Available online at http://www.journalijdr.com

IJDR

ISSN: 2230-9926

International Journal of Development Research
Vol. 08, Issue, 06, pp.21074-21080, June, 2018

ORIGINAL RESEARCH ARTICLE OPEN ACCESS

UNCOVERING USER'S SEARCH PATTERNS TO PERSONALISE WEB SEARCH

Smita Sankhe and *Nirmala Shinde

Department of Computer Engineering, K. J. Somaiya College of Engineering, Mumbai, India

ARTICLE INFO

Article History:
Received 09th March, 2018
Received in revised form
20th April, 2018
Accepted 18th May, 2018
Published online 30th June, 2018

Key Words:

Data mining, Hierarchical clustering, Machine learning, Natural language processing, Search methods, User Modeling, Web mining.

ABSTRACT

In today's world, search engines have become a very convenient method of searching and retrieving information. But this increasing use of search engines goes hand in hand with the ever-increasing data available on the internet. With such large number of websites available, it is essential to have these websites sorted in decreasing order of their relevance to the user's query for effective operation and retrieval of data. This paper explores various domains related to Computer Science and proposes a framework that seems the best fix to this problem. We have proposed a new system to provide personalized web search according to the user's internet surfing patterns. The system extracts the user's history and scrapes the web pages' content (title, keywords, headings, sub-headings, meta tags). These documents are then clustered using Word2Vec model and Latent Semantic Indexing to give better results. User's search query is mapped to the profile and an appropriate cluster is selected. The SERP returned by the search engine is mapped to the selected cluster to find the similarity index. A linear regression model is used to assign the final score which takes the regency, frequency, popularity and user's feedback along with the similarity measure to re-rank the SERP.

Citation: Smita Sankhe and Nirmala Shinde. 2018. "Uncovering user's search patterns to personalise web search", *International Journal of Development Research*, 8, (06), 21074-21080.

INTRODUCTION

The user profile is created using the user's browser history file and other typical behaviors on the internet, such as the bookmarked web pages, accounts created on various portals, subscriptions, etc. All these attributes can be used to construct certain patterns for this profile, which will be employed for effective re-ranking of SERPs for future queries. We also make an effort to date this user profile, that is, the more recent entries and activities hold a higher significant value and vice versa. This is because human interests and behaviors tend to change over periods of time. This period of time can be neither quantified nor generalized. Hence, the method of dating entries ensures that recent activities hold more relevance than the previous activities, which led us to achieve better results when tested again other results. Some users can be new to the browser, others can have a full-fledged browser history, sufficient to analyse its history files to identify patterns and trends.

***Corresponding author:* Nirmala Shinde,**
Department of Computer Engineering, K. J. Somaiya College of Engineering, Mumbai, India.

User profiling was found to be effective only for the experienced/regular users. For new users, using information from browser history files is not feasible. Hence, we made provisions for new users by taking inputs about user's interests in a separate form that consists of keywords (interest topics) and accordingly tailoring the SERP to provide effective personalization. Gradually, as the user uses the browser, the history file will record his activities and once this history file possesses information exceeding a certain threshold, the program will switch to providing dynamic personalization of web search. The model also uses more than one algorithm for many tasks, to ensure that features and details of all attributes are correctly captured and addressed by the model and are used effectively to improve its accuracy and performance. Apart from the model, certain novel modules and functionalities have also been incorporated into our project which increases the personalization factor and successfully reduce the user's time for information retrieval. This problem can certainly appear to be small and negligible at the microeconomic level, but this is not the case. For most organizations, the employees tend to make use of Search Engines on a daily basis for a wide range of tasks.

