An Overview of Ranking Algorithms for Search Engines

Ankur Gupta
PG Scholar
Delhi College of Engineering, Delhi
ankurgupta165@yahoo.com

Rajni Jindal
Faculty
Delhi College of Engineering, Delhi
rajnijindal@dee.ac.in

ABSTRACT

With billions of web pages available on the web, a user query entered in a search engine may return thousands of web pages, and thus, it becomes extremely important to rank these results in such a manner that the most “relevant” or “authoritative” pages are displayed first. This task of prioritizing the results is performed by ranking algorithms, and various search engines employ different schemes for ranking the results. This is also a dynamic field of research with different researchers proposing their algorithms corresponding to their interpretation of “relevance” or “authoritativeness” of a page to the user. The aim of this paper is to present an overview of various works done in this field. With the explosion of information on the Internet, a survey like this that presents and compares state-of-the-art ranking algorithms would not only help search engines to optimize their ranking schemes but also help the Internet content developers to code their information that facilitates searching.

Keywords
Search, ranking algorithms, PageRank, information retrieval, survey

1. Introduction

Ranking algorithms form an indispensable part of any search engine and a large amount of research has been done on them because they determine the quality of a search engine from the user’s perspective. Had the web been composed of a few hundred pages a manual ranking scheme could have been sufficient, as was the case during the early years of the WWW with the Yahoo! search engine [5]. However, with the explosion of information it could no longer be practical to rank millions of pages and automated means had to be developed in the form of ranking algorithms. The paper gives an overview of the various ranking algorithms that have been developed to enhance the search experience of the users over the World Wide Web. The paper is organized as follows, section 2 discusses the need for ranking algorithms, section 3 presents a survey of major ranking algorithms and section 4 concludes this paper.

2. Need for Ranking

There are billions of web pages on the web and it is more than likely that when a user enters a word to be searched for there will be thousands of web pages containing that word. It is obviously impractical for the user to visit all of these pages. Thus, one of the goals of a search engine is to provide the user with results that are most likely to be beneficial to him/her in least possible amount of response time. When the search engines return the result of a user query, only a predetermined number of documents are presented to the user. Thus, it is vital that the most relevant documents are included in the result and are prioritized in the display. This important task is performed by the ranking function. A ranking function that prioritizes the documents most relevant to a user will satisfy the user. It is this part of the search engine that this paper attempts to explore.

3. Overview of Major Ranking Algorithms

This section gives an overview of the major ranking algorithms.

3.1. PageRank

PageRank [3, 7] is based on the Random Surfer model and is the mainstay of the highly popular Google search engine. The Random Surfer model assumes that a user randomly keeps on clicking the links on a page and if s/he gets bored of a page then switches to another page randomly. Thus, a user under this model shows no bias towards any page or a link. PageRank (PR) is the probability of a page being visited by such a user under this model. PR is based on the link structure of the web, i.e. it analyses the in-links and out-links of the pages in the web and gives ranks to them accordingly. The algorithm assumes that if a page has a link to another page then it votes for that page. Therefore, each in-link to a page raises its importance under this concept and is captured by the PageRank algorithm. This idea is similar to the academic citations where a paper having higher number of citations or backlinks is considered to be important. PR is a recursive algorithm in which the PR of a page depends upon the PR of the pages linking to it. Thus, not only the number of in-links to a page influences its PR but also the PR of the pages linking to it. A page confers importance to the pages it references to by evenly distributing its PR value among all its out-links. Following is a simplified example of the PR algorithm,

\[
PR(A) = \frac{PR(B)}{4} + \frac{PR(C)}{1} + \frac{PR(D)}{2}
\]

where page A is being referenced by pages B, C and D; PR(X) is the PageRank of a web page X and the denominators signify the number of out-links present in the numerator, i.e. B, C and D respectively have 4, 1 and 2 out-links. As is clear from the above equation, B contributes 1/4th of its PR to A and would similarly distribute its PR among its 3 other out-links. The
fewer out-links a page has, higher will be the contribution of its PR value to those outgoing links. In the above example, the page C transfers its entire PageRank to its sole out-link.

