A REPORT

ON

Improving Search Rankings by Incorporating Implicit User Feedback

AKSHIT BHATIA	2013B3A7722P
NEEL KASAT	2013B3A7670P
PRANJAL GUPTA	2013B4A7470P
VARUN VASUDEVAN	2013A7PS103P

Prepared in partial fulfilment of the course on

Information Retrieval

Submitted to
Prof. Poonam Goyal

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
NOVEMBER, 2016

ACKNOWLEDGEMENT

We would like to thank Prof. Poonam Goyal for providing us with the opportunity to carry out this project. Chandramani Mam for her valuable guidance, which kept us motivated throughout this project. Lastly, we would like to express our gratitude to all other students who have helped us in our project.

This project had been a unique opportunity and has given us great exposure of this field. We have learnt a lot during these past months .

PROBLEM STATEMENT

Similarity of the document with respect to a search query is an important and foremost criteria to rank a document while evaluating the document for retrieval during a search. Relevance, by means of PageRank, is used to evaluate the importance of the document in the graph of World Wide Web induced by hyperlinks between the document. This relevance forms the core of the Google search engine.

The aim of the project is to rate the relevant documents on the basis of User feedbacks on the earlier results and use it alongside PageRank to give the final rankings. While PageRank ranks the document based on the implicit structure of the web, incorporating User feedbacks through a system of rating called “Elo Ratings” , gives importance to User’s taste and preferences at large. By this means, rankings of the popular documents which do not score big in Similarity and Relevance can be improved upto a certain significant level so that they appear higher in the search results.

We are trying to implement the entire concept along with PageRank and Vector-space model of retrieval on the set of documents retrieved from Wikipedia and give the final results for the search query.

BACKGROUND OF THE PROBLEM

a. Motivation of the problem

Relevance is always considered to be optimum and exhaustive solution to effective information retrieval. Relevance is determined by the user interaction with the system, in the end it is up to the user to decide whether a retrieved page is relevant or not. Therefore we have given significant weightage to relevance while giving scores to the page which would decide the rank of the pages in return. Thus, incorporating the user feedback in the system.

b. Technical issues included in your work

One significant issue right from the start was the sheer volume of data which we had used was very large. This posed several problems at various stages of the project.
Right from scraping we faced problems because of the size of data. We had scrapped 1,02,103 documents whose link structure had to be maintained and had to be preprocessed to remove unwanted things like html tags. We would have loved to work in a distributed system but since we didn't have the necessary knowledge or resources we restricted ourselves to optimizing in the given environment. We made use of multithreading wherever possible to streamline the process.
The problem didn't end there either the scraped data needs to be indexed in order to retrieved as quickly as possible. Somewhere around 20,47,76,165 tokens were generated which had to be stored in the form of inverted index so as to enable quick access.
Later we were able to split the document and process them into sizeable blocks which could be stored in database and could be queried upon quickly.
Even in page ranking because of the large number of database operations over multiple iterations rendered the system slow and was not at all feasible. In order to solve this we took reduced dataset of all the edges from the database used it and wrote it back when we were done with it.
It took while to decide upon the constants and weights for Elo ratings. While merging we had to see how much weightage is to be given for the user feedback was also our sole discretion and it took us a while before deciding upon the various scores.
Above all since we were bounded by time in this project it always posed us some amount of crunch as all the above mentioned operations took great deal of time individually. For speeding up the entire process we have taken the first 50 results .
The biggest problem was restriction of net inside the campus which restricted us to run at most 20 threads.

RELATED WORK

Strategies and techniques used in the project has been adopted from the following literature :

1. **The anatomy of a large-scale hypertextual web search engine**
   Sergey Brin, Lawrence Page - 1998

The paper talks about an efficient prototype of a large scale hypertextual search engine which makes use of the PageRank algorithm for inducing ranking in the relevant set of results. It also gives the sequence of implementation and methods for bettering design goals.

2. **The PageRank Citation Ranking: Bringing Order to the Web**
   L Page, S Brin, R Motwani, T Winograd - 1999

Gives the insight upon the mathematics behind the Pagerank and implementation techniques for scaling to the large collection of pages and links structure.

3. **Rating the Chess Rating System (Elo Rating)**
   Arpad Elo, Mark E. Glickman

Elo ratings are the classic system of rating players in the 2-sided games like chess, tennis, Football, Basketball, Cricket etc. The system works by evaluating probabilities of players for winning the game based on current player ratings, and then modifying it to new based on the qualitative and quantitative outcome of the game.

