Comparative Analysis of Web Search and Ranking Algorithms

Mihir Kelkar

University of Maryland, Baltimore County

CMSC 676 - Information Retrieval

Email: mihir2@umbc.edu

***Abstract*—Due to the rapid expansion of the world wide web, people's reliance on obtaining information from the internet via search engines is increasing exponentially. In such a situation where retrieval of relevant data (on time) is important; the search and ranking engine involved in the process is of pivotal importance. This term paper deals with the comparison and analysis of various web page ranking algorithms and presents their advantages, disadvantages, peculiarities and special use cases for the purpose of ranking web pages based on several factors. This analysis aims at finding their relative strengths and limitations.**

Keywords*—Data Retrieval, Web Page Ranking, Google page rank Algorithm, HITS Algorithm.

I. Introduction

Given the increasing amount of data generated and added to the internet every second, presenting relevant data to an internet user's information need is a challenge. Relevance of data is extremely subjective. Relevance of retrieved information for the same query may change as per several factors which includes the user's age, geographic location, the device the user query is sent from, the time elapsed since query, time of the day query was sent and lastly the maybe even the user's interpretation of the data. In order to provide the best results of relevance web search engines need to fine tune their results by considering many if not all of these factors. Needless to say, providing relevant information is a complicated process. In fact, search engines are the only websites in the world which aim at reducing the time users spend on their website by providing relevant information in as few query modifications as possible. Ranking algorithms are of great importance to this process since an efficient search and ranking of query words can lead to a faster retrieval of data for an user. Furthermore, ranking of web documents comes with its own set of challenges; for eg. some web documents are meant for the sole purpose of navigating to a different document, some documents are not self descriptive, some are random strings of 0 and 1s which are non human readable.

The motive behind this term paper is to analyze the currently important algorithms for ranking and retrieving web pages and to present their relative advantages and limitations.

A. Web Mining: A Background

Web mining is the process of classification of the web pages and the internet users based on page contents and user behaviour in the past. Web mining can be further classified into **WCM** : Web Content Mining , **WSM** : Web Structure Mining and **WUM** : Web Usage Mining. As the name suggest ; **WCM** leverages web page content for extraction, exploration and crawling purposes. **WUM** pertains to creating log files about user behaviour and search trends to generate log files which are cross referenced for retrieval and ranking optimizations for that specific user's queries. The concept of **WUM** can be further extended from an user to a bunch of users. Eg. rank a particular result higher for University of Maryland's IP address. The WSM model focuses on studying the structure of the relationship between two web pages connected by a hyper-link. This involves studying the number of links that go out from a page and the number of link that lead to the given web page. The WSM model relies on gauging the importance of a document by assuming that an important document will have several documents which lead to it, making it a focal point for information. This model happens to be the basis of the most successful web document ranking systems i.e Google's page rank algorithm developed by Larry Page and Sergey Brin.

II. Page Rank Algorithm

A. History of Page Rank

Named after Google's co-founder Larry Page, PageRank is Google Search's most important ranking algorithm. Developed at Stanford University in California, PageRank was also the first ranking algorithm which was used by Google. As previously stated, PageRank is based on counting and weighting the incoming and outgoing links from a webpage with the underlying assumption that every important page is cited well in other documents. As a matter of fact, PageRank was inspired by the way scholarly papers which are cited often are considered important. A variant of Page Rank is also used in Mainland China's most successful search engine ”Baidu”.

B. An Overview of the PageRank Algorithm

Page rank is a link analysis algorithm which assigns a weight to each incoming and outgoing links in a interconnected corpus as a connected graph. Web pages are the nodes in this graph with hyperlinks which forms the edges between them. Reciprocal links between two web pages is corresponding to a bi-directional edge in a graph. However some nodes in this graph can be identified as ”reference” or ”authority” nodes which have a certain guarantee of authenticity (eg. data listed on the White House's press release website). The distance or connection to such authority hubs along with a node's degree of connectivity in the graph lead to a numerical index for every node in the graph. This numerical index for that particular node is actually its *PageRank*.

C. Explanation of the PageRank Algorithm

For an efficient explanation of the way page rank works, we have to keep in mind that page rank assigns scores to individual webpages and not to all webpages under the same web domain as a single entity. Thus, multiple pages of the same website may have different (albeit slightly) Pageranks. Lets initially assume that the entire world wide web is restricted to 4 web pages connected to each other via hyper links.

