Meta Search Engine Optimization Using Genetic Algorithm

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

LOKESH NOWPADA	**121910319002**
JISHITHA BANDARU	**121910319005**
V ROHITH REDDY	**121910319017**
YERUVA GEETHA	**121910319030**

Under the esteemed guidance of

Dr.M.VENKATA RAMANA
ASSISTANT PROFESSOR, GST

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM

(Deemed to be University)

VISAKHAPATNAM

OCTOBER 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

We, hereby declare that the Project review entitled "Meta Search Engine Optimization Using Genetic Algorithm" is an original work done in the Department of Computer Science and Engineering, School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 31-03-2023.

Registration No:	Name:	Signature:
121910319002	Lokesh Nowpada	N. Lokesh.
121910319005	Jishitha Bandaru	B. Jishitha
121910319017	V.Rohith Reddy	V. Rohith Reddy
121910319030	Yeruva Geetha	Geetha

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM (Deemed to be University)

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "Optimization of Re-ranking Mechanism in Meta Search Engines" is a bonafide record of work carried out by Lokesh Nowpada (121910319002), Jishitha Bandaru (121910319005), V.Rohith Reddy (121910319017), Yeruva Geetha (121910319030) submitted in partial fulfillment of requirement for the award of degree of Bachelors ofTechnology in Computer Science and Engineering.

PROJECT GUIDE   **HEAD OF THE DEPARTMENT**

Dr.M.VENKATA RAMANA   **Dr.R.SIREESHA**

(ASSISTANT PROFESSOR)   **(PROFESSOR)**

CSE, GST   **CSE, GST**

GITAM   **GITAM**

ACKNOWLEDGEMENT

LOKESH NOWPADA   **121910319002**

JISHITHA BANDARU   **121910319005**

V.ROHITH REDDY   **121910319017**

YERUVA GEETHA   **121910319030**

TABLE OF CONTENTS

CONTENTS	PAGE NO.

LIST OF FIGURES

1. ABSTRACT

The World Wide Web is getting updated with innumerable web documents and the tasks for search engines to index these documents gets more and more complicated. Individual search engines have their own sophisticated indexing algorithms which index numerous records. Since the number of documents is vast and it is understandably difficult to always be accurate in indexing the web documents, it is not wrong to say that each search engine has its own uncovered range of documents. This hidden web is referred to as the deep web or hidden web. Whenever a user passes a query, the search engine turns towards its indexes to find the best result, but there is a possibility for the user's desired document to be lost in the hidden web with respect to that particular search engine. There is another disadvantage in the utilization of a single search engine which is that the SE's ranking algorithm could be biased to some records which might result in spammed results which are not very helpful to the user. To overcome these obstacles, meta-search engines could be used. They are the tools that pass the query to various search engines in order to cover most of the relevant web documents that can be returned to the user. But there are some barriers that are to be tamed in the implementation of meta-search engines. The merging of numerous URLs returned from various search engines should be ranked again to give an unbiased order of results that could really aid the user. This is important because it is more helpful for the users since they could save their time and effort by finding the desired web document at the top of the results list. In this paper, we demonstrate an optimal approach for re-ranking the web results.

KEYWORDS: MSE, Scraping, Result Merging, DOWA Operator, Genetic Algorithm.

2. INTRODUCTION

The individual search engines use their own ranking algorithms to index and rank the documents. Their mechanisms are very much distinctive and hidden to everyone. One of the cons in using an individual search engine is that the final ranked list might consist of biased decision making that is done to promote a certain document to the user. Meta search engine is a tool which can be utilized by the users in the same way any normal search engine is used. The query posed by the user is sent through various search engines and the results are gathered and ranked by using certain result merging techniques and are presented to the user so that the most relevant web documents appear at the top of the list.

There are a lot of algorithms and mechanisms that have been used for efficient result merging. While each method has a significant advantage, they also have a disadvantage that depends on the real time application. It is difficult for any result merging method to be perfectly applicable during all circumstances. So, our goal is to maximize the effectiveness as much as possible. To deliver the relevant results, meta-search engine's rank merging algorithms should be well optimized. In this paper, we discuss a system to re-rank the results in meta search engines.