4. **The Glicko System**
   Mark E. Glickman - June 1998

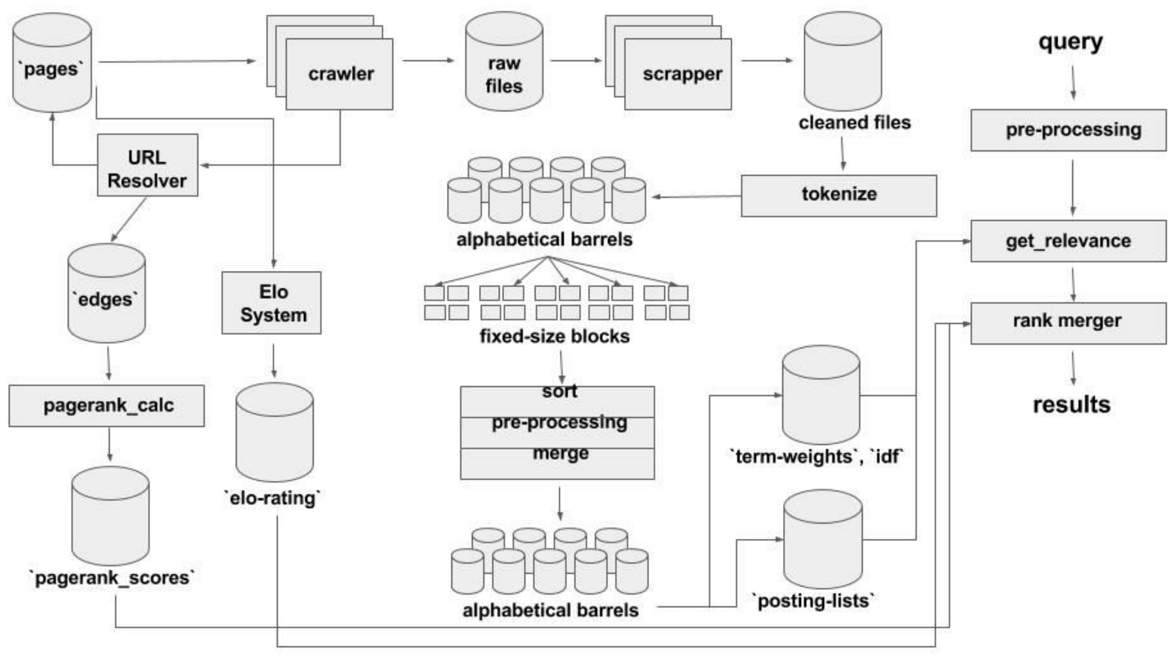
Glicko ratings addresses the problems of the classic Elo system and extends it to include confidence and trust of the ratings. Another extension to this system of ratings is extension to multi-player games which is the need of application in this project.

5. **Improving Web Search Ranking by Incorporating : User Behavior Information**
   Eugene Agichtein, Eric Brill, Susan Dumais - 2006

The paper talks about a number of user behaviours that can be incorporated to improve upon the existing relevance rankings of the search results. Among a number of behavioural aspects, the click-through information plays a major role in enhancing the existing ranking.

SYSTEM DESCRIPTION

1. Block Diagram



2. Crawling and Scraping

Crawling and Scraping basically deals with the collection of the dataset. In our case, we chose Wikipedia pages to crawl since the data in Wikipedia is very much structured and we did not need much pre-processing for converting the HTML page to text document.
Crawler and Scraper run alternatively in separate threads to achieve the above process. Simultaneously, 20 threads work together and is coordinated by the main thread in `wikie.py` file

- The **Crawler** (`crawler.py`) gets in the URL from the Main Thread and fetches and stores the corresponding Wikipedia page in raw form. A separate function in the Crawler Thread extracts all the URL in the page of the same domain (en.wikipedia.org) and adds it to the database, to be fetched later by the crawler. We crawled a total of **106,378 pages** among the repository of 1.4 million links. The crawled approx. 100,000 pages **14.3 million** links to the pages among themselves.

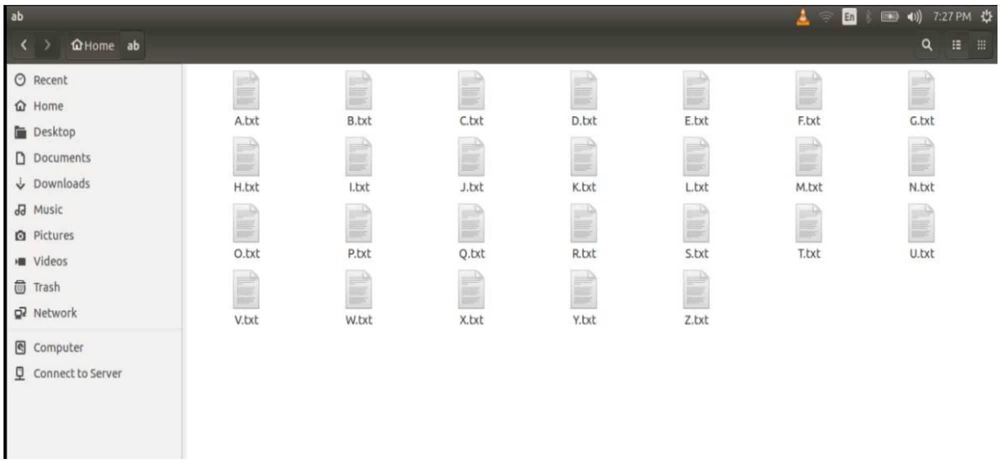
- The **Scraper** (`scraper.py`) takes a page from the raw files repository and processes the HTML document to remove HTML tags, irrelevant portions of the page. The end result of the process is a text document with only relevant text information of the corresponding document.

