposes of statically ranking Web pages) despite ignoring the

structure of the Web. We combine these and other static features using machine learning to achieve a ranking system that is significantly better than PageRank (in pairwise agreement with human labels).

A machine learning approach for static ranking has other advantages besides the quality of the ranking. Because the measure consists of many features, it is harder for malicious users to manipulate it (i.e., to raise their page's static rank to an undeserved level through questionable techniques, also known as Web spamming). This is particularly true if the feature set is not known. In contrast, a single measure like PageRank can be easier to manipulate because spammers need only concentrate on one goal: how to cause more pages to point to their page. With an algorithm that learns, a feature that becomes unusable due to spammer manipulation will simply be reduced or removed from the final computation of rank. This flexibility allows a ranking system to rapidly react to new spamming techniques.

A machine learning approach to static ranking is also able to take advantage of any advances in the machine learning field. For example, recent work on adversarial classification [12] suggests that it may be possible to explicitly model the Web page spammer's (the adversary) actions, adjusting the ranking model in advance of the spammer's attempts to circumvent it. Another example is the elimination of outliers in constructing the model, which helps reduce the effect that unique sites may have on the overall quality of the static rank. By moving static ranking to a machine learning framework, we not only gain in accuracy, but also gain in the ability to react to spammer's actions, to rapidly add new features to the ranking algorithm, and to leverage advances in the rapidly growing field of machine learning.

Finally, we believe there will be significant advantages to using this technique for other domains, such as searching a local hard drive or a corporation's intranet. These are domains where the link structure is particularly weak (or non-existent), but there are other domain-specific features that could be just as powerful. For example, the author of an intranet page and his/her position in the organization (e.g., CEO, manager, or developer) could provide significant clues as to the importance of that page. A machine learning approach thus allows rapid development of a good static algorithm in new domains.

This paper's contribution is a systematic study of static features, including PageRank, for the purposes of (statically) ranking Web pages. Previous studies on PageRank typically used subsets of the Web that are significantly smaller (e.g., the TREC VLC2 corpus, used by many, contains only 19 million pages). Also, the performance of PageRank and other static features has typically been evaluated in the context of a complete system for dynamic ranking, or for other tasks such as question answering. In contrast, we explore the use of PageRank and other features for the direct task of statically ranking Web pages.

We first briefly describe the PageRank algorithm. In Section 3 we introduce RankNet, the machine learning technique used to combine static features into a final ranking. Section 4 describes the static features. The heart of the paper is in Section 5, which presents our experiments and results. We conclude with a discussion of related and future work.

2. PAGERANK

The basic idea behind PageRank is simple: a link from a Web page to another can be seen as an endorsement of that page. In general, links are made by people. As such, they are indicative of the quality of the pages to which they point – when creating a page, an author presumably chooses to link to pages deemed to be of good quality. We can take advantage of this linkage information to order Web pages according to their perceived quality.

Imagine a Web surfer who jumps from Web page to Web page, choosing with uniform probability which link to follow at each step. In order to reduce the effect of dead-ends or endless cycles the surfer will occasionally jump to a random page with some small probability α, or when on a page with no out-links. If averaged over a sufficient number of steps, the probability the surfer is on page *j* at some point in time is given by the formula:

$$P(j) = \frac{(1-\alpha)}{N} + \alpha \sum_{i \in \mathbf{B}_j} \frac{P(i)}{|\mathbf{F}_i|} \qquad (1)$$

Where $\mathbf{F}_i$ is the set of pages that page *i* links to, and $\mathbf{B}_j$ is the set of pages that link to page *j*. The PageRank score for node *j* is defined as this probability: *PR(j)=P(j)*. Because equation (1) is recursive, it must be iteratively evaluated until *P(j)* converges (typically, the initial distribution for *P(j)* is uniform). The intuition is, because a random surfer would end up at the page more frequently, it is likely a better page. An alternative view for equation (1) is that each page is assigned a quality, *P(j)*. A page "gives" an equal share of its quality to each page it points to.

PageRank is computationally expensive. Our collection of 5 billion pages contains approximately 370 billion links. Computing PageRank requires iterating over these billions of links multiple times (until convergence). It requires large amounts of memory (or very smart caching schemes that slow the computation down even further), and if spread across multiple machines, requires significant communication between them. Though much work has been done on optimizing the PageRank computation (see e.g., [25] and [6]), it remains a relatively slow, computationally expensive property to compute.

3. RANKNET

Much work in machine learning has been done on the problems of classification and regression. Let $\mathbf{X}=\{\mathbf{x}_i\}$ be a collection of feature vectors (typically, a feature is any real valued number), and $\mathbf{Y}=\{y_i\}$ be a collection of associated classes, where y_i is the class of the object described by feature vector $\mathbf{x}_i$. The classification problem is to learn a function *f* that maps $y_i=f(\mathbf{x}_i)$, for all *i*. When y_i is real-valued as well, this is called regression.