This documentation is organized as follows: In Section 3, we discuss various research works that are related to our project. Section 4 depicts the architecture and the core algorithm of our meta search engine. Section 5 details the problem statement and the proposed solution. The methodology of the set of events that take place in the project are listed in Section 6. The initial requirement of gathering the web results of various search engines is discussed in Section 6.1. In Section 6.2 the mechanism to calculate positional ranks is described. The computation of semantic-based ranks is outlined in Section 6.3. In Section 6.4, the final steps of weight calculation and final rank computation are represented. In Section 8, the implementation of the proposed system is experimented and the results are calculated and the conclusion is made in Section 9.

3. RELATED WORKS

According to [1] various result merging mechanisms can be deployed to diminish the uncovered regions of the deep web. For merging different results from different sources, various ranking techniques are to be utilized to sort the web documents in such a way that the users are presented with the documents that are most relevant to them. The research done in [2] gives a view on the architectural insights of information retrieval in meta search engines.

The paper [3] exemplifies the implementation of a result merging algorithm by using two types of ranking mechanisms namely, linear search and semantic search and the final rank is achieved by the summation of them. Though this uses two distinct attributes to make the ranking, their simple addition does not always guarantee the optimal prioritization of both the mechanisms.

In the research [5], the authors illustrate a method of re-ranking by converting the problem into a linear programming model and solving it in a mathematical form. This calculation of optimum weights for each result generates an advantage when compared with pre-assigned weights. The drawback of this mathematical formulation is the assumption that each search engine has the same priority but in real-time situations there could be a necessity of giving more preference to one search engine than the other.

The utilization of WordNet as a knowledge base for finding the semantic similarity between words is explained in [6]. WordNet is a lexical database that consists of a vast number of words and their synonyms which can be used for text classification, summarization and so on. This database can be used as a knowledge base and the similarity between words can be generated. The study in [7] gives a review of the pros and cons of various kinds of measures that can be used to calculate the score of similarity between words.

The study in [8] demonstrates the application of the genetic algorithm to merge the results generated by various search engines where the set of web documents undergo fitness evaluation and then are crossed over and mutated to generate new solutions until the optimal solution is obtained. This is an iterative approach that occurs until the stopping condition is achieved.

The paper [10] evaluates the precision of various web search engines when encountered with multi-faceted queries. The search engine needs to pick the web documents which are most relevant to the user's requirement even if the query has diverse descriptions. This ability of various search engines has been evaluated and compared by the authors.

A gist of evaluation methods in TREC tracks is discussed in [12]. The authors discuss the pros and cons of various evaluation strategies to calculate the accuracy and precision of mechanisms related to information retrieval and question answering.

The TREC 2009 diversity task [13] aims to result in a ranked list of pages that gives multiple unique descriptions for a query in such a way that there is no redundancy. The web track consists of various queries and their unique sub definition.

The methods for improvement of accuracy in a meta-search engine has been discussed in [14]. The authors explain about the application of traditional methods for merging various documents of varying ranks. The advantages as well as the disadvantages were stated in the experimental analysis of those methods. Their analysis states that on using a better optimization mechanism, unpredictably accurate results of greater quality could be produced.

The research in [15] states the disadvantage of every search engine which is the ignoring of some web pages while implementing their indexing algorithms. This hidden web might consist of web documents that could be the result the user is browsing for. The authors put forward a new method of page rank algorithm and latent semantic indexing which help in indexing the hidden web for bringing the best results out of the web.

The authors of [16] explain about developing a web personalization system. Their study shows that the users navigational pattern which means the user's surfing approach is not totally sufficient to derive the user's behaviour and expectations. They propose a new web personalization system that takes in the timing and semantic data in addition to the user's surfing pattern in order to classify the user's interests and show them relevant recommendations. Their system results in a propitious web personalization model that uses real as well as synthetic data.

In the study of [17], they propose a new extension scheme to extract the suggestions from various sources which they referred to as Meta-suggestions. The Meta-suggestions are aggregated from the various suggestions given by the numerous search engines by means of a plugin which is similar to meta-search. However, they stated the limitations with respect to the performance of the individual target engines. Nevertheless, their model gives interesting meta-suggestions that seem promising.

4. PROBLEM IDENTIFICATION AND OBJECTIVES

4.1 The Problem

A meta search engine works by passing a query to numerous search engines at the same time. Then, the results are merged and displayed to the user. In this process of merging the vast number of results, ranking needs to be done to sort the results and produce the list in such a way that the results that are desired by the user are found at the top of the list. Re-ranking of the results obtained from the source websites is a crucial task in the implementation of the meta search engine. Proper ranking scheme for delivering the user the documents which they are more probably looking for at the top of the page is subjective. The ranking scheme or mechanism needs to be optimized for getting better results.