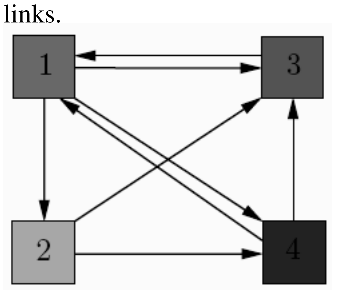


Fig. 1. Four interconnected web documents.

Whenever one web page references another webpage in the graph , we add a link that page to the other. Eg. page 1 here links to all other pages in Fig. 1, hence we add links to all other pages from page 1. Every page transfers a section of its importance to all pages it links too. As a general assumption for simplicity, a link to k pages leads to a distribution of *1 / k* on all links. Node 1 has 3 outgoing links, so it will pass on 1/3 of its importance to its three outgoing links. Node 3 has one outgoing link, all of Node 3's importance is therefore passed onto Node 1.

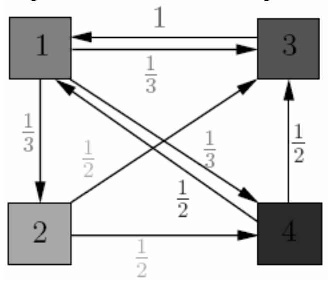


Fig. 2. Importance distribution among webpages

If represented in the form of a matrix, the transmission matrix for the above 4 webpages would look as follows:

$$\text{Importance Matrix} = \begin{bmatrix} 0 & 0 & 1 & 0.5 \\ 0.33 & 0 & 0 & 0 \\ 0.33 & 0.5 & 0 & 0.5 \\ 0.33 & 0.5 & 0 & 0 \end{bmatrix}$$

Suppose that initially the importance is uniformly distributed among the 4 nodes, each getting 0.25. Denote by v the initial rank vector, having all entries equal to 0.25. Each incoming link increases the importance of a web page, so at step 1, we update the rank of each page by adding to the current value the importance of the incoming links. This is the same as multiplying the matrix A with v . At step 1, the new importance vector is v1 = Av. We can iterate the process, thus at step 2, the updated importance vector is v2 = A(Av) = A^2v. Numeric computations give:

$$v = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}, \ Av = \begin{pmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{pmatrix}, \ A^2 v = A\,(Av) = A \begin{pmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{pmatrix} = \begin{pmatrix} 0.43 \\ 0.12 \\ 0.27 \\ 0.16 \end{pmatrix}$$

$$A^3 v = \begin{pmatrix} 0.35 \\ 0.14 \\ 0.29 \\ 0.20 \end{pmatrix}, \ A^4 v = \begin{pmatrix} 0.39 \\ 0.11 \\ 0.29 \\ 0.19 \end{pmatrix}, \ A^5 v = \begin{pmatrix} 0.39 \\ 0.13 \\ 0.28 \\ 0.19 \end{pmatrix}$$

$$A^6 v = \begin{pmatrix} 0.38 \\ 0.13 \\ 0.29 \\ 0.19 \end{pmatrix}, \ A^7 v = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}, \ A^8 v = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}$$

From a probabilistic perspective, we can consider the popularity or importance of a given webpage as that of a random surfer on the internet starts following hyper-links and ends up at the webpage. In our model, a random surfer currently viewing webpage 2 has 0.5 probability that he will move on to either webpage 3 or webpage 4. We can model the probabilistic process as a random walk on a graph with the initial probability being equally distributed as a function of the number of documents present. Each page has equal probability 0.25 to be chosen as a starting point. So, the initial probability distribution is given by the column vector [0.25, 0.25, 0.25, 0.25]. The probability that page i will be visited after one step is equal to Ax, and so on. The probability that page i will be visited after k steps is equal to A^kx. The sequence Ax, A^2x, A^3x, ..., A^kx, ... converges in this case to a unique probabilistic vector v*. In this context v* is called the stationary distribution and it will be our Page Rank vector.

Even if the PageRank vector is calculated from any of the above listed methods, it would point out that the first page is the most relevant page in the aforementioned documents.

D. Challenges in the PageRank Algorithm

One of the challenges that the classic pageRank algorithm has to deal with is that of dealing with dangling nodes(nodes with no outgoing edges) and components in the graph which might be disconnected from the rest of the graph.