3. Indexing

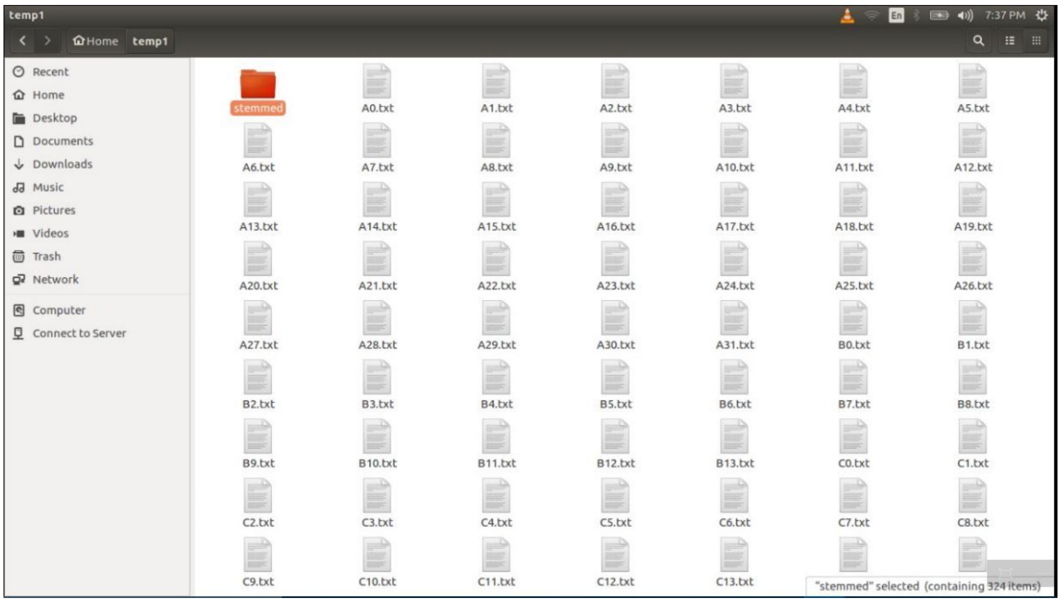
Indexing is the first step in the process of information retrieval which involves tokenizing the sentences in the document into constituent tokens and further creation of inverted index. Inverted index helps easy and fast access of the document_id of a particular word. In order to improve the runtime performance of index we have written the code in C++ implementing fast IO.On an average each document from scraped and cleaned wikipedia page contains 10,000 words which includes stop words, duplicates and other garbage words.
We have used 1,00,000 documents from wikipedia which is indexed in the manner mentioned subsequently.

3.1 Blocks and Barrels creation-

It is quite evident with the fact that since the number of documents is huge and each document has large number of words. In order to store, sort and perform other operations. The preliminary sorting involves separation of words in files according to alphabets.
In spite of separating them into preliminary blocks each file contains a large amount of data. Therefore we broke those blocks down further. Now the resulting files will have 6,50,000 words each.
Now that we have smaller files which are good enough to be loaded into the primary memory could be used for sorting and then it will be merged.


ab
Home
ab
7:27 PM
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
Connect to Server
A.txt B.txt C.txt D.txt E.txt F.txt G.txt
H.txt I.txt J.txt K.txt L.txt M.txt N.txt
O.txt P.txt Q.txt R.txt S.txt T.txt U.txt
V.txt W.txt X.txt Y.txt Z.txt

The above image shows the barrel creation initially on the basis of alphabets and later into barrels of equal size


temp1
Home
temp1
7:37 PM
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
Connect to Server
stemmed A0.txt A1.txt A2.txt A3.txt A4.txt A5.txt
A6.txt A7.txt A8.txt A9.txt A10.txt A11.txt A12.txt
A13.txt A14.txt A15.txt A16.txt A17.txt A18.txt A19.txt
A20.txt A21.txt A22.txt A23.txt A24.txt A25.txt A26.txt
A27.txt A28.txt A29.txt A30.txt A31.txt B0.txt B1.txt
B2.txt B3.txt B4.txt B5.txt B6.txt B7.txt B8.txt
B9.txt B10.txt B11.txt B12.txt B13.txt C0.txt C1.txt
C2.txt C3.txt C4.txt C5.txt C6.txt C7.txt C8.txt
C9.txt C10.txt C11.txt C12.txt C13.txt
"stemmed" selected (containing 324 items)

Equal sized barrel used for sorting.

3.2 Sorting the Files-

Just an example as to how many words there are in each alphabets-

20580200	A.txt	5934332	N.txt
8866313	B.txt	5269817	O.txt
14576232	C.txt	11969192	P.txt
7656249	D.txt	449712	Q.txt
8358756	E.txt	8768258	R.txt
9020824	F.txt	16728894	S.txt
5031381	G.txt	26975221	T.txt
7185906	H.txt	3148949	U.txt
7795792	I.txt	2615025	V.txt
2117098	J.txt	9797659	W.txt
3304063	K.txt	123975	X.txt
6438893	L.txt	1058459	Y.txt
10598410	M.txt	406555	Z.txt

On an average roughly there are about 3*10^7 which is big to be sorted on the whole as it won't be able to load the entire list into the RAM (Initially even before dividing the tokens into alphabets there were 204776165 total).

So we divided the blocks further into size of 650000.

Applying quicksort on each of the blocks and merging the subsequently to get sorted file. We made use of standard merging technique (this procedure is similar to the merging in merge sort).