On such a large scale, surfing through several results from the SERPs to find the right match consumes the precious time of their employees. This shoots up the overall expenses of that organization. The more the dependency on Search Engines of an organization, the higher will be the associated expenses. This is just one of the numerous problems we come across in our day-to-day life. Finding the right information on the internet nowadays is like finding a needle in a haystack. After researching extensively on all existing solutions and their drawbacks, we designed our own model that improves upon them, while simultaneously minimizing the drawbacks. The user profile is created using the user's browser history file and other typical behaviors on the internet, such as the bookmarked web pages, accounts created on various portals, subscriptions, etc. All these attributes can be used to construct certain patterns for this profile, which will be employed for effective re-ranking of SERPs for future queries. We also make an effort to date this user profile, that is, the more recent entries and activities hold a higher significant value and vice versa. This is because, in our surveys, we noticed that human interests and behaviors tend to change over periods of time. This period of time can be neither quantified nor generalized. Hence, the method of dating entries ensures that recent activities hold more relevance than the previous activities, which led us to achieve better results when tested again other results.

Some users can be new to the browser, others can have a full-fledged browser history, sufficient to analyze its history files to identify patterns and trends. User profiling was found to be effective only for the experienced/regular users. For new users, using information from browser history files is not feasible. Hence, we made provisions for new users by taking inputs about user's interests in a separate form that consists of keywords (interest topics) and accordingly tailoring the SERP to provide effective personalization. Gradually, as the user uses the browser, the history file will record his activities and once this history file possesses information exceeding a certain threshold, the program will switch to providing dynamic personalization of web search. To begin with, we have only designed the program to support Google since it is the most widely used and preferred search engine around the world, and is easy to use and access. A browser history will keep changing overuse. Due to this constantly changing property of the history file, the model is designed to be dynamic in order to handle this dynamicity of browser's history files. The model aims to resolve inherent ambiguities in search queries by analyzing user's browser history and using this information to provide context to the queries. The proposed model has been hybridized in many aspects to extract advantages of both features, static and dynamic.

Analysing browser's history and clustering on the browser's history pages repeatedly can become computationally heavy and time-consuming. Hence, the model is designed to be semi-dynamic, that is, it will not perform hierarchical clustering for every task; It will only do it once in a day or as specified by the user or when there is little to no load on the server, to improve its real-time performance and decrease computations required to arrive at a result. The model also uses more than one algorithm for many tasks, to ensure that features and details of all attributes are correctly captured and addressed by the model and are used effectively to improve its accuracy and performance. This will be explored in-depth in the later part of this paper. Apart from the model, certain novel modules and functionalities have also been incorporated into our project which increases the personalization factor and successfully reduce the user's time for information retrieval. In this paper, we explore incorporate a number of algorithms and techniques coming from various domains which seem suitable for our ideas and help us in achieving our objectives. We also substantiate these decisions with concrete reasoning and germane evidence.

The remainder of the paper is organized as follows:

- Literature Survey
- Architecture of Proposed System
  - User Profile Generation / Updating
  - SERP scoring and ranking
- Experimental and Result Analysis
- Conclusion
- Future Scope

Literature Survey: Our research led us to discover interesting implemented systems and identify their drawbacks, which are as follows: Reference [Makvana et al., 2014] carries out Personalization of web search results by processing user's query and re-ranking the results depending upon the interest of the user. A 3-Level weblog user profile is created from the weblog data, where the 1st level contains the keyword of the query, 2nd level contains the sub-queries and the 3rd level contains the records of the URLs visited by the user. The algorithm re-ranks the results using Vector Space Model depending upon the user's activities. Then, cosine similarity is used to find the correlation between the search query and the data stored in the log file. So when a user searches for a query, it retrieves the results in a hierarchical manner. Reference [Kumar, 2014] proposes to construct an Enhanced User Profile by considering the browsing history of the user and enhancing it by using domain knowledge. Domain knowledge (DMOZ) is the background knowledge that is used to enhance the user profile. Some web pages are crawled from the DMOZ directory for specific categories. For classification of web pages, Alchemy API is used which shows the probability of a web page belonging to a particular category, and are mapped to the respective DMOZ categories. Cosine similarity is calculated between the URL and the URLs present in the knowledge base to prepare Enhanced User Profile. [Renjini, 2016] personalizes the search results depending on Query expansion and Clustering. The user profile is created using the search history. It uses k-means clustering to group similar documents into clusters, and the search is performed only on the centroids of each cluster instead of all documents. Query expansion is done with identical words by using Word Net. The history is refreshed every 60 days to include the most recent interests only. [Kacem, 2017] aims to improve the sessions' results by considering the user's current interactions are more relevant to the user than the earlier ones. The vector of keywords is used to represent the user profile. TF-IDF scheme is used to assign the weights along with its temporal characteristics. Session search aims to deliver the most meaningful results to the present need of the user [Altszyler, 2016] aims to compare the performance of LSA and Word2Vec embedding's in medium and small-sized corpora. It was shown that on training the database with medium to large size corpora, Word2Vec outperforms LSA but in small corpora, LSA gives accurate semantic relation[Anikó Hannák, 2017]. Presents a way to measure the extent of personalization of web search. Jaccard Index views the result lists as sets and is defined as the size of the intersection over the size of the union. Second, the value of Kendall's Tau coefficient for two ranked lists is obtained by measuring the difference in results among two ranked sets.