4.2 The Proposed Solution

In the proposed system, we make use of two criteria to re-rank the results. The first criteria is to consider the position of the document in that particular search engine's result which determines the probable rank given by that search engine's algorithm. By calculating the rank based on the document's position, we can derive the priority given by that search engine for every web document. This helps in not disregarding the search engine's individual opinion of the priority of that particular document. Ranking based on position could be done by considering the positioning of that web document URL in that particular search engine result and the number of results produced by that search engine. This rank is directly proportional to the priority given by that search engine for that web document. In other words, the document at the top of the list has more rank and the one at the bottom has reasonable lesser rank.

The second criteria is to calculate the measure of meaning that the web document gives that is more relevant to the search query. For this reason, we need a lexical database that consists of numerous words and their meanings and synonyms. By crawling the whole web document, we can find the similarity between individual words and the search query, thus adding to the document's score based on semantics. From this score, the rank of that web document can be determined with respect to semantics.

In the end, we equalize the priority of both of the criteria by calculating the weights for them to avoid the loss of priority given to any of them and then add them to get a resulting rank which is the final score that determines the worth of the document, hence ending the proposed re-ranking mechanism.

5. SYSTEM DESIGN

5.1 Proposed System Architecture

In this section, we describe the process of implementing the proposed re-ranking mechanism to optimize the order of the documents resulting from various search engines to provide a better surfing experience for the user. The goal is to assist the user to receive the most relevant document from the web. This is possible by passing the user's query to numerous search engines concurrently and fetching the resulting documents and then applying our approach to re-rank the results order and handing them over to the user in the decreasing order of their ranks. The entire architecture is depicted in Figure 1.
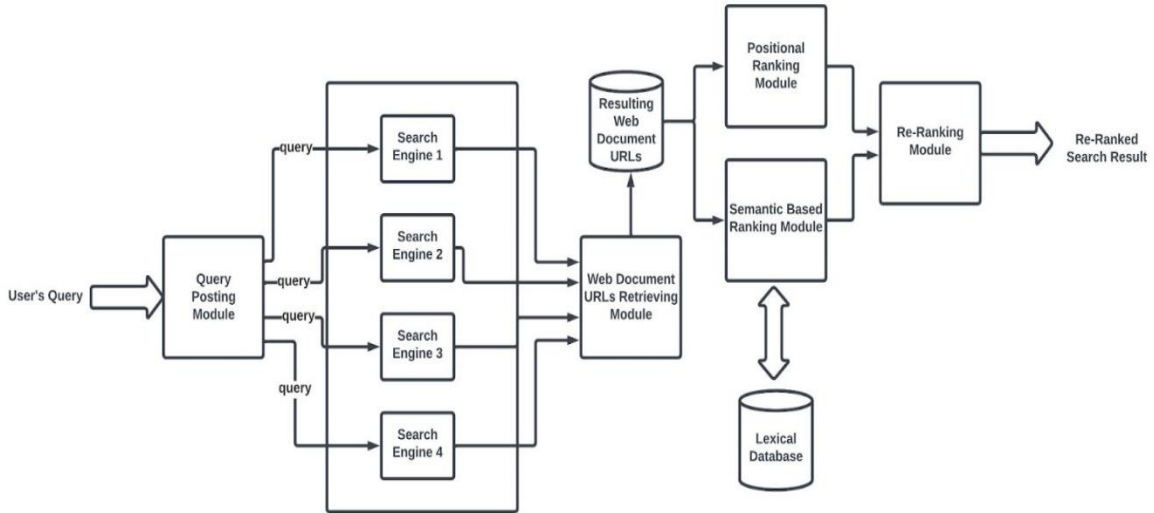

Figure 1: Proposed System Architecture

Firstly, the user just needs to pass the search term through the simple interface that is provided, and the term is implicitly passed as a query to the Query Posting Module. The task of the Query Posting Module is to pass the search term inputted by the user to several search engines in the format that is specific to that particular search engine. The web documents resulted by the search engines are handed over by the Web Document URLs Retrieving Module. The resulting URLs from the URLs Retrieving Module are stored in the system's list. The list of the resulting web documents is passed to the Positional Ranking module, which calculates the rank of the web document based on the position of it in that particular search engine's result. This ranking reflects the essence of the indexing methods used by the individual search engines and hence it is a crucial ranking criteria.   Simultaneously the retrieved web documents are passed to Semantic-based Ranking Module where the computation of the rank of individual web documents is done by the measure of the presence of the words that are synonyms of the user's query. In other words the

similarity of the semantics of the contents of the web document with that of the search term is used to calculate the rank based on the semantics. For the calculation of this similarity score, a lexical database called WordNet is used that has a huge number of words and their synonyms.