3.3 Porter stemmer-

In English language morphemes are the smallest meaningful unit which cannot be divided further. Often these are present as prefix or suffix. Morphemes can be classified as-

Bound or UnBound:
Morphemes which can exist on their own are termed unbound and if otherwise it is.
Inflectional or Derivational:
Morphemes which change the use of a word are called inflectional whereas morphemes which give us plurals etc. are called derivational morphemes.
We focused a great deal on removing the inflectional and derivational morphemes since they are fixed in number and they can be removed without the need for breaking down the word.
Porter Stemmer reduces words like “run”,”running” and ‘runs“ into “run”. This helps reduce the index size significantly.

3.4 Removing Duplicates-

Upon merging we may get documents which belong to the same document. Therefore instead of maintaining duplicate term we calculate the term_frequency and maintain a separate column for term_frequency. This helps us reducing the number of words in the document.

3.5 Merging-

Once removing the duplicates we can merge the documents back but still maintaining one for each alphabet.

3.6 TF-IDF and Posting calculation-

Use the above set of files for calculation of TF, IDF, weights and postings.
(made use of linked list for storing the postings).

4. PageRank

The set of edges among 100,000 documents is 14.3 million. PageRank score is calculated from each document based on the collection of its links with other documents. The score signifies the relevance of the page in the network of WWW.

The pageRank was calculated by the following iterative formula :

$$Rank^{(i+1)} = \beta \cdot M \cdot Rank^{(i)} + (1 - \beta) \cdot \vec{e}/n$$

Here, *Rank* is a vector that holds the PageRank score of the pages and M is the adjacency Matrix. The formula is recursively applied over the vector *Rank* till the values are relatively stable and converge to a point. Ideally, log(N) iterations are sufficient to calculate a good estimate of the PageRank scores.

5. Elo Ratings and Rank Merging

Elo rating is the system of rating used for evaluating teams/individuals in sports. We have used a similar system to evaluate the pages of the search result based on the user behaviour over the pages fetched by a search query. If the user clicks a page, its rating is increased while the rating of others is decreased proportionally based on the position in which the result occurs. This means, if the last result is clicked, which is the least expected, then its ranking will be boosted to the maximum, however, if the top result is clicked, then there will only be a slight increase increase in the rating. Similarly, for the pages that are not clicked, ratings are decreased in the order of position in which the results appear.

In a way, the user behaviour is fed back in the system to improve the ratings, which ultimately affects the final rank of the page in future searches of similar query. As the number of evaluations of a query increases, the elo rankings (calculated by ratings), reach a constant which signifies the ordering based on the popularity of the page, irrespective of the PageRank.

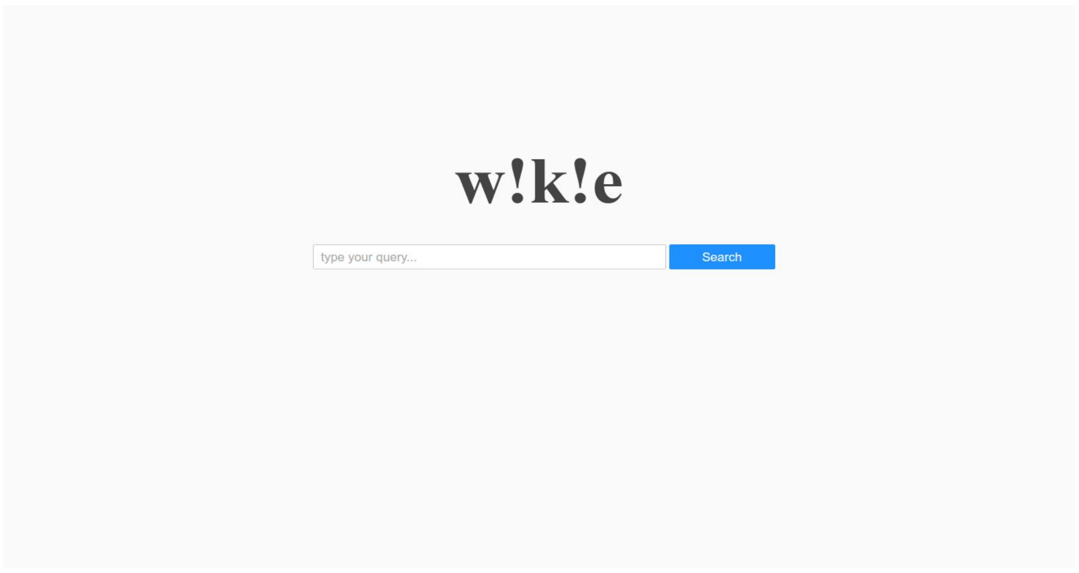
Weights are calculated to give weightage to the 3 separate ratings, viz. Similarity, PageRank, Elo, based on the confidence we have for applying Elo. The 3 rankings are finally merged to give a final ranking, which is served to the User.