Architecture of Proposed System: The entire architecture is designed to be dynamic, robust and requires minimal computations. Since the program will be generating the output in real-time applications, speed and efficiency are absolutely imperative.

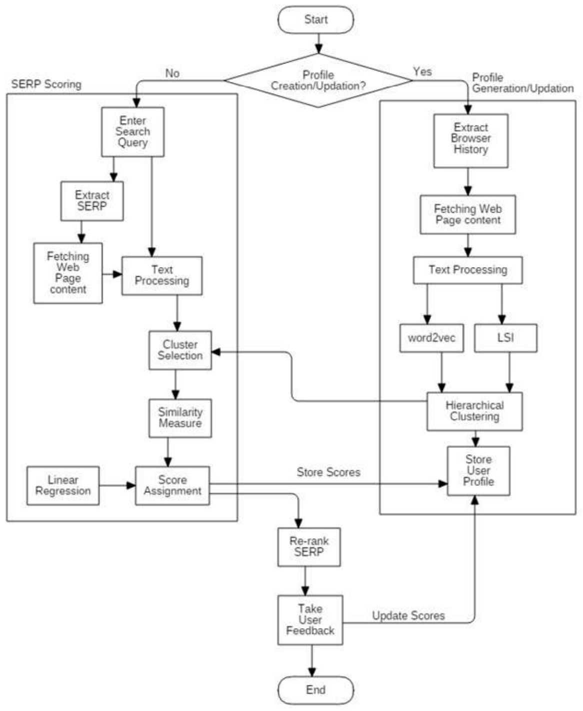


Fig. 3.1. System Architecture

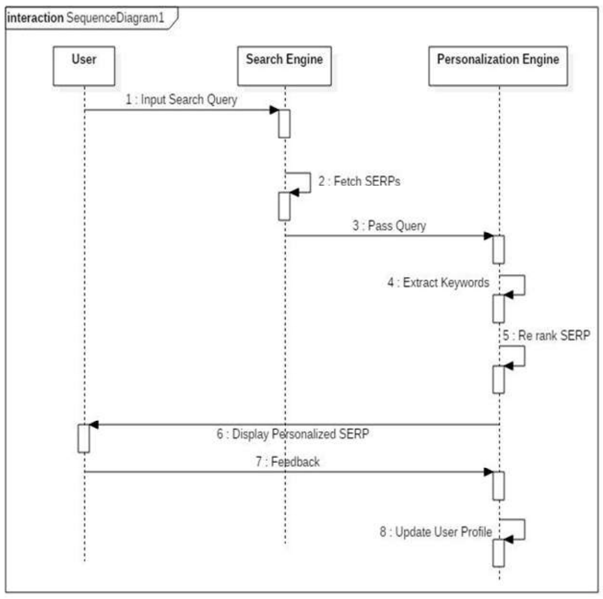


Fig. 3.2. System Work Flow

There proposed system consists of two broad groups of steps:

- User Profile Generation / Updation
- SERP Scoring and Ranking

We selected two models for calculating the distances between every pair of documents in the corpus: Latent Semantic Indexing and Word2Vec. LSI overcomes the problems of synonymy and polysemy. LSI model training can also be resumed from any point, thus resulting in reducing training times for the LSI model and faster computations. The second selected model is the Continuous skip-gram architecture of the Word2Vec model since it is known for outperforming all its counterparts in medium-sized datasets. According to our surveys, the window size of 10, 7 negative samples and number of dimensions restricted to 350 seemed to be the most optimal trade-off between accuracy and computational time. The combination of text processing models is chosen such that they perform optimally irrespective of the size of the corpora since LSI is marked to perform well even with a small training corpus whereas Word2vec skip-gram model is marked to perform exceptionally when trained on the large-sized corpus.

User Profile Generation / Updation

Fetching contents of the history file: The program begins with accessing the history file of Google Chrome browser. If the history file is opened successfully, then the program initiates reading from the history file and copies this content into a CSV file. This CSV file is the primary file of the program, where all the processing and storing of data occurs.

Web Scraping: Each result entry extracted from the SERP is a web page and has the characteristics of an HTML file. After completing the extraction of SERPs, we scrape through top 27 resultant web pages returned by the program, and extract the key attributes from each of these pages' tags, such as the metadata tags, title tags, keyword tags and heading tags to mention a few, and are stored in their respective fields.

Text Processing

Tokening and Removal of stop words: All this scraped information is collectively referred to as the corpus of documents. For text processing, the entire corpus was parsed and converted into tokens. After tokenizing the corpus, the program filters out the stopwords present among the tokens. This step ensures that only keywords are preserved and the unnecessary words are eliminated since the stop words are of no real value and are found to impact the results negatively when preserved in the document.

Stemming: Now, stemming is performed on each token of the corpus using the Porter stemmer. This is done to generalize words with similar origins but different tenses, to combine singular and plural words and remove affixes from words; ultimately, the canonical form of each word is obtained. Resolving all these conflicts enables the tracing back words that belong to the same word or same stem, in essence. It also reduces the number of words in the dictionary, thereby resulting in faster processing. Unlike lemmatizers, stemmers don't need any extra knowledge about the context of the word and can only map to some word present in the dictionary. Our dictionary not being exhaustive is another reason to prefer stemmer over lemmatizer.

Converting tokens into vectors: A large portion of the constructed dataset shall consist of text. Machine learning or data mining models cannot directly process raw text to make predictions. Texts need to be converted to some numerical value in order to be of use for the machine learning model. After obtaining a corpus with normalized documents, the corpus is then processed and transformed into a Dictionary. A dictionary is a data structure that saves each word as its key and maps it to the corresponding key ID as its value. For example, let's assume 'motor' is the first word of the dictionary and occurs twice in a document. Then, 'motor' will be represented as {'motor':2}. This dictionary is later used as a reference to convert each document into Bag of Words (BoW) model. For each document, the BoW model represents each word by its key ID paired with its number of occurrences in that document. This conversion greatly contributes to dimensionality reduction and makes this the raw text fit for use by the machine learning models. As noted above, models perform differently under different circumstances. On the basis of this observation, we designed an equation that would capture the advantages of both models and be used to increase the combined efficiency of the models. This equation is dependent on the number of dimensions. The contribution made by these 2 models is dictated by the following equation:

$$S = A * x + B * y \qquad (1)$$

Where,

S = Combined Similarity Score
X= similarity value returned by Word2Vec model.
Y= similarity value returned by LSI model
A and B are coefficients, which are calculated as follows:
$A = (n - 0.05) / (10 \text{ million})$ for $n < 8$ million
$A = 0.75 + (n - 8) / (40 \text{ million})$ for $8 <= n < 10$ million
$A = 0.8 + (n - 10 \text{ million}) / (100 \text{ million})$ otherwise
And $0 <= A <= 0.87$
$B = 1 - A$
n = size of the corpora