Static ranking can be seen as a regression problem. If we let $\mathbf{x}_i$ represent features of page *i*, and y_i be a value (say, the rank) for each page, we could learn a regression function that mapped each page's features to their rank. However, this over-constrains the problem we wish to solve. All we really care about is the order of the pages, not the actual value assigned to them.

Recent work on this *ranking* problem [7][13][18] directly attempts to optimize the ordering of the objects, rather than the value assigned to them. For these, let $\mathbf{Z}=\{<i,j>\}$ be a collection of pairs of items, where item *i* should be assigned a higher value than item *j*. The goal of the ranking problem, then, is to learn a function *f* such that,

$$\forall \langle i, j \rangle \in \mathbf{Z}, f(\mathbf{x}_i) > f(\mathbf{x}_j)$$

Note that, as with learning a regression function, the result of this process is a function (f) that maps feature vectors to real values. This function can still be applied anywhere that a regression-learned function could be applied. The only difference is the technique used to learn the function. By directly optimizing the ordering of objects, these methods are able to learn a function that does a better job of ranking than do regression techniques.

We used RankNet [7], one of the aforementioned techniques for learning ranking functions, to learn our static rank function. RankNet is a straightforward modification to the standard neural network back-prop algorithm. As with back-prop, RankNet attempts to minimize the value of a cost function by adjusting each weight in the network according to the gradient of the cost function with respect to that weight. The difference is that, while a typical neural network cost function is based on the difference between the network output and the desired output, the RankNet cost function is based on the difference between a pair of network outputs. That is, for each pair of feature vectors $\langle i,j \rangle$ in the training set, RankNet computes the network outputs o_i and o_j. Since vector i is supposed to be ranked higher than vector j, the larger is o_j-o_i, the larger the cost.

RankNet also allows the pairs in $\mathbf{Z}$ to be weighted with a confidence (posed as the probability that the pair satisfies the ordering induced by the ranking function). In this paper, we used a probability of one for all pairs. In the next section, we will discuss the features used in our feature vectors, $\mathbf{x}_i$.

4. FEATURES

To apply RankNet (or other machine learning techniques) to the ranking problem, we needed to extract a set of features from each page. We divided our feature set into four, mutually exclusive, categories: page-level (*Page*), domain-level (*Domain*), anchor text and inlinks (*Anchor*), and popularity (*Popularity*). We also optionally used the PageRank of a page as a feature. Below, we describe each of these feature categories in more detail.

PageRank

We computed PageRank on a Web graph of 5 billion crawled pages (and 20 billion known URLs linked to by these pages). This represents a significant portion of the Web, and is approximately the same number of pages as are used by Google, Yahoo, and MSN for their search engines.

Because PageRank is a graph-based algorithm, it is important that it be run on as large a subset of the Web as possible. Most previous studies on PageRank used subsets of the Web that are significantly smaller (e.g. the TREC VLC2 corpus, used by many, contains only 19 million pages)

We computed PageRank using the standard value of 0.85 for α.

Popularity

Another feature we used is the actual popularity of a Web page, measured as the number of times that it has been visited by users over some period of time. We have access to such data from users who have installed the MSN toolbar and have opted to provide it to MSN. The data is aggregated into a count, for each Web page, of the number of users who viewed that page.

Though popularity data is generally unavailable, there are two other sources for it. The first is from proxy logs. For example, a university that requires its students to use a proxy has a record of all the pages they have visited while on campus. Unfortunately, proxy data is quite biased and relatively small.

Another source, internal to search engines, are records of which results their users clicked on. Such data was used by the search engine "Direct Hit", and has recently been explored for dynamic ranking purposes [20]. An advantage of the toolbar data over this is that it contains information about URL visits that are not just the result of a search.

The raw popularity is processed into a number of features such as the number of times a page was viewed and the number of times any page in the domain was viewed. More details are provided in section 5.5.

Anchor text and inlinks

These features are based on the information associated with links *to* the page in question. It includes features such as the total amount of text in links pointing to the page ("anchor text"), the number of unique words in that text, etc.

Page

This category consists of features which may be determined by looking at the page (and its URL) alone. We used only eight, simple features such as the number of words in the body, the frequency of the most common term, etc.

Domain

This category contains features that are computed as averages across all pages in the domain. For example, the average number of outlinks on any page and the average PageRank.

Many of these features have been used by others for ranking Web pages, particularly the anchor and page features. As mentioned, the evaluation is typically for dynamic ranking, and we wish to evaluate the use of them for static ranking. Also, to our knowledge, this is the first study on the use of actual page visitation popularity for static ranking. The closest similar work is on using click-through behavior (that is, which search engine results the users click on) to affect dynamic ranking (see e.g., [20]).