6. GUI

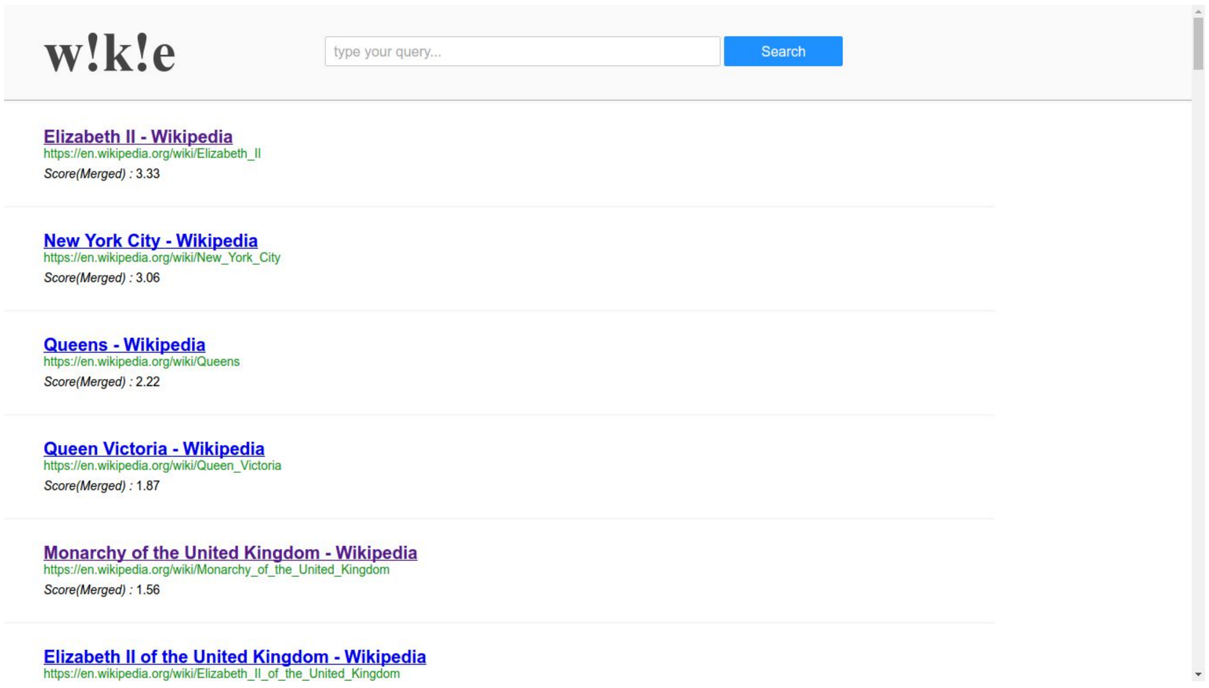
As much as the background process is important presentation is also an essential feature. The fast and robust nature of our search engine coupled with the easy access for the user gives a good overall experience.

We have made use of Django framework for presenting the search engine to the user in the form of front end. We have made it a server-client model and hosted it on localhost which could later be hosted on a remote host and could very well be a functional search engine used by the public. The benefits of using Django -

- 6.1 Easy access to the database
- 6.2 Good template models
- 6.3 Load balances to a great deal making it efficient
- 6.4 Neat layered distribution which made integration of the rest of the project easy



A view of our GUI which is essentially a form. Takes the user input in the form of GET data and gives it to the backend which upon processing gives the results back.



Elizabeth II - Wikipedia
https://en.wikipedia.org/wiki/Elizabeth_II
Score(Merged) : 3.33

New York City - Wikipedia
https://en.wikipedia.org/wiki/New_York_City
Score(Merged) : 3.06

Queens - Wikipedia
https://en.wikipedia.org/wiki/Queens
Score(Merged) : 2.22

Queen Victoria - Wikipedia
https://en.wikipedia.org/wiki/Queen_Victoria
Score(Merged) : 1.87

Monarchy of the United Kingdom - Wikipedia
https://en.wikipedia.org/wiki/Monarchy_of_the_United_Kingdom
Score(Merged) : 1.56

Elizabeth II of the United Kingdom - Wikipedia
https://en.wikipedia.org/wiki/Elizabeth_II_of_the_United_Kingdom

EVALUATION STRATEGY

Crawling and Scraping:

We made use of the following libraries in python-

1. Request
2. BeautifulSoup

for crawling the Web and Scraping the wikipedia pages.

Page Rank and Elo Rating:

The formula used for getting **PageRank** is :

$$PR(A) = (1-d) + d\ (PR(T_1)/C(T_1) + ... + PR(T_n)/C(T_n))$$

where,

PR(A) is the PageRank of page A,
PR(Ti) is the PageRank of pages Ti which link to page A,
C(Ti) is the number of outbound links on page Ti and
d is a damping factor which can be set between 0 and 1.

The basic approach of PageRank is that a document is in fact considered more important the more other documents link to it, but those inbound links do not count equally. First of all, a document ranks high in terms of PageRank, if other high ranking documents link to it.

Calculating ELO Rating :
Expected Probability of being clicked :

$$Sexp = 1/(1 + 10 \sum_b^z -(Ra - Rj))$$

Formula for updating **Elo Rating :**

$$r_{post} = r_{pre} + K(S - S_{exp}),$$

where , K = 100, S = Result of the search (0 = not clicked, 1 = clicked)

Rank Merger Formula :

W_r (weight given to the similarity score) = **0.4**
W_p (weight given to the PageRank ranks) + W_e(weight given to the Elo ranking) = **0.8**

Contribution of Elo in $W_p + W_e$ = $(0.5) * (1 - e^{-[hm]/5})$

hm is the Harmonic Mean of the number of times the Elo has been applied to the results. This ensures that the Contribution of elo increases as the number of times the system used is increased and reaches to a final value.