We use 10 million as a metric because it was reported by [Altszyler, 2016] that at a corpus size of 100 million, Word2Vec with negative sampling Continuous skip-gram model outperforms LSA, and hence the contribution made by Word2Vec at this level is made significantly higher than LSI's contribution to accommodate these advantages into our combined models. As noted by the paper [Altszyler, 2016], the performance of Word2Vec continuous skip-gram model severely decreased when trained on a small-scale dataset, whereas it outperformed LSI model when trained on medium to large-scale datasets. Additionally, LSI was marked to accurately identify polysemy and synonymy. Using these vectors, we use ensemble model, comprising of Word2Vec and Latent Semantic Indexing similarity to find the distance between different pairs of documents. Using this distance as a metric, we use hierarchical clustering to cluster documents based on their similarity. Using hierarchical clustering offers 3-fold benefits. First, the shape of clusters need not be spherical. Since we are considering the total population, their usage behavior and statistics cannot be generalized. Hence, this program required a more generalized and dynamic technique that could handle clusters of all shapes without compromising its accuracy and performance. Additionally, the number of clusters need not be predefined, which is a huge plus, because every user would have a different number of entries in his history file, the difference can vary up to 7 figures. Hence, hierarchical clustering would assist in deciding upon the correct number of clusters depending on the number of entries and fully automating the task of clustering. Lastly, it provides hierarchical relations between clusters and can capture concentric clusters. It becomes easier to divide clusters and helps in selecting the more appropriate hierarchical level that would give the highest accuracy. We use the similarity metric dictated by the word2vec and LSI models as input to the hierarchical clustering model, which is trained on the average linkage method. Then, a dendrogram for this model is constructed which gives valuable insights regarding the structure of the corpus. This model is then saved as the User Profile.

SERP Scoring and Ranking

SERP Fetching and Scraping: The user's search query is entered in the Search Engine. The SERP returned is then fetched and the contents of these web pages are extracted in the similar format as for the History Pages. The query to be run through Google is taken as input from the user. The query is of prime importance since the whole model is designed to optimize the results for it. After running the query through Google, top 27 results from Search Engine Result Pages (SERPs) are extracted. The program provides the flexibility to change this variable number (27) as preferred by the user, but through our research, we have come to the conclusion that 27 is the ideal number of results to be taken into consideration since it maintains a perfect trade-off between computational costs and accuracy. Hence, the number 27 is set as the default value for this entry.

Text Processing: The web page contents are again vectorized and converted to Bag-of-Words (BoW) format. The query terms are also tokenized and vectorized.

Selection of the cluster: We then use the vector form of input query and compute its distance to all the leaf clusters of the dendrogram (User Profile). Based on these distances, we select the top k most similar clusters and trace them back to their least common ancestor. All the leaf nodes (children) of this ancestor are combined by performing union operation on them. This combined cluster should have a count of documents exceeding 5% of the total number of documents which makes it significant when compared to the 5% rule of neglecting. For any value of k, if the cluster fails the 5% rule, the value of k should be incremented until it becomes true.

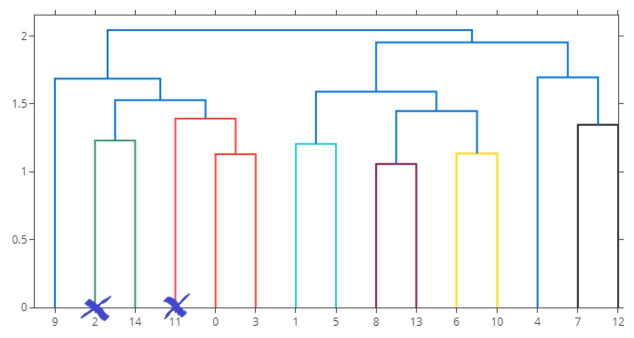
We use the following algorithm to find the value of k:

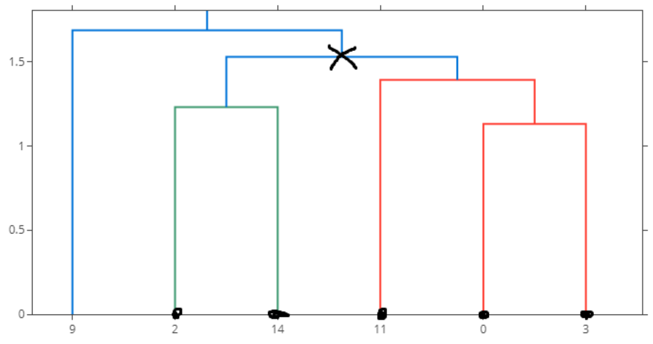
```
n = len(corpus)
  for i :=  0 to n

If count(intersection(top i clusters)) >=  5/100*n :
then k := i

    else k++
```

For example: Let us assume that for k=2, cluster number 2 and 11 are the top 2 results.

After performing the algorithm on the above dendrogram, we get:

At the intersection point, we consider all of the children nodes and take its union to get the resultant cluster.





Here, it would include cluster numbers (2, 14, 11, 0, 3). This algorithm fully utilizes the hierarchical order and gives exceptional results as compared to other cluster counterparts. Also, this combined cluster is used for computing the score of the each of the scraped Search Engine results. Similar operations are performed on the user query taken as input from the user. The distances between the vector form of this query and all other clusters are calculated, and the one with a minimum distance of all is selected. From this selected cluster, we extract all the key attributes. This set of attributes is now used to evaluate the top 'n' results fetched from the SERPs. The similarity of each result belonging the SERPs is calculated, and these values are saved in the respective fields. The user is provided with a like button and a dislike button next to each of the results provided to him. The like button would indicate that the user is happy with that result, and would increment the count in the positive feedback column, whereas the dislike button expresses the undesirability of that result entry, and increments the count for that domain in the negative feedbacks column.

Time Dimension Sensitivity: It was observed that more recent searches and browser activities held more significance and offered more value than their less recent counterparts. Hence, we have attempted to sensitize the time dimension of browser activities. nWe rounded off the time vector of each activity to a number of days. The equation that was used to compute the score is as follows:

Time Score Equivalent = pi / (x + pi)
Where pi= 3.14159265...

x = number of days obtained after rounding off, ranging from 0 to infinity.

It was observed in [Kacem, 2017] that activities, not more than 2 months old have been noted to contribute most significantly. The equation is formulated such that it allows for only the last 60 days activities to be deemed significant according to the 5% rule to deem a value negligible. The 5% rule states that any value less than 5% of the maximum value with which it is compared, can be neglected. Using this equation, we not only take into account activities older than 60 days but also ensure that their contribution, while taking into account, is not very significant.

Re-ranking Algorithm: After forming clusters and scraping data from key tags of web pages belonging to the web pages, we use the LSI and Word2Vec models to find the similarities between all the documents belonging to the selected cluster with each of the web pages of the SERP. This similarity value is used in conjugation with various other fields to compute the final score of each entry of the SERP.
This score is used to sort and rank the web pages in descending order and this final list is presented to the user. In the final phase of this processing, we use a dataset with the following tuple to train our Linear Regression model: We use the normal equation to solve and fine-tune the parameters of the Linear Regression model, since the combination is very cheap computationally for a dataset with small features and linearity, such as ours. It fits the data exceptionally well and also eliminates the need to manually set the learning rate parameter like in gradient descent. After training the model, the scaled equation of the model's prediction value (score) is described as follows:

Score = 0.61*Similarity + 0.08*Frequency + 0.13*Positive Feedback - 0.36*Negative Feedback + 0.11*Bookmarked + 0.25*Time Dimension + Search Engine Rank*0.15 + 0.17.
After applying this algorithm, we extract the top 14 results out of the set of 27 results and present it to the user. The number 14 was chosen to be the optimal value since, during our surveys, it was observed that 91% of the people did not scroll through more than 14 top results.