Because we use a wide variety of features to come up with a static ranking, we refer to this as *fRank* (for feature-based ranking). fRank uses RankNet and the set of features described in this section to learn a ranking function for Web pages. Unless otherwise specified, fRank was trained with all of the features.

5. EXPERIMENTS

In this section, we will demonstrate that we can out perform PageRank by applying machine learning to a straightforward set of features. Before the results, we first discuss the data, the performance metric, and the training method.

5.1 Data

In order to evaluate the quality of a static ranking, we needed a "gold standard" defining the correct ordering for a set of pages. For this, we employed a dataset which contains human judgments for 28000 queries. For each query, a number of results are manually assigned a rating, from 0 to 4, by human judges. The rating is meant to be a measure of how relevant the result is for the query, where 0 means "poor" and 4 means "excellent". There are approximately 500k judgments in all, or an average of 18 ratings per query.

The queries are selected by randomly choosing queries from among those issued to the MSN search engine. The probability that a query is selected is proportional to its frequency among all

of the queries. As a result, common queries are more likely to be judged than uncommon queries. As an example of how diverse the queries are, the first four queries in the training set are "chef schools", "chicagoland speedway", "eagles fan club", and "Turkish culture". The documents selected for judging are those that we expected would, on average, be reasonably relevant (for example, the top ten documents returned by MSN's search engine). This provides significantly more information than randomly selecting documents on the Web, the vast majority of which would be irrelevant to a given query.

Because of this process, the judged pages tend to be of higher quality than the average page on the Web, and tend to be pages that will be returned for common search queries. This bias is good when evaluating the quality of static ranking for the purposes of index ordering and returning relevant documents. This is because the most important portion of the index to be well-ordered and relevant is the portion that is frequently returned for search queries. Because of this bias, however, the results in this paper are not applicable to crawl prioritization. In order to obtain experimental results on crawl prioritization, we would need ratings on a random sample of Web pages.

To convert the data from query-dependent to query-independent, we simply removed the query, taking the maximum over judgments for a URL that appears in more than one query. The reasoning behind this is that a page that is relevant for some query and irrelevant for another is probably a decent page and should have a high static rank. Because we evaluated the pages on queries that occur frequently, our data indicates the correct index ordering, and assigns high value to pages that are likely to be relevant to a common query.

We randomly assigned queries to a training, validation, or test set, such that they contained 84%, 8%, and 8% of the queries, respectively. Each set contains all of the ratings for a given query, and no query appears in more than one set. The training set was used to train fRank. The validation set was used to select the model that had the highest performance. The test set was used for the final results.

This data gives us a query-independent ordering of pages. The goal for a static ranking algorithm will be to reproduce this ordering as closely as possible. In the next section, we describe the measure we used to evaluate this.

5.2 Measure

We chose to use *pairwise accuracy* to evaluate the quality of a static ranking. The pairwise accuracy is the fraction of time that the ranking algorithm and human judges agree on the ordering of a pair of Web pages.

If S(x) is the static ranking assigned to page x, and H(x) is the human judgment of relevance for x, then consider the following sets:

$$\mathbf{H_p} = \{x, y : H(x) > H(y)\} \quad \text{and} \quad \mathbf{S_p} = \{x, y : S(x) > S(y)\}$$

The pairwise accuracy is the portion of $\mathbf{H_p}$ that is also contained in $\mathbf{S_p}$:

$$\text{pairwise accuracy} = \frac{\left|\mathbf{H_p} \cap \mathbf{S_p}\right|}{\left|\mathbf{H_p}\right|}$$

This measure was chosen for two reasons. First, the discrete human judgments provide only a partial ordering over Web pages, making it difficult to apply a measure such as the Spearman rank order correlation coefficient (in the pairwise accuracy measure, a pair of documents with the same human judgment does not affect the score). Second, the pairwise accuracy has an intuitive meaning: it is the fraction of pairs of documents that, when the humans claim one is better than the other, the static rank algorithm orders them correctly.

5.3 Method

We trained fRank (a RankNet based neural network) using the following parameters. We used a fully connected 2 layer network. The hidden layer had 10 hidden nodes. The input weights to this layer were all initialized to be zero. The output "layer" (just a single node) weights were initialized using a uniform random distribution in the range [-0.1, 0.1]. We used *tanh* as the transfer function from the inputs to the hidden layer, and a linear function from the hidden layer to the output. The cost function is the pairwise cross entropy cost function as discussed in section 3.

The features in the training set were normalized to have zero mean and unit standard deviation. The same linear transformation was then applied to the features in the validation and test sets.

For training, we presented the network with 5 million pairings of pages, where one page had a higher rating than the other. The pairings were chosen uniformly at random (with replacement) from all possible pairings. When forming the pairs, we ignored the magnitude of the difference between the ratings (the rating spread) for the two URLs. Hence, the weight for each pair was constant (one), and the probability of a pair being selected was independent of its rating spread.