Final score is calculated as :

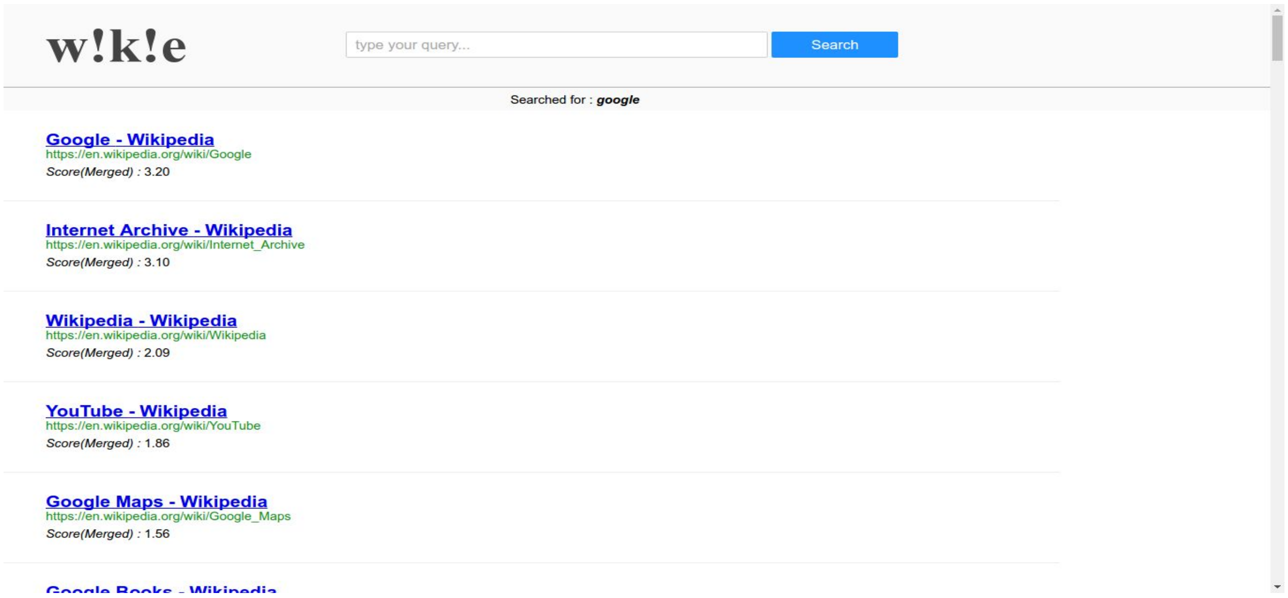
$$\mathbf{Score(Merged) = W_r * (1/R_r+1) + W_p * (1/R_p+1) + W_e * (1/R_e+1)}$$

where,
R_r - Rank from similarity, R_p - Rank from pagerank score, R_e - Rank from Elo-rating score

EXPERIMENTAL RESULTS AND EVALUATION

We were able to successfully incorporate user behaviour by using Elo ratings. The search results changed dynamically based on the user choice which decides the relevance of the page.
Upon searching the query “google maps” we get the following result-

The below picture corresponds to the colds start. The search is dominated by page rank.



Searched for : *google*

Google - Wikipedia
https://en.wikipedia.org/wiki/Google
Score(Merged) : 3.20

Internet Archive - Wikipedia
https://en.wikipedia.org/wiki/Internet_Archive
Score(Merged) : 3.10

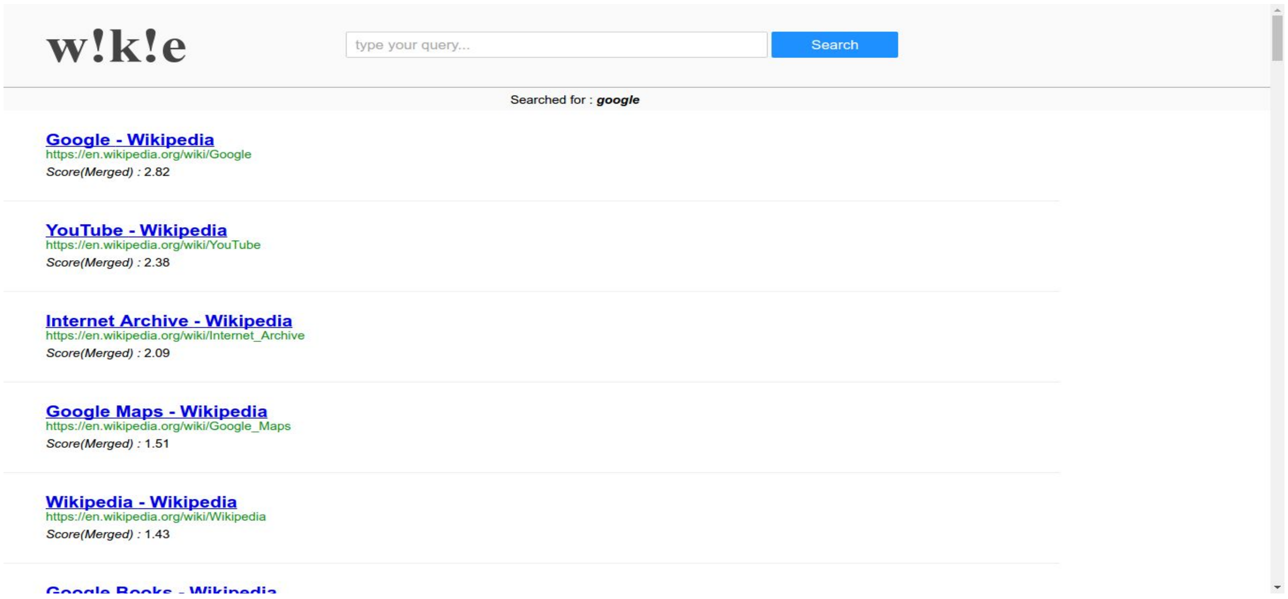
Wikipedia - Wikipedia
https://en.wikipedia.org/wiki/Wikipedia
Score(Merged) : 2.09