In case of dangling nodes, an iterative computation of the page rank vector would eventually give us a PageRank vector with all elements as zeroes which is counter-intuitive.In the cases of disconnected components, the PageRank vector would never be able to compute the values for the part of the graph which a random surfer cannot reach.

E. *Proposed Solution to the limitations of PageRank*

The PageRank algorithm is based on the underlying assumption that a page's importance is infact an aggregation of the pages that it connects to and is connected to. In order to counter the challenges of dangling nodes and disconnected graphs, the algorithm uses the connect of a **damping factor**.The PageRank theory assumes an imaginary random surfer who is clicking on links will eventually stop clicking. The probability, at any step, that the person will continue is a damping factor d. The damping factor is used in conjunction with the PageRanks of incoming links to a page.

The PageRank vector for a page is now derived as follows : M = (1 - d) * A + d * B where A is the matrix which shows how much importance each webpage "shared" to its connections, B is an identity matrix and d is the damping factor. The matrix M models the random surfer model as follows: most of the time, a surfer will follow links from a page: from a page i the surfer will follow the outgoing links and move on to one of the neighbors of i. A smaller, but positive percentage of the time, the surfer will dump the current page and choose arbitrarily a different page from the web and "teleport" there. The damping factor "d" reflects the probability that the surfer quits the current page and "teleports" to a new one. Since he/she can teleport to any web page, each page has probability to be chosen. This justifies the structure of the matrix B.The value of the damping factor is adjusted by Google everytime the Googlebot adds a substantial amount of crawled data to its index. The usual value is somewhere between 0.15 and 0.85

III. HITS Algorithm

A. *Introduction and History of HITS Algorithm*

Hyperlink Induced Topic Search (also commonly known as the Hubs and Authorities Algorithm) is a link-analysis algorithm that served as the primary technology for web search and ranking before the advent of the PageRank algorithm. The essence of the HITS Algorithm is derived from the earlier development of the Internet in which certain pages served as "hubs" of information which weren't authoritative themselves but instead linked to several authoritative pages over a vast spectrum of information needs. The Algorithm seeks to identify such "hubs" and "authority" pages. A good hub is a page which is connected to several authority pages whereas a good authority page was a page linked to several hubs. Scores in this scheme are determined by a page's rating on the hub index and the authority index. Twitter uses the HITS Algorithm to push suggestions.

B. *Overview of the HITS Algorithm*

The algorithm presumes that a good hub is a document that points to many others, and a good authority is a document that many documents point to. Hubs and authorities exhibit a mutually reinforcing relationship: a better hub points to many good authorities, and a better authority is pointed to by many good hubs. To run the algorithm, we need to collect a base set, including a root set and its neighborhood, the in- and out-links of a document in the root set.The root set can be obtained by taking the top n pages returned by a text-based search algorithm.A base set is generated by augmenting the root set with all the web pages that are linked from it and some of the pages that link to it. The base set and all the hyper-links from these pages cumulatively form a graph. The entire HITS Computation is performed only on this graph.

C. Explanation of the HITS Algorithm

The Authority and hub values of a web apge are defined in terms of a mutual recursion. An authority value is computed as the sum of the scaled hub values that point to that page.A hub value is the sum of the scaled authority values of the pages it points to. The algorithm performs a series of iterations, each consisting of two basic steps:

Authority Value Update:The Authority score for a node is updated to be equal to the sum of all the hub scores that point to it. A higher authority score is maintained by being linked to pages which are identified as hubs.

$$\mathrm{auth}(p) = \sum_{i=1}^{n} \mathrm{hub}(i)$$

Hub Value Update: Update each node's Hub Score to be equal to the sum of the Authority Scores of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.

$$\mathrm{hub}(p) = \sum_{i=1}^{n} \mathrm{auth}(i)$$

The Algorithm used to calculate the Authority and Hub values for a given web page is as follows:

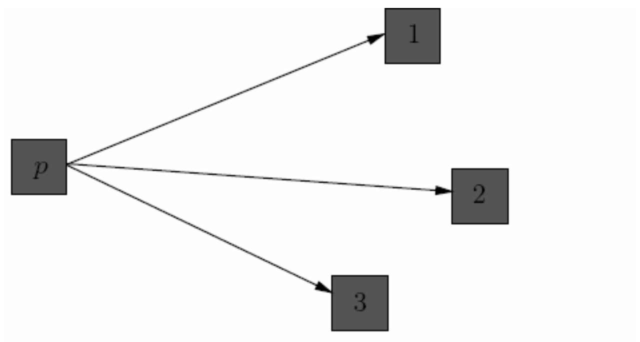
1)*Assign an authority and hub score of one to each web page*

2)*Update Authority Value*

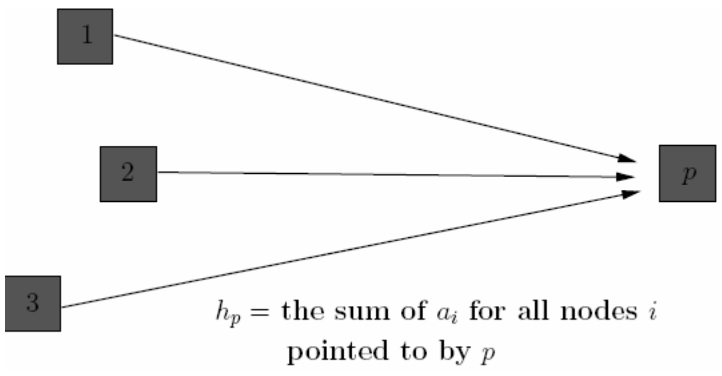
3)*Update Hub Value*

4)*Normalize the values of Hub scores and Authority scores. Normalization is done by dividing each value with the square root of the sum of squares of all authority / hub scores*

5)*Repeat Step Two if necessary*



a_p = the sum of h_i for all nodes i pointing to p


1
2
3
p
h_p = the sum of a_i for all nodes i pointed to by p

Let A be the adjacency matrix of the graph, the authority weight vector is denoted by v and the hub weight vector by u.

$$v = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \text{ and } u = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}$$

Updating the Authority and Hub values eventually translate to :

v = A^t * u u = A * v

After about k iterations, this would translate to the following:

v_k = (A^t * A) v_{k-1}

u_k = (A * A^t) u_{k-1}

So, the values for both vectors are eventually expected to converge to a ”solid” value

D. Limitations of the HITS Algorithm

In order to get a set rich in both hubs and authorities for a query Q, we first collect the top documents that contain the highest number of occurrences of the search phrase Q. These, may not be of tremendous practical relevance. The pages from this set called root (RQ) are essentially very heterogeneous and in general contain only a few (if any) links to each other. So the web subgraph determined by these nodes is almost totally disconnected. Authorities for the query Q are not extremely likely to be in the root set RQ. However, they are likely to be pointed out by at least one page in RQ. So it makes sense to extend the subgraph RQ by including all edges coming from or pointing to nodes from RQ. We denote by SQ the resulting subgraph and call it the seed of our search. Notice that SQ we have constructed is a reasonably small graph as compared to the entire internet. The essence of the algorithm lies in the fact that the algorithm eventually converges on a score to give us a result. However, larger the graph, the more time it will take to converge thus making the algorithm impractical for certain purposes.

Some other problems that can occur with the HITS approach are as follows:
1)High rank value is given to some popular website that is not highly relevant to the given query.
2) Drift of the topic occurs when the hub has multiple topics as equivalent weights are given to all of the outlinks of a hub page.

E. Improvements to the HITS Algorithm

Implementing a weighted version of the HITS algorithm is one of the proposed solutions to overcome the limitations of the classical HITS Algorithm. A general overview of the suggestion is as follows:
1)Before starting a HITS-based algorithm, if there exists a root link whose in-degree is among the three smallest ones and whose out-degree is among the three largest ones, then set its weight to 4 for in-links of all the root links. 2)Otherwise, set all to 1. Run the HITS-based algorithm for one iteration without normalization. If there exists a root link whose authority value is among the three smallest ones and whose hub value is among the three largest ones, set weight to 4 for in-links of all the root links.

In the above two steps, usually the in-degree of a small-in-large-out link is as small as 0, 1, or 2, while the out-degree can be more than several hundred. Intuitively, in most cases, it is hard to believe that a root link with no or few in-links can point to many highly relevant documents. Even if it points to many good documents, due to the large number of documents in the base set, there may be some duplicates between the out-links of the small-in-large-out link and the neighborhood of other links, and these good duplicated documents still have the chance to top the hub set or the authority set. The method of setting in-link weights are very simple and can be further improved by adaptively changing the weights of both in- and out-links of a small-in-large-out link.