Finally, the ranked lists from the SESRC Module and Semantic-based Ranking Module are passed to the Re-ranking Module where the ranks are combined based on the priority of the criteria to ultimately derive the final ranks of the web documents that are inversely proportional to their position in the resultant. This means that the URL with the highest rank is at the top of the resulting re-ranked list.

5.2 Algorithm

Input: User's query, Output: Re-ranked list

```
i := 1;

st := search term;

while i<SEₙ do

    st is passed as a query to SEᵢ;

    SRᵢ := search results of SEᵢ;

end

foreach SRᵢ do

    foreach Dⱼ in SRᵢ do

        LRDi,j := rank assigned by the specific SEᵢ
    end
end

foreach SRᵢ do

    foreach Dⱼ in SRᵢ do

        SeRDi,j := similarity score of the document
words wrt st
    end
end

for each RDi,j do

    RDi,j := Weighted_Sum(LRDi,j , SeRDi,j)
end
```

Let SE_i be the ith search engine used by the system whose results are stored in SR_i. Each D_j determines the jth web document URL in that Search Engine. Compute the rank of the particular URL based on its position in the SE and store it in LRDi,j which reflects the rank assigned by the specific SE. Similarly, for every web document, check for the synonyms for the user's search term 'st' and calculate the measure of similarity found in that document from which ranking is done and the semantically ranked list is stored in SeRDi,j. The ultimate re-ranked list, 'RDi,j' can be calculated by the summation of LRDi,j and SeRDi,j.

This procedure helps in bringing efficient results because here, we consider the ranking done by the search engine's indexing algorithms and the semantics of the web document. Hence, it helps in optimizing the resultant.

6. METHODOLOGY

In this section, we discuss the various prerequisites and technical requirements to implement this system. The initial requirement is to design search engine specific scrapers to pass the user query and fetch the resulting web documents.

6.1 Search Engine Scraping

The initial requirement is to design search engine specific scrapers to pass the user query and fetch the resulting web documents. For this step, we have made use of a python library called [4] "ScrapeSearchEngine" which provides scraping services to scrape various search engines like Google, Yahoo, DuckDuckGo, Bing and so on. This library provides functions that are specific to each search engine. The parameters that are to be passed are the search term which has been given by the user and the user agent that the system's web browser uses. The search results that are produced by the search engine are returned by the function as an ordered list.

6.2 Positional Ranking

The results that are in the form of a list are now calculated for their positioning based on the search engine's result. The ranking that is given by the search engine can be assumed by considering the position of the web document URL in the search engine's result. We consider it by assigning the highest rank to the record at the top of the result and assigning the lowest rank to the record at the bottom of the result. The implementation can be done by subtracting the position of the URL from the number of results. This is depicted by the below pseudo code in Figure 2.

```
foreach SRᵢ do
        foreach Dⱼ in SRᵢ do
                LR_Di,j := (size-position+1)
        end
end
```

where, size = min(size(SR_1),size(SR_2),… size(SR_n)).

Figure 2: Positional Ranking Pseudo Code

6.3 Ranking Based on Semantics

This ranking method involves crawling the whole web document in the search for words that are similar to the user's query. By calculating the similarity scores of all the words, we can determine the possibility that a particular web document is the one that the user is looking for. To implement this crawling of the web documents, we have made use of the python library "BeautifulSoup" that helps in extracting the source code of a web document which can later be dissected for the required text with basic programming. After receiving the extracted components of the web documents, they can be searched word-by-word and the similarity scores of the web document can be calculated. We have used the WordNet database which is a huge lexical database consisting of numerous synonyms. The python library, nltk provides the wordnet class that consists of many methods that can connect with the lexical database and compute the similarity scores. The corresponding pseudo code is present in Figure 3.

```
foreach SRᵢ do
        foreach Dⱼ in SRᵢ do
                foreach word in Dⱼ do
                        score := similarity_score(word,query)
                        if score>0 do
                                SeRᴅi,j := SeRᴅi,j + score
                        end if
                end
        end
end
```