Following is the actual PageRank algorithm proposed in [3],

\[ PR(A) = (1 - d) + d \sum_{i=1}^{n} \frac{PR(T_i)}{C(T_i)} \]

where d is the damping factor explained later; n is the number of in-links to page A; T_i is one of the n pages linking to page A and C(T_i) is the number of links out of page T_i, i.e., out-links of page T_i. For the sum of PageRanks of all web pages to be equal to one as stated in [3] the above equation needs to be re-written as follows,

\[ PR(A) = \frac{(1 - d)}{N} + d \sum_{i=1}^{n} \frac{PR(T_i)}{C(T_i)} \]

where N is the total number of web pages being considered. This makes PageRanks to form a probability distribution over all web pages being considered.

In the above equations, d is the damping factor that can be set between 0 and 1, and is usually set to 0.85 [3]. d denotes the probability of a user switching to another random page. As is evident from the above equations, even if a page doesn’t have any in-links it still has a minimum PR value of \( \frac{(1-d)}{N} \).

However, if a page doesn’t have any out-links, called a dangling link, it creates a problem because then its PR value cannot be distributed among other pages. Furthermore, there may be loops in the link structure that have only in-links and no out-links. Such cases need to be treated separately, described in [7]. One of the methods is to remove pages that don’t have any out-links from the PR calculations and to add them later on after other pages have attained their PR values. Another method is to distribute the PR values of such pages among all the other pages as if it were connected to all pages.

A page that is highly referenced will be highly ranked by the PR algorithm capturing the notion of it being important in the web community. Also, if a page has less number of in-links but they are highly ranked then this will also result in the page being given a high score. For instance, Yahoo! website is a highly popular website owing to its following and if any page gets a reference from the Yahoo! website then it naturally means that it is an important site and should be given a high PR. PageRanks of all pages are pre-computed offline which makes PR very fast and efficient. Moreover, it has low memory requirements as it is only dependent on the link structure of the Web. However, PageRank is not query dependent and may return results of popular pages that are not relevant to the query [12]. Also link structure is susceptible to manipulation which may not be detected by the PR algorithm. If a popular site references other sites for commercial purposes then these out-links will be ranked highly in PageRank even if they may not be relevant to the user query. Prevention against such spamming of search results is another active area of research.

### 3.2. Hubs and Authorities

The idea of topic distillation was put forth in [5] with the concept of ‘authorities’ and ‘hubs’. The algorithm addresses the ‘abundance problem’ where too many pages, all of which are not relevant to the query, are available for a broad search topic. It uses the link structure of the web to discover pages that can be considered to be the most ‘authoritative’ on a broad search topic. By authoritativeness of a Web page it means how relevant and important that page is for a topic in the WWW community. Under this algorithm a page is considered to be an authority on a topic if it is referenced by many pages relevant to that topic. Pages that link to many such related authorities are called as hubs. The idea is similar to PageRank but here the emphasis is not just to count the references from all pages in WWW but only from pages relevant to the topic. Still, the algorithm finds out relevant pages by using only the link structure of the web and avoids content analysis. This algorithm, also known as HITS (hypertext induced topic selection), works in following steps –

a. It initially uses the results returned by a text-based search engine for the query term as the root set.

b. It then expands this root set by including a predefined number of pages that link to this set and also those that are linked to by this set. The set thus formed, called a base set, is a focused subgraph of the WWW and would be relatively small, relevant and have many authorities.

c. Next, analysis is performed on this base set to compute hubs and authorities. Since the base set is a focused selection of pages, authorities may be identified by ordering the pages on the basis of their in-degree, i.e. number of pages linking to them. However, merely using the in-degree as a measure for authoritativeness doesn’t work because it fails to differentiate between ‘popular’ and ‘relevant’ pages. A page is relevant if it is related to the search topic, but popular pages will anyhow be highly referenced even if they were not related to the topic. Hubs help to resolve this situation because authoritative pages relevant to the topic, and hence the search query, would not only have high in-degrees but there would also be a sizeable overlap between the sets of pages that point to them. These sets of pages would be the hubs pointing to authorities on the same topic. Thus, there is a kind of coupling between good hubs and good authorities and they need to be identified independently. An iterative algorithm is used for this purpose as described below.