IV. The Tag Rank Algorithm

A. Introduction to the TagRank Algorithm

The TagRank algorithm calculates the heat of the tags by using time factor of the new data source tag and the annotations behavior of the web users. This algorithm provides a better method for ranking the web pages especially in the context of social media. Since much of the data generated nowadays is either through some kind of social media or a crowd funded web app, this algorithm’s importance is on the rise.

B. Overview of TagRank Algorithm

Some amount of data annotation exists in all kinds of social networks and crowd sourced applications. Although this data enriches our already extensive collection, retrieving highly relevant data fast becomes a problem. On the socially annotated part of the web , common web search algorithms aren't very efficient in indexing either.

Naive Approach on indexing social annotated web content Pages maybe ranked by the time of being annotated. As an example if n webapges are annotated as "videos", most recent videos will be retrieved every time a user queries for videos.

Co-occurence Approach on indexing social annotated web content The idea of Co-occurrence Approach is that for a bookmark, all tags annotated to it include the related tag: we find one tag, we always find another one. (For example, web and design always occur at the same time.) This means there is some relation between these two tags. The co-occurrence of them better reflects the content of web resource which the bookmark mapped, so the reliability is better.

This is somewhat similar to term term co-relations in text corpuses.

C. Explanation of the TagRank Algorithm

A general assumption in social anootation web is that the more bookmarks are annotated, the more self-identity the corresponding webpage will get.For bookmarks that often refer to the similar tag, the bookmark which is created more times recently is given a higher significance than the one which has been created less times recently.

1) For each tag, choose a respective bookmark.Rank each of them in a falling sequence.Each bookmarks is assigned an initial value called TagRank.

$$\mathrm{Num} \rightarrow \mathrm{TR}_{\mathrm{bm}} \quad \mathrm{TR} \in [1, 100]$$

2)We assume that all bookmarks are ranked not only according to the times of being annotated, but also take into account the total annotation times and recent annotation times to give the value Formula of the bookmarks corresponding to a tag.

$$\mathrm{TR}_{\mathrm{bm}} = \rho_i^{*}\mathrm{TR}_{\mathrm{bm}}$$

We can now assign every bookmark a proper TagRank value according to its total times of being annotated and the times of being annotated recently. In this way we can know users' attitude to these bookmarks. Due to the peculiar nature of the social annotation web, the web pages which are annotated by users are simply much more in number than newer ones generated. It's equal for those tags and bookmarks which occur recently. Though the heat of these tags is very high, because of their low initial TR value, their ranking is very low, and even can't be indexed.

3)We divided the time between the initial annotation and the most recent annotation into time slices where each time slice represents a unit month.So every two consecutive time slices represent the popularity of the tag during this time period.

$$\rho_1 = (\mathrm{Num}_{\,\mathrm{i+1}} - \mathrm{Num}_{\,\mathrm{i}})/\mathrm{Num}_{\,\mathrm{i}} \quad \mathrm{i} \in [1, \mathrm{n\text{-}1}]$$

4) Analyze the heat of all tags from position i to position n - 1. Select the maximum heat for the particular tag.

$$\rho = \max \rho_{\mathrm{i}}$$

5)Finally according to the formula: TagRank, we gain the TR value based on heat, and acquire a new ranking of bookmarks:

$$\mathrm{TR} = \mathrm{TR}\,^{*}\,(1 + \rho)$$

The entire Tag Rank Algorithm can be expressed lucidly in pseudo-code as follows:

```
TagRank()
    Begin
    Extract tag from webpages
    pretreeat and process the tag
    for all tags in bookmarks
    do
        calculate the tag's anootation —
        time and set an initial value
        Calculate the tag's heat in the —
        time slice
        Caluclate the Tag's final heat factor
    Calculate the value of TagRank
    End
```

V. Edge Rank Algorithm

A. History and Introduction to Edge Rank

EdgeRank is the Facebook algorithm that decides which stories appear in each user's newsfeed. The algorithm hides boring stories, so if your story doesn't score well, no one will see it. This algorithm is of extreme importance since it possibly dictates your social interaction with the rest of your friends affects the way information about about you is retrieved by your social circle. The first thing someone sees when they log into Facebook is a feature called newsfeed. This is a summary of what's been happening recently among their friends / followers on Facebook. However, not every story that gets published from your social connections shows up on your newsfeed. Edge rank is an an algorithm which retrieves data which is relevant to you by studying the user's social behaviour on the websites. Edge rank takes into consideration the posts you view, the pages you like, the user profiles you visit, the number of times you visit the user profile of a specific user, the amount of private messages you use with other users and then creates a specific summary of relevant data tailored to your interest.