Figure 3: Semantic-based Ranking Pseudo Code

6.4 Weight Calculation using Genetic Algorithm

In this final step, we combine the ranks delivered by both the ranking methods. To even-out the priority of both the criteria, we will find out the weights with which the scores can be multiplied. The application of Genetic Algorithm for finding these weights is presented in Figure 4.
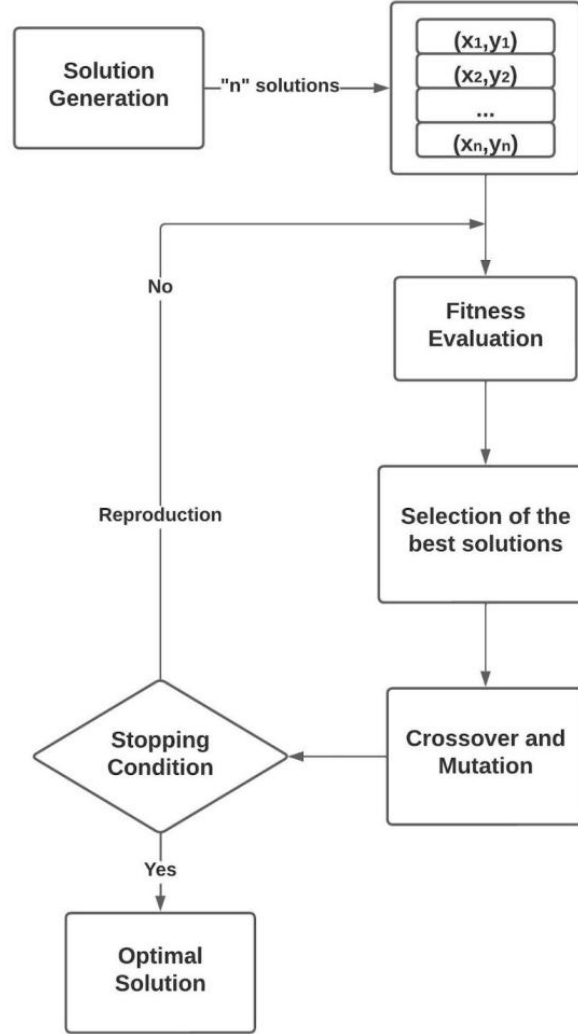

Figure 4: Flow of Events in Genetic Algorithm

First, we generate a huge number (10000) of solutions using a random object. Here, the solution is a 2-element tuple.

$$L_i * x + S_i * y; \ x,y \ \text{are the weights;}$$

L and S are the positional and semantic based ranks of the ith document.

Calculate the fitness value for this set of randomly generated solutions. The fitness function in this experiment is formulated by using DOWA (Dependent OWA) operator [9]. The pseudo code for the fitness function is present in Figure 5.

```
function fitness_eval(list,L,S,x,y)
    arg_vector = []
    foreach D_i in list do
        arg_vector.add(^LD_i*x + ^SD_i*y)
    μ = sum(arg_vector)/length(arg_vector)
    deviation_mean = []
    foreach arg_i in arg_vector do
        deviation_mean.add(|arg_i-μ|)
    similarity = []
    foreach dev_i in deviation_mean do
        similarity.add(1-(dev_i/sum(deviation_mean)))
    dowa=0
    for(i=1;i<=n;i++) do
        dowa += (similarity_i/sum(similarity)) * arg_vector_i
    if deviation_mean.matchesCriterian() do
        fitness = dowa
    else do
        fitness = α
    return fitness
```

Figure 5: Weights Calculation Pseudo Code

where, **(x,y)** - solution tuple ;

       list - list of documents

       L - list of Positional ranks

       S - list of Semantic based ranks

       D_i - i^{th} Document in the list

matchesCriteria() - specific function that checks the predicted optimality w.r.t. the deviation means generated in that iteration;

α - Max fitness value

After the calculation of the fitness value, parent selection takes place where the best solutions in that iteration are selected. Check whether the largest value among them crosses the threshold and stop the loop for optimization. From the best solutions, perform crossover and mutation and generate new solutions. Repeat the above steps with these newly generated solutions until the best solution is found.