Each page p is associated with two nonnegative weights- one is the authority weight \( x_p \) and the other is the hub weight \( y_p \), such that,

\[ \sum_{p \in \text{base set}} (x_p)^2 = 1 \quad \text{and} \quad \sum_{p \in \text{base set}} (y_p)^2 = 1 \]

Where page p is part of the base set as calculated in above steps.

Due to the coupling between authorities and hubs, a good hub would be one that points to many good authorities and a good authority would be pointed to by many good hubs. Therefore,
the authority weight of a page should be the sum of the hub weights of pages linking to it, while the hub weight of a page should be the sum of the authority weights of the pages that it links to, i.e. –

\[ O: \quad y_p = \sum y_q \quad \text{where page } p \text{ links to page } q \]

These two operations are applied alternately and iteratively to determine hubs and authorities in the given set of pages and finally the results are ordered according to their weights.

Since HITS is entirely performed during query time it is slow and inefficient. Moreover, any pages that are left out from the initial root set cannot be part of the result even if they were relevant to the user query and thus, it is suited for popular queries.

### 3.3. Hilltop

Hilltop [2] also is an approach to authoritative ranking of popular queries similar to HITS but doesn’t use topic distillation. The aim of this algorithm is to identify documents that are ‘experts’, similar to the hub pages in HITS, on the query topic and use them as means to guide the search results.

An expert page is defined to be one that is based on a particular topic and links to many non-affiliated pages on that topic. Pages are non-affiliated if they are written by authors not belonging to the same organization which ensures an unbiased selection of expert pages. Hilltop is a two step process in which the first step deals with the identification of expert pages on the query topic and the second step uses these experts to compute the ranking of the target pages. In response to a query the most relevant expert pages on the query topic are selected. In the next step, authoritative pages on the query topic are determined by following the out-links of the expert pages selected in the previous step.

#### 3.3.1. Expert Score

The score of an expert is computed as a 3 tuple of the form \((S_0, S_1, S_2)\). \(S_i\) is computed by considering only key phrases that contain \(k-i\) of the query terms where \(k\) is the number of terms in the input query \(q\). Thus, \(S_0\) will be the score computed from the phrases containing all query terms. By key phrases we mean the phrases occurring at places considered important in a Web page such as title, headings and anchor text. The expert score is calculated depending upon the match between the query terms and the key phrases in a document and is given as,

\[ \text{Expert Score}(E) = 2^{32} \times S_0 + 2^{16} \times S_1 + S_2. \]

#### 3.3.2. Target Score

Targets are those pages that are pointed to by at least two non-affiliated experts and are themselves not affiliated to them. A predefined number of most relevant expert pages are used for the calculation of target scores.

[2] gives rules when a phrase qualifies a particular edge \((E,T)\) from an expert page \(E\) to a target page \(T\). If \(ocw(T)\) is the number of distinct key phrases in \(E\) that contain a query keyword \(w\) and qualify the edge \((E,T)\), then,

\[ \text{Edge Score}(E,T) = \text{Expert Score}(E) \times \sum \text{occ}(w,T) \]

If for any query keyword \(ocw(T) = 0\) then \(\text{Edge Score}(E,T) = 0\) because Hilltop considers only those edges that contains all query terms in the key phrases that qualify it. Finally, the Target Score of each target is calculated as the sum of edge scores of all edges incident on it and the list of targets are ranked by their Target Score.

\[ \text{Target Score}(T) = \sum_\text{edges incident on } T \text{Edge Score}(E,T) \]

Hilltop, like HITS, is also tuned for popular queries. It would not return results for queries for which no experts could be identified. This algorithm also performs content analysis during query time because of which it is slow and has high memory requirements.