YouTube - Wikipedia
https://en.wikipedia.org/wiki/YouTube
Score(Merged) : 1.86

Google Maps - Wikipedia
https://en.wikipedia.org/wiki/Google_Maps
Score(Merged) : 1.56

Google Books - Wikipedia

Upon repeatedly selecting youtube by the user the result got changed and youtube got a better score.



Searched for : *google*

Google - Wikipedia
https://en.wikipedia.org/wiki/Google
Score(Merged) : 2.82

YouTube - Wikipedia
https://en.wikipedia.org/wiki/YouTube
Score(Merged) : 2.38

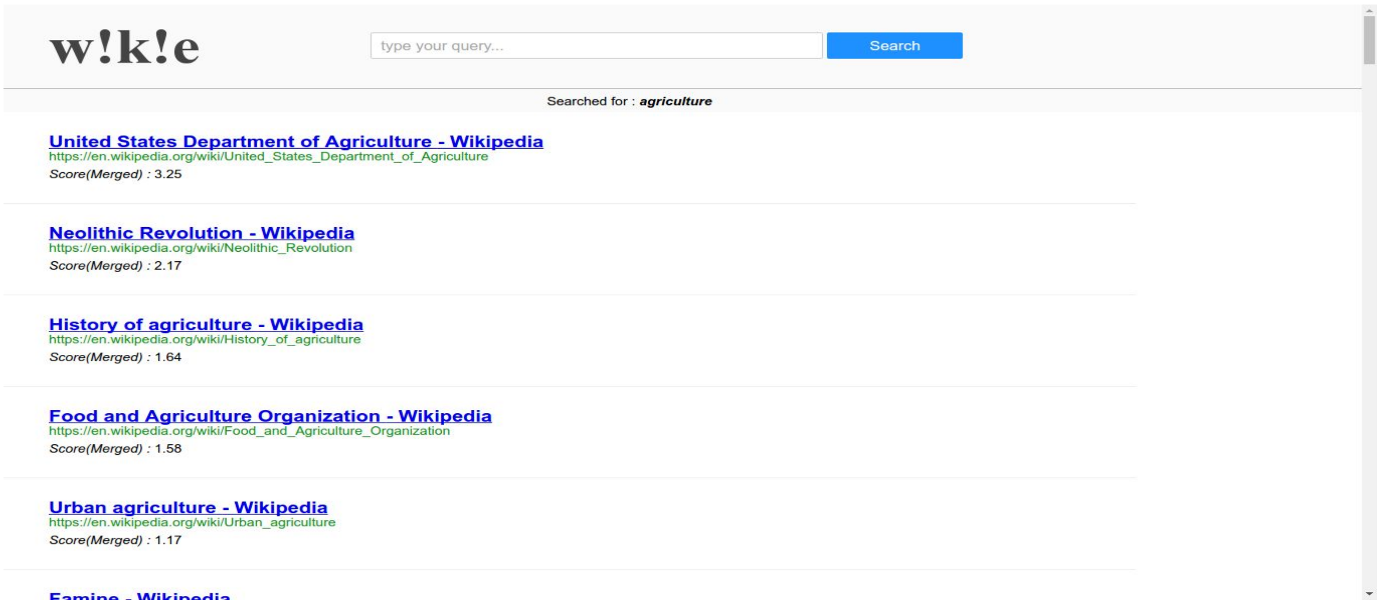
Internet Archive - Wikipedia
https://en.wikipedia.org/wiki/Internet_Archive
Score(Merged) : 2.09

Google Maps - Wikipedia
https://en.wikipedia.org/wiki/Google_Maps
Score(Merged) : 1.51

Wikipedia - Wikipedia
https://en.wikipedia.org/wiki/Wikipedia
Score(Merged) : 1.43

Google Books - Wikipedia

Another example-
The user searched “agriculture” and the initial result-



Searched for : *agriculture*

United States Department of Agriculture - Wikipedia
https://en.wikipedia.org/wiki/United_States_Department_of_Agriculture
Score(Merged) : 3.25