On finding the best solution, the final rank for each web document can be determined by substituting the (x,y) tuple into the equation : $L_i * x + S_i * y$

Based on these ranks, the web documents can be arranged in decreasing order and can be presented on the user interface.

7. EXPERIMENTAL RESULTS

For measuring the veracity of this algorithm in real-time situations, we pass various queries and use the obtained results to make the calculation. To achieve that, we use query sets from the TREC dataset [11]. In this dataset, every item consists of a query and its description that guides us to find whether a obtained web document is relevant or not. For example, a record in the TREC dataset is in the displayed in Figure 6.

```
▼<topic number="8" type="ambiguous">
  <query>appraisals</query>
  <description>How are home values appraised? I want to know how home appraisals are done. </description>
  <subtopic number="1" type="inf"> What companies can give an appraisal of my home's value? </subtopic>
  <subtopic number="2" type="inf"> I'm looking for companies that appraise jewelry. </subtopic>
  <subtopic number="3" type="inf"> Find examples of employee performance appraisals. </subtopic>
  <subtopic number="4" type="nav"> I'm looking for web sites that do antique appraisals. </subtopic>
 </topic>
```

Figure 6: A record in the TREC dataset

The queries are classified into two types namely, faceted and ambiguous. Ambiguous queries are the queries which might hold a different meaning during distinct situations. On running these queries in our meta search engine ranking algorithm, and marking the web documents based on their relevance, we calculate the TREC Style Average Precision (TSAP) which can be compared with that of the result given by simple summation.

Initially the relevance of the web document with respect to the query must be checked by using the dataset. If the web document is relevant, the reliability (r_i) of the web document can be calculated as reciprocal of its rank. Lastly the TSAP@N can be calculated by the formula specified in Figure 7.

$$TSAP@N = \frac{(\sum_{i=1}^{N} r_i)}{N}$$

where, r_i is the reliability of the ith web document

Figure 7: TSAP@N Calculation

On calculating the Average TSAP@N of the novel approach [3] and proposed approach, the obtained results reflect the better precision score of the proposed system. The results are presented in Figure 8. The Average TSAP@N values have been calculated by adding all the TSAP@10 values obtained from the passing of each query and dividing them by the number of queries. The value of N is 10 because the top 10 results for each query have been considered. Here, the dataset contains 50 queries. The average value of TSAP@N taken over passing 50 queries for proposed system is about 0.2885 which is higher than that of the novel approach method [3] by 5.91%.
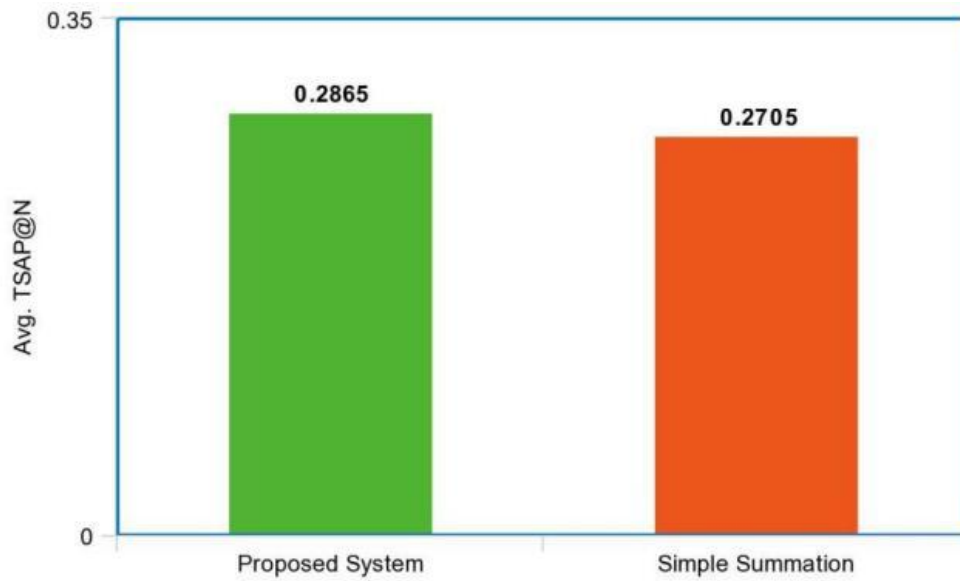

Figure 8: Proposed Approach vs Simple Summation

The comparison of the proposed approach with that of Simple summation is displayed in Figure 9. The proposed system shows an average TSAP@N value of about 0.2881 when computed for 38 queries which exceeds the value of simple summation by 3.44%. For ambiguous queries, the average TSAP@N for the proposed system is 0.2814 which leads the value of simple summation by 14.76%. The results favor the weighted summation approach when compared to the simple summation method.