### 3.4. Query Sensitive Page Ranking

In [10] the idea of identifying result pages relevant to the query is further explored. It defines a global importance scope and a local importance scope for a Web page. By global importance of a page it means the authoritativeness of the page, as is determined by algorithms like PageRank and HITS, while the local importance of a page relates to its relevance to the query, i.e. how specific it is to the search term. It proposes an algorithm that is sensitive to the query by combining both global as well as local importance of the Web pages during ranking. It incorporates global importance scope by using the search results of a search engine for the query term and calls it the voting set. For any two pages \(p\) and \(q\) in this set, page \(p\) is said to vote for page \(q\) if any of the following conditions hold–

1. \(p\) has a link to \(q\)
2. \(p\) has key phrases of \(q\), like its title or headings in its content
3. \(p\) explicitly mentions that its information is referred from page \(q\)

Care is taken to cancel votes between affiliated pages to remove any bias. The number of votes a page gets is a measure of its query sensitiveness. The final ranks to the pages are assigned as a combination of their query sensitiveness and global importance values like PageRank or on the basis of their query sensitiveness alone. This algorithm works on the results returned by a traditional search engine or meta-search engine and thus has low memory requirements. Since it works upon few of the returned results it can work relatively fast. However, its dependence upon other search engines limits its efficiency.

### 3.5. QueryFind, Combining User’s Feedback and Expert’s Agreement

In QueryFind [11] the relevance of a Web page with a query word is determined by using a combination of user’s feedback in the form of page clicks and the recommendation of a content-oriented source search engine. A querying relation set is constructed with the specific query and the web pages that were clicked with respect to that query. Here the assumption is that the higher the number of clicks to a Web page in a set, the
more important it is to the user searching that query. The information regarding Web page usage such as time spent on a page and number of clicks to a page is maintained in the query logs of search engines and can thus be utilized. The algorithm uses the results from a content-oriented source search engine, YAM, in addition to user’s feedback. The results returned by the content-oriented search engine are then reordered based upon the number of user clicks on those pages, thus, ensuring the relevance of pages to the search query. The algorithm uses two scores – user’s feedback ranking score and content-oriented ranking score and combines them to generate the final scores as follows:

\[ F_i = \frac{C_i}{\sum_{j=1}^{n} C_j} \quad \text{and} \quad B_i = \frac{Q_i}{\max_{i=1}^{n} Q_i} \]

where \( C_i \) is the clicked times of the URL \( i \); \( n \) is the total number of different URLs, and \( F_i \) is the normalized user’s feedback ranking score lying between 0 and 1. \( B_i \) is the normalized content based ranking score of URL \( i \) and \( Q_i \) is the content-oriented ranking score of URL \( i \).

Both the above scores are combined to give the final ranking scores \( S_i \) to the URLs as, \( S_i = F_i \times B_i^{1/2} \).

The algorithm makes use of query logs to make search results sensitive and hence has high memory requirements. It also makes use of a search engine for the initial set of results, thus, limiting its efficiency.

### 3.6. Personalized Web Page Ranking based on Trust and Similarity

In global ranking schemes like PageRank and HITS, the page rankings that are computed are same for any user. However, in personalized page ranking the ranking of a page is specific to every user and it may be possible that the same page is ranked at the top for some user A but ranked lowly for another user B if their tastes differ. The concepts of trust and similarity are used in [9] to target specific user preferences and compute personalized page rankings. By analyzing implicit and explicit user feedback each user’s trust and similarity values are computed and propagated within each user’s neighbourhood. The users are required to create their profiles indicating other users whom they trust and thus, form a network of trusted friends. Other information like their area of interest and profession is also stored which is used for calculating similarity between trusted peers. Trust values between two users decreases with every hop and is not propagated after a certain distance. The similarity values are calculated on the basis of the match between users’ profiles, pages visited and page ratings. Combining the trust and similarity values a local metric value is calculated between every user and his trusted peer as follows:

\[ LM(X,Y) = a \times TV(X,Y) + b \times Sim(X,Y) \]

where \( LM(X,Y) \) is the local metric assigned to user \( X \) by user \( Y \); \( TV(X,Y) \) and \( Sim(X,Y) \) are the trust value and similarity value respectively, between users \( X \) and \( Y \).