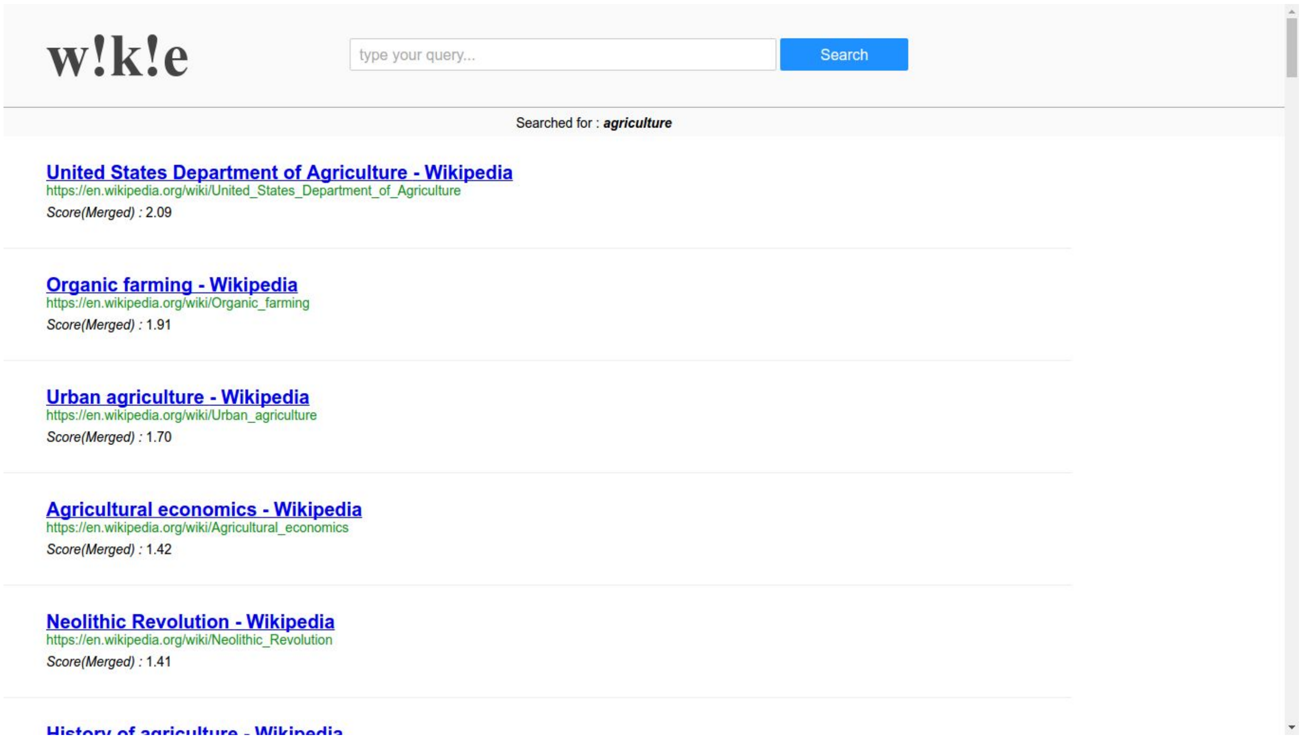
Neolithic Revolution - Wikipedia
https://en.wikipedia.org/wiki/Neolithic_Revolution
Score(Merged) : 2.17

History of agriculture - Wikipedia
https://en.wikipedia.org/wiki/History_of_agriculture
Score(Merged) : 1.64

Food and Agriculture Organization - Wikipedia
https://en.wikipedia.org/wiki/Food_and_Agriculture_Organization
Score(Merged) : 1.58

Urban agriculture - Wikipedia
https://en.wikipedia.org/wiki/Urban_agriculture
Score(Merged) : 1.17

Upon clicking organic farming and urban agriculture the result got changed to the following given more score to the above mentioned also reducing the scores of other documents.



Searched for : *agriculture*

United States Department of Agriculture - Wikipedia
https://en.wikipedia.org/wiki/United_States_Department_of_Agriculture
Score(Merged) : 2.09

Organic farming - Wikipedia
https://en.wikipedia.org/wiki/Organic_farming
Score(Merged) : 1.91

Urban agriculture - Wikipedia
https://en.wikipedia.org/wiki/Urban_agriculture
Score(Merged) : 1.70

Agricultural economics - Wikipedia
https://en.wikipedia.org/wiki/Agricultural_economics
Score(Merged) : 1.42

Neolithic Revolution - Wikipedia
https://en.wikipedia.org/wiki/Neolithic_Revolution
Score(Merged) : 1.41

CONCLUSION AND FUTURE WORKS

- We would really want to apply the methods above in a distributed system just like real search engines do to make the procedure faster.
- User behaviour could be trained by using a more superior model like Neural Network which would capture the behaviour more dynamically.
- Implementing features like safe search (filtering the search for restricted access) and a coupled image search where even relevant images from the pages are retrieved and presented to the user.
- Making the indexing and PageRank calculations dynamic where new pages will be crawled,scraped, preprocessed and would be added on the fly is also surely a task in hand.
- Right now we have done all the file handling IO operations on normal text files the speed could further be improved by using binary files instead of text files.

The System can be efficiently used to enhance the search result by incorporating User Feedbacks. This data can serve as the benchmark for evaluation popular trends, hence displaying a trending page higher in the search rankings. The system can also be vital in evaluating the pages based on its significance among the Users rather than just evaluating by link structure (PageRank scores) or similarity to the query (Similarity score).