	Average TSAP@N values	
	Faceted Queries (38 queries)	Ambiguous Queries (12 queries)
Proposed System	0.2881819131	0.2814748677
Simple Summation	0.2785735171	0.2452248677

Figure 9: Faceted and Ambiguous Queries

8. CONCLUSION

The experimental findings summarize that the proposed approach re-ranks the web documents with much more precision than that of simple summation. On computation of weighted summation, we try to give equivalent priority to both positional ranking as well as semantic-based ranking. Positional ranking reflects the order of preference that is given by each search engine. Whereas Semantic-based ranking considers the overall meaning of the web document which is analogous to the user query. We try to give priority to the search engine's preference as well as the affinity of the web document's meaning to the query description by performing weighted summation of both the rankings.

To deliver better results, more parameters can be considered to make a more effective decision that brings much more attributes into the picture. This helps in exposing the diverse contents of the hidden web to the users and providing them with their desired web document.

9. REFERENCES:

[1] Lu Y, Meng W, Shu L, Yu C, Liu KL. Evaluation of result merging strategies for metasearch engines. InWeb Information Systems Engineering–WISE 2005: 6th International Conference on Web Information Systems Engineering, New York, NY, USA, November 20-22, 2005. Proceedings 6 2005 (pp. 53-66). Springer Berlin Heidelberg.

[2] Madhavi A, Chari KH. Architecture based study of search engines and meta search for information retrieval. Int. J. Eng. Res. Technol. (IJERT). 2013.

[3] Siji Rani S, Goutham S. A novel approach for meta-search engine optimization. InAmbient Communications and Computer Systems: RACCCS-2018 2019 (pp. 377-386). Springer Singapore.

[4] ScrapeSearchEngine library https://pypi.org/project/scrape-search-engine/

[5] Amin GR, Emrouznejad A. Optimizing search engines results using linear programming. Expert Systems with Applications. 2011 Sep 1;38(9):11534-7.

[6] Richardson R, Smeaton A, Murphy J. Using WordNet as a knowledge base for measuring semantic similarity between words.

[7] Meng L, Huang R, Gu J. A review of semantic similarity measures in wordnet. International Journal of Hybrid Information Technology. 2013 Jan;6(1):1-2.

[8] Kumar J, Kumar R, Dixit M. Result merging in meta-search engine using genetic algorithm. InInternational Conference on Computing, Communication & Automation 2015 May 15 (pp. 299-303). IEEE.

[9] Boongoen T, Shen Q. Clus-DOWA: A new dependent OWA operator. In2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence) 2008 Jun 1 (pp. 1057-1063). IEEE.

[10] Wu S, Zhang Z, Xu C. Evaluating the effectiveness of Web search engines on results diversification. Information Research: An International Electronic Journal. 2019 Mar;24(1): n1.

[11] 2009 TREC Query Set - https://trec.nist.gov/data/web/09/wt09.topics.full.xml

[12] Teufel S. An overview of evaluation methods in TREC ad hoc information retrieval and TREC question answering. Evaluation of text and speech systems. 2007:163-86.

[13] Clarke CL, Craswell N, Soboroff I. Overview of the trec 2009 web track. WATERLOO UNIV (ONTARIO); 2009 Nov 1.

[14] Fu-yong, Y., Jin-dong, W.: An implemented rank merging algorithm for meta search engine. In: International Conference on Research Challenges in Computer Science (2009)

[15] Pavani, K., Sajeev, G.P.: A novel web crawling method for vertical search engines. In: 2017 International Conference Advances in Computing, Communications and Informatics (ICACCI) (2017)

[16] Sajeev, G.P., Ramya, P.T.: Effective web personalization system based on time and semantic relatedness, Conference on Advances in Computing, Communications and Informatics (ICACCI), Sept. 21–24, 2016 (2016)

[17] SEUNGMIN KIM, EUNCHAN NA, AND SEONG BAEG KIM : Developing a Meta-Suggestion Engine for Search Queries, IEEE Access, 2022 (2022)