B. Explanation of the Edge Rank Algorithm

An EdgeRank score is calculated by considering three important parameters;
1)Affinity Score
2)Edge Weight
3)Time Decay

Affinity Score: Affinity score is the metric that is used to estimate the level of interaction between two nodes on the entire social graph. Lets consider that a user X and his brother user Y are the two nodes currently under consideration. Then the affinity score of User X for User Y is computed by taking into account the number of mutual connections, number of messages exchanged, profile views amongst both users and an "acknowldged relationship" between both nodes. An acknowledged relationship points to the fact that both nodes agree that they are closely related by either indicating themselves as a relative or a spouse or even close friends on their user profiles.Facebook calculates affinity score by looking at explicit actions that users take, and factoring in 1) the strength of the action, 2) how close the person who took the action was to the user under consideration, and 3) how long ago they took the action.Explicit actions include clicking, liking, commenting, tagging, sharing, and friending. Each of these interactions has a different weight that reflects the effort required for the action–more effort from the user demonstrates more interest in the content. Commenting on something is worth more than merely liking it, which is worth more than merely clicking on it. Passively viewing a status update in your newsfeed does not count toward affinity score unless you interact with it.Affinity score measures not only a certain user's actions, but also his/ her friends' actions, and their friends' actions. For example, if user X commented on a fan page, it assigns somewhat more of a weight to the affinity score of the fanpage if a friend of user X did the same thing or a friend of a friend of user X did. Not all friends' actions are treated equally. If User X clicks on someone's status updates and write on their wall regularly, that person's actions influence X's affinity score significantly more than another friend who X tends to ignore. Lastly if X used to interact with someone a lot and then stops over time, affinity score for that person begins to wane

Edge Weight Facebook classifies interactions as users as edges in the social graph. Thus each interaction has a differnet amount of emphasis attached to it, in mathematical terms they have different edge weights.Every action that a user takes creates an edge, and each of those edges, except for clicks, creates a potential story. By default, you are more likely to see a story in your newsfeed about User X commenting on a fan page than a story about X liking a fan page because a commenting activity has a higher weight than a liking activity. Facebook changes the edge weights to reflect which type of stories they think user will find most engaging. For example, photos and videos have a higher weight than links. Quite predictably this could be adjusted on a per-user level–If user X has an inclination for paying more attention to posts with photos attached in them, the weights of photo posts for X will be much higher than any other posts. Even the process of taking action might influence the edge weights. Lets say User X searches for a particular page and likes it, it has a much higher weight than User X liking a sponsored advertised page. Also, newly rolled out features on the websites have extremely high edge weights so that more users start noticing them and ause them. For eg. When Location tagging was introduced, its weight was much higher than most interactions.

Time Decay The very essence of communication in a social structure is that it needs to be synchronous. Trends change quickly and discussion topics migrate. In order to maintain relevance of displayed stories, they have to be migrated as old and rolled out of active summaries. Thus, every time you log-in to your account; your time stamp is used to make a decesion as to whether a story would be of relevance to you now. If you log in more, your time decay will be much higher; which means that your newsfeed actually displays stories which are changing , very few stories from before the last time you logged in are repeated. However, if you log-in once a week, your time decay si very low and older stories will appear in your summary.

The mathematical representation of Edge rank the news feed could be as follows:

EdgeRank = $\sum_{i=1}^{n} U_e \mathrm{W}_e \mathrm{TD}_e$

VI. Reddit's Story Ranking Algorithm

A. Introduction to Reddit and its story Ranking Algorithm

Reddit is one of the fastest growing entertainment and social websites where registered users can share content which includes images, text or multimedia. Reddit is one of the few websites on the internet which allow a user based downvote of content. Most of Reddit's code is open source and maintained publicly.