Those users whose LM values are above a certain threshold will contribute to the personalized recommendations. Personalized page rank, \( LocalRating(P) \) of a page \( P \) for each user is calculated using a combination of explicit and implicit rankings given to a page as follows –

\[ LocalRating(P) = \alpha \times ExplicitRating(P) + \beta \times ImplicitRating(P) \]

where ExplicitRating is calculated using the local metric values and ImplicitRating is gathered using information like time spent on the page, the number of visits etc. This algorithm requires users to set up their profiles for the personalization to take effect and will not be successful for non-registered users. It also depends upon the networking of trusted peers and its performance would be low during the initial stages due to lack of enough information on which to personalize the results.

### 3.7. PageRank Modifications

The following algorithms have used the basic PageRank algorithm and modified or fine tuned it for a specific purpose.

#### 3.7.1. Intelligent Surfer Model

In [8] a query-dependent, content-sensitive version of PageRank is proposed and is based on a Directed Surfer model. As opposed to the Random Surfer model such a model assumes that the surfer would have preference of certain kinds of links or pages than others, such as those links appearing at the top of a page or pages similar in content to the query term. In [8] the behaviour assumed is that the surfer would move next to a page that is similar in content to the search query than to any random page or link and following is the modified PageRank algorithm–

\[ P_q(j) = (1 - d) \hat{P}_q(j) + d \sum_{i=1}^{n} P_q(i) P_{q(i,j)} \]

Where \( P_q(j) \) is the query-dependent PageRank score of page \( j \) when the query is \( q \) and \( i \) is the set of pages incident on page \( j \). \( P_q(i,j) \) is the probability of the surfer moving to page \( j \) given that s/he is on page \( i \) and is searching for query \( q \). \( \hat{P}_q(j) \) specifies where the surfer would move if s/he gets bored of the current page and was no longer following the links on that page. If \( R_q(j) \) is defined as a measure of relevance of page \( j \) to query \( q \) then the probability distributions \( P_q(j) \) and \( P_q(i,j) \) are defined as –

\[ P_q(j) = \frac{R_q(j)}{\sum_{k \in W} R_q(k)} \quad \text{and} \quad P_q(i,j) = \frac{R_q(j)}{\sum_{k \in F_i} R_q(k)} \]

Where \( W \) is the set of all Web pages being considered and \( F \) is the set of pages that are referenced by \( i \), i.e. set of out-links on page \( i \).

It is evident from the above equations that \( \hat{P}_q(j) \) and \( P_q(i,j) \) provide the biasing element to this query-dependent PageRank (QD-PR) and if \( R_q(j) \) is reduced to a constant value 1 then this QD-PR would reduce to the random surfer PageRank.
Intelligent Surfer involves real-time computations to calculate the similarity of the query to the Web pages and is hence slow. However, its results are sensitive to the user query.

3.7.2. Biased Surfer Model

Very similar to [8] is the BiasRank proposed by [4] that is based on a Biased Surfer model. While in Intelligent Surfer the user is assumed to move to another page that is most closely related to the search query, in Biased Surfer model the user is assumed to move towards that page whose content is most similar to the current page that the user is on. Thus, Intelligent Surfer matches the content of pages with the user query while Biased Surfer matches content of pages with other pages. This difference allows the calculations in Biased Surfer to be performed all during off-line as the algorithm is independent of the query term. BiasRank is calculated as-

$$BR(A) = (1 - d) + d \sum_{i=1}^{\infty} [FN(U_{i\rightarrow A}) \times BR(U_i)]$$

where BR(A) is the BiasRank of page A and \(FN(U_{i\rightarrow A})\) is the measure of relevance of page i to page A. Thus, the more a page 'fancies' another page, the more will it contribute its ranking towards it. This is different from PageRank where a page distributes its ranking equally among all its out-links. In [4] the notion of fanciness of one page for another is computed on the frequency of common words between the two pages and is defined as–

$$FN(U_{x\rightarrow z}) = Q \times C_{x,z} + 1$$

where \(Q\) is a factor \(\geq 0\), x is a page that has t out-links and \(C_{x,z}\) is the cosine similarity of the keyword frequency vectors of pages x and z and could be replaced by any other content-similarity measuring function. The algorithm based on this value of FN is a subset of BiasRank and is called WordRank. If in the above equation the factor Q is set to zero then the BiasRank equation reduces to PageRank. This algorithm gives fast results as it doesn’t have query time computations, however, it has high memory requirements due to content similarity information between document pairs.