User Feedback: The user is provided with a like button and a dislike button next to each of the results provided to him. The like button would indicate that the user is happy with that result, and would increment the count in the positive feedback column, whereas the dislike button expresses the undesirability of that result entry, and increments the count for that domain in the negative feedbacks column. This is again saved in the User Profile tuple for calculation the rank of the domain.

Experimental and Result Analysis: We arranged for all students of our class to test this program and recorded their feedbacks on it. We also arranged for some random people to start as new users to test the static profile building feature of the program and recorded the results. For both sets of people, we made provisions for them to compare the previous SERPs with the modified, re-ranked SERPs and make comparisons on the basis of their personal relevance and relatability. We used a grading score to evaluate our software, ranging from 0 to 10, with 10 being the most relatable and highly relevant pages being promoted up the ladder plus extreme convenience in navigation and browsing through the SERPs and 0 being all relevant and important pages being demoted down the ladder plus extremely problematic to browse through the SERPs and navigation.

The entire architecture is designed to be dynamic, robust and requires minimal computations. Since the program will be generating the output in real-time applications, speed and efficiency are absolutely imperative.

There proposed system consists of two broad groups of steps:

- User Profile Generation / Updating
- SERP Scoring and Ranking

We selected two models for calculating the distances between every pair of documents in the corpus: Latent Semantic Indexing and Word2Vec. LSI overcomes the problems of synonymy and polysemy. LSI model training can also be resumed from any point, thus resulting in reducing training times for the LSI model and faster computations.

Dataset for Evaluation: The dataset is created dynamically by extracting the browsing history of the Google Chrome Browser of the user. The browsing history of the users is stored at ~/AppData/Local/Google/Chrome/User Data/Default/History location. This file is in SQLite format. We first extract the history of the user from this location and then analyse the file to extract relevant details such as the URL and the number of times the user has visited that link.

A sample of the extract history is as follows:

```
docs.google.com 36319
mail.google.com 4114
drive.google.com 3414
google.co.in 20095
app.applyyourself.com 788
localhost 701
youtube.com 750
localhost:5984 504
linkedin.com 383
classroom.google.com 240
google.com 234
github.com 220
apps.grad.uw.edu 217
netflix.com 214
toefl-registration.ets.org 178
gmail.com 176
choose.illinois.edu 164
1337x.to 156
heinz.cmu.edu 154
applyweb.com 142
facebook.com 141
accounts.google.com 137
```

The second selected model is the Continuous skip-gram architecture of the Word2Vec model since it is known for outperforming all its counterparts in medium-sized datasets. According to our surveys, the window size of 10, 7 negative samples and number of dimensions restricted to 350 seemed to be the most optimal trade-off between accuracy and computational time. The combination of text processing models is chosen such that they perform optimally irrespective of the size of the corpora since LSI is marked to perform well even with a small training corpus whereas Word2vec skip-gram model is marked to perform exceptionally when trained on the large-sized corpus. For comparison, we consider the top 14 results returned for a generic guest profile and for a specific user's profile. To gain more concrete results on the measurement of personalization of results, we use the ones suggested by [Anikó Hannák, 2017] and compare these two sets of results with each other, Jaccard Index and Kendall's Tau coefficient. Jaccard Index is an indicator of the number of overlapping entries present in both sets that range from 0 to 1. Here, 0 would indicate that there are is no such resultant entry that is present in both the sets and 1 would imply that both result sets contain the exact same entries. On the other hand, Kendall's Tau coefficient ranges from 0 to 1 and measures ordinal relation between these two sets, which would indicate the correctness of the ranking assigned to the calculated fraction of the resultant sets. We get the following graph on comparing personalized results returned by our program for 10 user queries with the SERPs returned by Google.