We trained the network for 30 epochs. On each epoch, the training pairs were randomly shuffled. The initial training rate was 0.001. At each epoch, we checked the error on the training set. If the error had increased, then we decreased the training rate, under the hypothesis that the network had probably overshot. The training rate at each epoch was thus set to:

$$\text{Training rate} = \frac{\kappa}{\varepsilon + 1}$$

Where κ is the initial rate (0.001), and ε is the number of times the training set error has increased. After each epoch, we measured the performance of the neural network on the validation set, using 1 million pairs (chosen randomly with replacement). The network with the highest pairwise accuracy on the validation set was selected, and then tested on the test set. We report the pairwise accuracy on the test set, calculated using all possible pairs.

These parameters were determined and fixed before the static rank experiments in this paper. In particular, the choice of initial training rate, number of epochs, and training rate decay function were taken directly from Burges et al [7].

Though we had the option of preprocessing any of the features before they were input to the neural network, we refrained from doing so on most of them. The only exception was the popularity features. As with most Web phenomenon, we found that the distribution of site popularity is Zipfian. To reduce the dynamic range, and hopefully make the feature more useful, we presented the network with both the unpreprocessed, as well as the logarithm, of the popularity features (As with the others, the logarithmic feature values were also normalized to have zero mean and unit standard deviation).

Applying fRank to a document is computationally efficient, taking time that is only linear in the number of input features; it is thus within a constant factor of other simple machine learning methods such as naïve Bayes. In our experiments, computing the fRank for all five billion Web pages was approximately 100 times faster than computing the PageRank for the same set.

5.4 Results

As Table 1 shows, fRank significantly outperforms PageRank for the purposes of static ranking. With a pairwise accuracy of 67.4%, fRank more than doubles the accuracy of PageRank (relative to the baseline of 50%, which is the accuracy that would be achieved by a random ordering of Web pages). Note that one of fRank's input features is the PageRank of the page, so we would expect it to perform no worse than PageRank. The significant increase in accuracy implies that the other features (anchor, popularity, etc.) do in fact contain useful information regarding the overall quality of a page.

Table 1: Basic Results

Technique	*Accuracy (%)*
None (Baseline)	50.00
PageRank	56.70
fRank	**67.43**

There are a number of decisions that go into the computation of PageRank, such as how to deal with pages that have no outlinks, the choice of α, numeric precision, convergence threshold, etc. We were able to obtain a computation of PageRank from a completely independent implementation (provided by Marc Najork) that varied somewhat in these parameters. It achieved a pairwise accuracy of 56.52%, nearly identical to that obtained by our implementation. We thus concluded that the quality of the PageRank is not sensitive to these minor variations in algorithm, nor was PageRank's low accuracy due to problems with our implementation of it.

We also wanted to find how well each feature set performed. To answer this, for each feature set, we trained and tested fRank using only that set of features. The results are shown in Table 2. As can be seen, *every single feature set individually outperformed PageRank* on this test. Perhaps the most interesting result is that the Page-level features had the highest performance out of all the feature sets. This is surprising because these are features that do not depend on the overall graph structure of the Web, nor even on what pages point to a given page. This is contrary to the common belief that the Web graph structure is the key to finding a good static ranking of Web pages.

Table 2: Results for individual feature sets.

Feature Set	*Accuracy (%)*
PageRank	56.70
Popularity	60.82
Anchor	59.09
Page	**63.93**
Domain	59.03
All Features	**67.43**

Because we are using a two-layer neural network, the features in the learned network can interact with each other in interesting, nonlinear ways. This means that a particular feature that appears to have little value in isolation could actually be very important when used in combination with other features. To measure the final contribution of a feature set, in the context of all the other features, we performed an ablation study. That is, for each set of features, we trained a network to contain all of the features *except* that set. We then compared the performance of the resulting network to the performance of the network with all of the features. Table 3 shows the results of this experiment, where the "decrease in accuracy" is the difference in pairwise accuracy between the network trained with all of the features, and the network missing the given feature set.

Table 3: Ablation study. Shown is the decrease in accuracy when we train a network that has all *but* the given set of features. The last line is shows the effect of removing the anchor, PageRank, and domain features, hence a model containing no network or link-based information whatsoever.

Feature Set	*Decrease in Accuracy*
PageRank	0.18
Popularity	0.78
Anchor	0.47
Page	**5.42**
Domain	0.10
Anchor, PageRank & Domain	0.60

The results of the ablation study are consistent with the individual feature set study. Both show that the most important feature set is the Page-level feature set, and the second most important is the popularity feature set.