3.7.3. Topic-Sensitive PageRank

Under the Topic-Sensitive PageRank proposed in [4], the Web is divided under a number of topics taken from the Open Directory Project (ODP) [13]. While in the traditional PageRank algorithm each page receives a single PR value, under Topic-Sensitive PR, each page will have as many PR values as the number of chosen topics. In [4] the Web is categorized under sixteen topics and thus, 16 PageRank vectors are calculated for the entire Web, all off-line. At query time the topics on which the query is based are identified and for each of the matching documents containing the query term a weighted combination of the various PR values related to the topic is calculated to form the Topic-Sensitive PageRank.

$$TP_d = \sum_j \text{Sim}(q, j) \times \text{Rank}(d, j)$$

Where \(TP_d\) is the Topic-Sensitive PR value of URL d, \(\text{Sim}(q, j)\) is the similarity of the query with topic j and \(\text{Rank}(d, j)\) is the importance score of document d with respect to topic j. The algorithm also calculates context-sensitive PageRanks by taking into consideration the context of the query. For instance, if the query term was clicked on a page instead of being directly entered then the document where the query was clicked is matched with the defined topics. Thus, for context sensitive queries, the only thing that needs to be changed in the above equation is to replace the term \(\text{Sim}(q, j)\) by \(\text{Sim}(qdoc, j)\) where qdoc is the document from where the query term was clicked.

This algorithm has both off-line and query time computations. Also the memory requirements are higher than PageRank. The effectiveness of the algorithm depends on the categorization of the topics according to the ODP editors and is thus, subjective to their interpretations.

3.7.4. Personalizing PageRank based on Domain Profiles

In [1] users are required to select features from the Internet Domain Name System in their profiles and then PageRank vectors specific to the user profiles are calculated. This algorithm has similar motivations to [4] but doesn’t require any content analysis as the personalization information is based entirely on the URLs. If there are ‘n’ binary features from which the users are required to create their profiles then there will be \(2^n\) PageRank vectors, i.e. each URL will have \(2^n\) PR values based upon \(2^n\) different combinations of the binary features. Personalized PageRank computation is done as follows–

$$R_U(A) = (1 - d) + d \sum_{i=1}^{n} \frac{W_U(T_i) \times R_U(T_i)}{C(T_i)}$$

Where all variables have same meanings as in traditional PageRank except - \(R_A\) which is the personalized PageRank of URL A based upon the user profile U, \(W_U(T_i)\) is the weight of page \(T_i\) based on profile U and is the biasing factor of this algorithm. If \(W_U\) is set to 1 then this algorithm reduces to traditional PageRank.

\(W_U(A) = 2^{-N}\) where N is the number of available domain features, and n is the number of matched features with the user profile U and the domain features of the URL A. For example, if the user profile has selected .com (commercial) and .in (India) as domain features then a URL like www.google.com would match one of the two features (.com) and the value of \(W_U(p) = 0.5\) where p is the google URL.

4. Conclusion

This paper has provided an overview and critique of the major search engine ranking algorithms. In terms of speed and
memory requirements PageRank remains the most efficient. Although PageRank is query independent, the scale at which it operates sustains the relevance of the search results to the query terms. However, pure link analysis cannot combat spam and PageRank is susceptible to it. Content similarity may, to some extent, be able to filter out spam pages but such algorithms have higher memory requirements. The trend is towards personalized search as the ultimate aim of any search engine is to satisfy the needs of the user and since users have distinct preferences each user should be treated differently. However, personalized ranking algorithms would require more space and tend to be slower due to query time computations. Thus, there is a tradeoff between search engine efficiency and the need to make results more sensitive to query terms and to the user preferences. Different algorithms have different applications depending upon their target users and no one algorithm is ideal. To provide a better search experience the search engines should combine the encouraging aspects of various ranking algorithms while maintaining their efficiency.

5. Future Scope
Search engine ranking algorithms is a very active field of research. As and when new algorithms are proposed they will be required to be compared with the existing algorithms to determine their efficiency. Combating spam and commercial linkage to sites is another important criterion on which to judge the effectiveness of various algorithms and needs to be dealt in detail in the future.

6. References