B. Explanation of the Algorithm

Reddit's ranking Algorithm uses time, number of upvotes and number of downvotes as important factors for deciding the rank of a story. The Ranking formula takes into consideration the above three factors. The algorithm is biased to rate newer posts higher than older posts. However, a steady stream of upvotes is then needed to maintain ranking of older posts.
Mathematically, the algorithm can be expressed as follows:
$T_s = T_{CurrentTime} - T_{TimeofPost}$
X = Number of Upvotes - Number of Downvotes

$$y = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

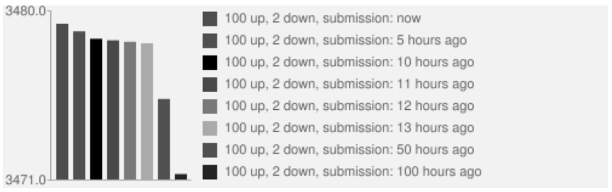
The paramemter Z is the maximal value from either the absolute value of x or 1

$$z = \begin{cases} |x| & \text{if } |x| \geq 1 \\ 1 & \text{if } |x| < 1 \end{cases}$$

Effectively, the entire forumla put together is as follows:
$f(T_s, Y, Z) = \text{Log}Z + YT_s / 45000$

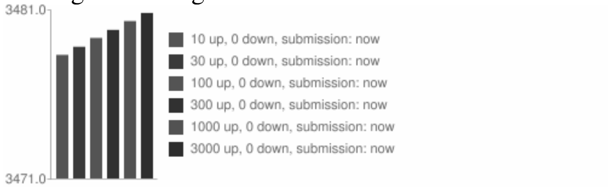
C. Effect of Posting Time on Story Rank

The Algorithm has an inherent bias towards newer posts. Unlike Edge Rank, instead of decaying a story's score over time, the Reddit Algorithm just assigns a higher value to newer stories. Thus the numerical values of scores for all stories this week would simply be higher than those of last week.

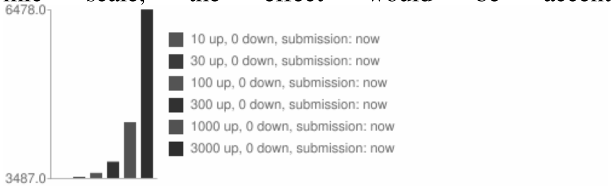


D. Effect of Logarithm of Upvotes

Taking the logarithm of the number of upvotes make the initial few upvotes count a lot more than
The following graph illustrates the effect of upvotes on rankings if the logarithm scale is not taken into consideration



Now, if we actually do consider the logarithmic scale, the effect would be accentuated.



E. Effect of Downvotes on Ranking

The effect of downvotes on the story ranking is pretty straight forward, the effect is generally linear to the differnce between upvotes and downvotes.

VII. Comparing Web Search and Ranking Algorithms

Algorithm	PageRank	HITS	TagRank
Main Technology	Web Structure Mining	Web structure Mining, Web Content Mining	Web Content Mining
Methodology	Algorithm computes scores for the ”importance” of a document based on how many pages its cited in how many pages it cites	Computes scores based on Hubs and Authority scores of pages	Annotation Time is used for Ranking
Input Parameter	Hyper-links	Forward and Back-links between hubs and authority pages	Popular tags and related bookmarks
Quality of Results	High	Less than those derived from PageRank	Less than Page Rank in general but works better on socially annotated data
Limitations	Results are predetermined during indexing and not during query	Topic drift and efficiency problem	Large data set requirement since its essentially a comparison problem

References

[1] *Sergey Brin and Lawrence Page*: The Anatomy of a Large-Scale Hypertextual Web Search Engine

[2] *Jon M. Kleinberg*:Authoritative Sources in a Hyperlinked Environment

[3] *Facebook H8 Conference*: Demystifying the Edge-Rank Algorithm

[4] *Reddit Official Github Source Code*:https://github.com/reddit/reddit/commits/master/r2/r2/lib/db/$_sorts.pyx$

[5] *S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, R. Kumar, P.Raghavan, S. Rajagopalan, A. Tomkins, Mining the Link Structure of the World Wide Web*

[6] *P Ravi Kumar, and Singh Ashutosh kumar, Web Structure Mining Exploring Hyperlinks and Algorithms for Information Retrieval*