Finally, we wished to see how the performance of fRank improved as we added features; we wanted to find at what point adding more feature sets became relatively useless. Beginning with no features, we greedily added the feature set that improved performance the most. The results are shown in Table 4. For example, the fourth line of the table shows that fRank using the page, popularity, and anchor features outperformed any network that used the page, popularity, and some other feature set, and that the performance of this network was 67.25%.

Table 4: fRank performance as feature sets are added. At each row, the feature set that gave the greatest increase in accuracy was added to the list of features (i.e., we conducted a greedy search over feature sets).

Feature Set	*Accuracy (%)*
None	50.00
+Page	63.93
+Popularity	66.83
+Anchor	67.25
+PageRank	67.31
+Domain	67.43

Table 5: Top ten URLs for PageRank vs. fRank

PageRank	*fRank*
google.com	google.com
apple.com/quicktime/download	yahoo.com
amazon.com	americanexpress.com
yahoo.com	hp.com
microsoft.com/windows/ie	target.com
apple.com/quicktime	bestbuy.com
mapquest.com	dell.com
ebay.com	autotrader.com
mozilla.org/products/firefox	dogpile.com
ftc.gov	bankofamerica.com

Finally, we present a qualitative comparison of PageRank vs. fRank. In Table 5 are the top ten URLs returned for PageRank and for fRank. PageRank's results are heavily weighted towards technology sites. It contains two QuickTime URLs (Apple's video playback software), as well as Internet Explorer and FireFox URLs (both of which are Web browsers). fRank, on the other hand, contains more consumer-oriented sites such as American Express, Target, Dell, etc. PageRank's bias toward technology can be explained through two processes. First, there are many pages with "buttons" at the bottom suggesting that the site is optimized for Internet Explorer, or that the visitor needs QuickTime. These generally link back to, in these examples, the Internet Explorer and QuickTime download sites. Consequently, PageRank ranks those pages highly. Though these pages are important, they are not as important as it may seem by looking at the link structure alone. One fix for this is to add information about the link to the PageRank computation, such as the size of the text, whether it was at the bottom of the page, etc.

The other bias comes from the fact that the population of Web site authors is different than the population of Web users. Web authors tend to be technologically-oriented, and thus their linking behavior reflects those interests. fRank, by knowing the actual visitation popularity of a site (the popularity feature set), is able to eliminate some of that bias. It has the ability to depend more on where actual Web users visit rather than where the Web site authors have linked.

The results confirm that fRank outperforms PageRank in pairwise accuracy. The two most important feature sets are the *page* and *popularity* features. This is surprising, as the page features consisted only of a few (8) simple features. Further experiments found that, of the page features, those based on the text of the page (as opposed to the URL) performed the best. In the next section, we explore the popularity feature in more detail.

5.5 Popularity Data

As mentioned in section 4, our popularity data came from MSN toolbar users. For privacy reasons, we had access only to an aggregate count of, for each URL, how many times it was visited by any toolbar user. This limited the possible features we could derive from this data. For possible extensions, see section 6.3, future work.

For each URL in our train and test sets, we provided a feature to fRank which was how many times it had been visited by a toolbar user. However, this feature was quite noisy and sparse, particularly for URLs with query parameters (e.g., http://search-.msn.com/results.aspx?q=machine+learning&form=QBHP). One solution was to provide an additional feature which was the number of times any URL at the given domain was visited by a toolbar user. Adding this feature dramatically improved the performance of fRank.

We took this one step further and used the built-in hierarchical structure of URLs to construct many levels of backoff between the full URL and the domain. We did this by using the set of features shown in Table 6.

Table 6: URL functions used to compute the Popularity feature set.

Function	*Example*
Exact URL	cnn.com/2005/tech/wikipedia.html?v=mobile
No Params	cnn.com/2005/tech/wikipedia.html
Page	wikipedia.html
URL-1	cnn.com/2005/tech
URL-2	cnn.com/2005
…	
Domain	cnn.com
Domain+1	cnn.com/2005
…	

Each URL was assigned one feature for each function shown in the table. The value of the feature was the count of the number of times a toolbar user visited a URL, where the function applied to that URL matches the function applied to the URL in question. For example, a user's visit to *cnn.com/2005/sports.html* would increment the *Domain* and *Domain+1* features for the URL *cnn.com/2005/tech/wikipedia.html*.

As seen in Table 7, adding the domain counts significantly improved the quality of the popularity feature, and adding the numerous backoff functions listed in Table 6 improved the accuracy even further.

Table 7: Effect of adding backoff to the popularity feature set

Features	*Accuracy (%)*
URL count	58.15
URL and Domain counts	59.31
All backoff functions (Table 6)	60.82

Backing off to subsets of the URL is one technique for dealing with the sparsity of data. It is also informative to see how the performance of fRank depends on the amount of popularity data that we have collected. In Figure 1 we show the performance of fRank trained with only the popularity feature set vs. the amount of data we have for the popularity feature set. Each day, we receive additional popularity data, and as can be seen in the plot, this increases the performance of fRank. The relation is logarithmic: doubling the amount of popularity data provides a constant improvement in pairwise accuracy.