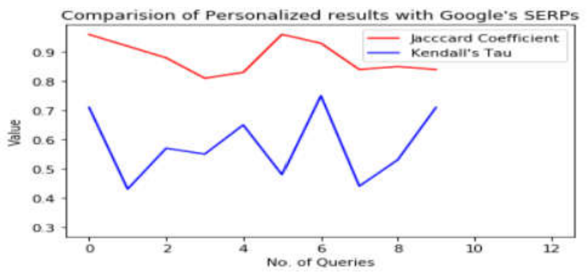


Fig 4.1. Comparison of personalized results with Google's SERPs

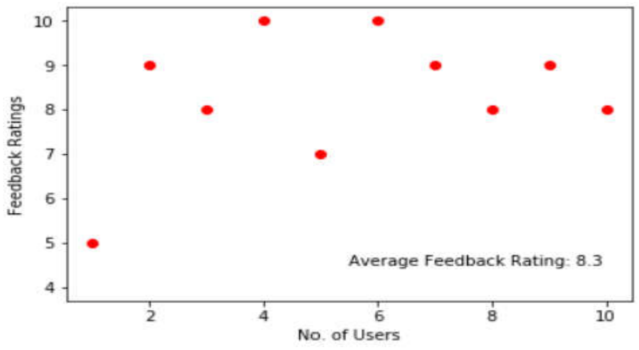


Fig 4.2. Statics of Average Feedback Rating

Similarity Measure	Real number belonging to [0, 1]
Bookmarked	Boolean (true / false)
Domain Frequency	Normalized Fraction
Positive Feedback	Integer
Negative Feedback	Integer
Search Engine Rank	Positive Integer
Time score	Real number belonging to [0, 1]

- From Jaccard Coefficient, we observe high correlation among the result sets returned by both methods
- From the graph of Kendall's Tau coefficient, we learned that although there is a high correlation between the two sets, there is low to medium ordinal association between them.
- To calculate user satisfaction, the experience was rated on a scale of 1 to 10
- And the average rating received from 10 users was an astounding 8.3 out of 10.
- The distribution was as follows:

Conclusion

We have proposed a new system to provide personalized web search according to the user's internet surfing patterns. Personalization plays an important role in finding out the results which are according to the user's context. Hence in this project various factors like user feedback and search history are used to obtain personalized result to the users.

User search history is observed to find out which the links that the user visits frequently and to understand what type of content he surfs. Feedback from the user gives direct information whether the result is useful or not while collaborative search works on the basic principle that like people have like interests. The semi-dynamic nature of the proposed system, allows addition of newer entries and the change of user's interest can also be incorporated. Personalization can be useful as it saves time and efforts both of a user in finding the expected results. Thus, we conclude that the proposed system achieved its objectives, thereby significantly reducing the time related to extracting information from search engines.

Future Work and Scope: The current ideas can be extended to leveraging information from other sources similar to the browser history, and for other purposes than the search engine. For example, re-ranking posts on social media such as Twitter and Facebook, modifying the program to run on mobile devices, using the LinkedIn information to construct a more effective user profile, etc. Also, our program was restricted by the computational costs. In future, with the improvement in GPUs and processors, computationally expensive methods can also be implemented, such as a complex, multi-layered deep learning model, or more SERPs can be scraped to allow for even better results.

REFERENCES

Altszyler E, Sigman M, and Slezak DF. 2016. "Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database". arXiv preprint arXiv:1610.01520.

Anikó Hannák, Piotr Sapieżyński, Arash Molavi Khaki, David Lazer, Alan Mislove, Christo Wilson. 2017. "Measuring Personalization of Web Search". arXiv preprint arXiv:1706.05011.

Kacem, A., Boughanem M. and Faiz, R. 2017. "Emphasizing temporal-based user profile modeling in the context of session search", Proceedings of the Symposium on Applied Computing - SAC '17.

Kumar, R. and Sharan, A. 2014. Personalized web search using browsing history and domain knowledge", 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT).

Makvana, K., Shah, P. and Shah, P. 2014. "A novel approach to personalize web search through user profiling and query reformulation", 2014 International Conference on Data Mining and Intelligent Computing (ICDMIC).

Renjini, L. and Ratheesh, T. 2016. "PSQCR — An efficient integrated approach for web search personalization", 2016 International Conference on Emerging Technological Trends (ICETT), 2016.

****