In summary, we have found that the popularity features provide a useful boost to the overall fRank accuracy. Gathering more popularity data, as well as employing simple backoff strategies, improve this boost even further.

5.6 Summary of Results

The experiments provide a number of conclusions. First, fRank performs significantly better than PageRank, even without any information about the Web graph. Second, the page level and popularity features were the most significant contributors to pairwise accuracy. Third, by collecting more popularity data, we can continue to improve fRank's performance.

The popularity data provides two benefits to fRank. First, we see that qualitatively, fRank's ordering of Web pages has a more favorable bias than PageRank's. fRank's ordering seems to correspond to what Web users, rather than Web page authors, prefer. Second, the popularity data is more timely than PageRank's link information. The toolbar provides information about which Web pages people find interesting right now, whereas links are added to pages more slowly, as authors find the time and interest.

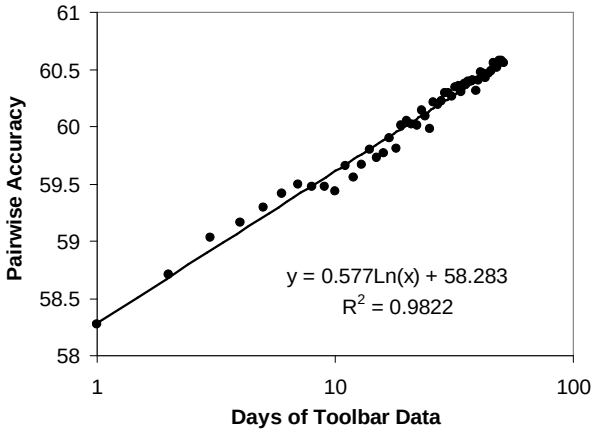

Figure 1: Relation between the amount of popularity data and the performance of the popularity feature set. Note the x-axis is a logarithmic scale.

6. RELATED AND FUTURE WORK

6.1 Improvements to PageRank

Since the original PageRank paper, there has been work on improving it. Much of that work centers on speeding up and parallelizing the computation [15][25].

One recognized problem with PageRank is that of *topic drift*: A page about "dogs" will have high PageRank if it is linked to by many pages that themselves have high rank, regardless of their topic. In contrast, a search engine user looking for good pages about dogs would likely prefer to find pages that are pointed to by many pages that are themselves about dogs. Hence, a link that is "on topic" should have higher weight than a link that is not. Richardson and Domingos's Query Dependent PageRank [29] and Haveliwala's Topic-Sensitive PageRank [16] are two approaches that tackle this problem.

Other variations to PageRank include differently weighting links for inter- vs. intra-domain links, adding a backwards step to the random surfer to simulate the "back" button on most browsers [24] and modifying the jump probability (α) [3]. See Langville and Meyer [23] for a good survey of these, and other modifications to PageRank.

6.2 Other related work

PageRank is not the only link analysis algorithm used for ranking Web pages. The most well-known other is HITS [22], which is used by the Teoma search engine [30]. HITS produces a list of *hubs* and *authorities*, where hubs are pages that point to many authority pages, and authorities are pages that are pointed to by many hubs. Previous work has shown HITS to perform comparably to PageRank [1].

One field of interest is that of static index pruning (see e.g., Carmel et al. [8]). Static index pruning methods reduce the size of the search engine's index by removing documents that are unlikely to be returned by a search query. The pruning is typically done based on the frequency of query terms. Similarly, Pandey and Olston [28] suggest crawling pages frequently if they are likely to incorrectly appear (or not appear) as a result of a search. Similar methods could be incorporated into the static rank (e.g., how many frequent queries contain words found on this page).

Others have investigated the effect that PageRank has on the Web at large [9]. They argue that pages with high PageRank are more likely to be found by Web users, thus more likely to be linked to, and thus more likely to maintain a higher PageRank than other pages. The same may occur for the popularity data. If we increase the ranking for popular pages, they are more likely to be clicked on, thus further increasing their popularity. Cho et al. [10] argue that a more appropriate measure of Web page quality would depend on not only the current link structure of the Web, but also on the change in that link structure. The same technique may be applicable to popularity data: the change in popularity of a page may be more informative than the absolute popularity.

One interesting related work is that of Ivory and Hearst [19]. Their goal was to build a model of Web sites that are considered high quality from the perspective of "content, structure and navigation, visual design, functionality, interactivity, and overall experience". They used over 100 page level features, as well as features encompassing the performance and structure of the site. This let them qualitatively describe the qualities of a page that make it appear attractive (e.g., rare use of italics, at least 9 point font, …), and (in later work) to build a system that assists novel Web page authors in creating quality pages by evaluating it according to these features. The primary differences between this work and ours are the goal (discovering what constitutes a good Web page vs. ordering Web pages for the purposes of Web search), the size of the study (they used a dataset of less than 6000 pages vs. our set of 468,000), and our comparison with PageRank.

Nevertheless, their work provides insights to additional useful static features that we could incorporate into fRank in the future.

Recent work on incorporating novel features into dynamic ranking includes that by Joachims et al. [21], who investigate the use of implicit feedback from users, in the form of which search engine results are clicked on. Craswell et al. [11] present a method for determining the best transformation to apply to query independent features (such as those used in this paper) for the purposes of improving dynamic ranking. Other work, such as Boyan et al. [4] and Bartell et al. [2] apply machine learning for the purposes of improving the overall relevance of a search engine (i.e., the dynamic ranking). They do not apply their techniques to the problem of static ranking.

6.3 Future work

There are many ways in which we would like to extend this work. First, fRank uses only a small number of features. We believe we could achieve even more significant results with more features. In particular the existence, or lack thereof, of certain words could prove very significant (for instance, "under construction" probably signifies a low quality page). Other features could include the number of images on a page, size of those images, number of layout elements (tables, divs, and spans), use of style sheets, conforming to W3C standards (like XHTML 1.0 Strict), background color of a page, etc.

Many pages are generated dynamically, the contents of which may depend on parameters in the URL, the time of day, the user visiting the site, or other variables. For such pages, it may be useful to apply the techniques found in [26] to form a static approximation for the purposes of extracting features. The resulting grammar describing the page could itself be a source of additional features describing the complexity of the page, such as how many non-terminal nodes it has, the depth of the grammar tree, etc.

fRank allows one to specify a confidence in each pairing of documents. In the future, we will experiment with probabilities that depend on the difference in human judgments between the two items in the pair. For example, a pair of documents where one was rated 4 and the other 0 should have a higher confidence than a pair of documents rated 3 and 2.

The experiments in this paper are biased toward pages that have higher than average quality. Also, fRank with all of the features can only be applied to pages that have already been crawled. Thus, fRank is primarily useful for index ordering and improving relevance, not for directing the crawl. We would like to investigate a machine learning approach for crawl prioritization as well. It may be that a combination of methods is best: for example, using PageRank to select the best 5 billion of the 20 billion pages on the Web, then using fRank to order the index and affect search relevancy.

Another interesting direction for exploration is to incorporate fRank and page-level features directly into the PageRank computation itself. Work on biasing the PageRank jump vector [16], and transition matrix [29], have demonstrated the feasibility and advantages of such an approach. There is reason to believe that a direct application of [29], using the fRank of a page for its "relevance", could lead to an improved overall static rank.

Finally, the popularity data can be used in other interesting ways. The general surfing and searching habits of Web users varies by time of day. Activity in the morning, daytime, and evening are often quite different (e.g., reading the news, solving problems, and accessing entertainment, respectively). We can gain insight into these differences by using the popularity data, divided into segments of the day. When a query is issued, we would then use the popularity data matching the time of query in order to do the ranking of Web pages. We also plan to explore popularity features that use more than just the counts of how often a page was visited. For example, how long users tended to dwell on a page, did they leave the page by clicking a link or by hitting the back button, etc. Fox et al. did a study that showed that features such as this can be valuable for the purposes of dynamic ranking [14]. Finally, the popularity data could be used as the label rather than as a feature. Using fRank in this way to predict the popularity of a page may useful for the tasks of relevance, efficiency, and crawl priority. There is also significantly more popularity data than human labeled data, potentially enabling more complex machine learning methods, and significantly more features.

7. CONCLUSIONS

A good static ranking is an important component for today's search engines and information retrieval systems. We have demonstrated that PageRank does not provide a very good static ranking; there are many simple features that individually out perform PageRank. By combining many static features, fRank achieves a ranking that has a significantly higher pairwise accuracy than PageRank alone. A qualitative evaluation of the top documents shows that fRank is less technology-biased than PageRank; by using popularity data, it is biased toward pages that Web users, rather than Web authors, visit. The machine learning component of fRank gives it the additional benefit of being more robust against spammers, and allows it to leverage further developments in the machine learning community in areas such as adversarial classification. We have only begun to explore the options, and believe that significant strides can be made in the area of static ranking by further experimentation with additional features, other machine learning techniques, and additional sources of data.

8. ACKNOWLEDGMENTS

Thank you to Marc Najork for providing us with additional PageRank computations and to Timo Burkard for assistance with the popularity data. Many thanks to Chris Burges for providing code and significant support in using training RankNets. Also, we thank Susan Dumais and Nick Craswell for their edits and suggestions.

9. REFERENCES

[1] B. Amento, L. Terveen, and W. Hill. Does "authority" mean quality? Predicting expert quality ratings of Web documents. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2000.

[2] B. Bartell, G. Cottrell, and R. Belew. Automatic combination of multiple ranked retrieval systems. In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1994.

[3] P. Boldi, M. Santini, and S. Vigna. PageRank as a function of the damping factor. In Proceedings of the International World Wide Web Conference, May 2005.

[4] J. Boyan, D. Freitag, and T. Joachims. A machine learning architecture for optimizing web search engines. In AAAI Workshop on Internet Based Information Systems, August 1996.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Proceedings of the Seventh International Wide Web Conference, Brisbane, Australia, 1998. Elsevier.

[6] A. Broder, R. Lempel, F. Maghoul, and J. Pederson. Efficient PageRank approximation via graph aggregation. In Proceedings of the International World Wide Web Conference, May 2004.

[7] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender. Learning to rank using gradient descent. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 2005.

[8] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. S. Maarek, and A. Soffer. Static index pruning for information retrieval systems. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 43-50, New Orleans, Louisiana, USA, September 2001.

[9] J. Cho and S. Roy. Impact of search engines on page popularity. In Proceedings of the International World Wide Web Conference, May 2004.

[10] J. Cho, S. Roy, R. Adams. Page Quality: In search of an unbiased web ranking. In Proceedings of the ACM SIGMOD 2005 Conference. Baltimore, Maryland. June 2005.

[11] N. Craswell, S. Robertson, H. Zaragoza, and M. Taylor. Relevance weighting for query independent evidence. In Proceedings of the 28th Annual Conference on Research and Development in Information Retrieval (SIGIR), August, 2005.

[12] N. Dalvi, P. Domingos, Mausam, S. Sanghai, D. Verma. Adversarial Classification. In Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining (pp. 99-108), Seattle, WA, 2004.

[13] O. Dekel, C. Manning, and Y. Singer. Log-linear models for label-ranking. In Advances in Neural Information Processing Systems 16. Cambridge, MA: MIT Press, 2003.

[14] S. Fox, K S. Fox, K. Karnawat, M. Mydland, S. T. Dumais and T. White (2005). Evaluating implicit measures to improve the search experiences. In the ACM Transactions on Information Systems, 23(2), pp. 147-168. April 2005.

[15] T. Haveliwala. Efficient computation of PageRank. Stanford University Technical Report, 1999.

[16] T. Haveliwala. Topic-sensitive PageRank. In Proceedings of the International World Wide Web Conference, May 2002.

[17] D. Hawking and N. Craswell. Very large scale retrieval and Web search. In D. Harman and E. Voorhees (eds), The TREC Book. MIT Press.

[18] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In Proceedings of the Ninth International Conference on Artificial Neural Networks, pp. 97-102. 1999.

[19] M. Ivory and M. Hearst. Statistical profiles of highly-rated Web sites. In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, 2002.

[20] T. Joachims. Optimizing search engines using clickthrough data. In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), 2002.

[21] T. Joachims, L. Granka, B. Pang, H. Hembrooke, and G. Gay. Accurately Interpreting Clickthrough Data as Implicit Feedback. In Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR), 2005.

[22] J. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM 46:5, pp. 604-32. 1999.

[23] A. Langville and C. Meyer. Deeper inside PageRank. Internet Mathematics 1(3):335-380, 2004.

[24] F. Matthieu and M. Bouklit. The effect of the back button in a random walk: application for PageRank. In Alternate track papers and posters of the Thirteenth International World Wide Web Conference, 2004.

[25] F. McSherry. A uniform approach to accelerated PageRank computation. In Proceedings of the International World Wide Web Conference, May 2005.

[26] Y. Minamide. Static approximation of dynamically generated Web pages. In Proceedings of the International World Wide Web Conference, May 2005.

[27] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, Stanford, CA, 1998.

[28] S. Pandey and C. Olston. User-centric Web crawling. In Proceedings of the International World Wide Web Conference, May 2005.

[29] M. Richardson and P. Domingos. The intelligent surfer: probabilistic combination of link and content information in PageRank. In Advances in Neural Information Processing Systems 14, pp. 1441-1448. Cambridge, MA: MIT Press, 2002.

[30] C. Sherman. Teoma vs. Google, Round 2. Available from World Wide Web (http://dc.internet.com/news/article.php/1002061), 2002.

[31] T. Upstill, N. Craswell, and D. Hawking. Predicting fame and fortune: PageRank or indegree?. In the Eighth Australasian Document Computing Symposium. 2003.

[32] T. Upstill, N. Craswell, and D. Hawking. Query-independent evidence in home page finding. In ACM Transactions on Information Systems. 2003.