International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 3 March 2023, pp: 1714-1719 www.ijaem.net ISSN: 2395-5252

Meta Search Engine Optimization Using Genetic Algorithm

Lokesh Nowpada[1], Jishitha Bandaru[2], Vemireddy Rohith Reddy[3], Yeruva Geeta[4], Dr. Mancha Venkata Ramana[5]
[1,2,3,4]Student; [5]Assistant Professor, Department of Computer Science and Engineering, GITAM (Deemed to be University), Visakhapatnam, Andhra Pradesh

Date of Submission: 20-03-2023 Date of Acceptance: 30-03-2023

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.

I. 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 research documentation is organized as follows: Section 2 discusses various research works that are related to our paper. Section 3 depicts the architecture as well as mechanisms of our meta search engine. The initial requirement of gathering the web results of various search engines is discussed in Section III.I. In Section III.II the mechanism to calculate positional ranks is described. The computation of semantic-based ranks is outlined in Section III.III. In Section III.IV, the final steps of weight calculation and final rank computation are represented.

II. LITERATURE SURVEY

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.

III. PROPOSED SYSTEM DESIGN

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.

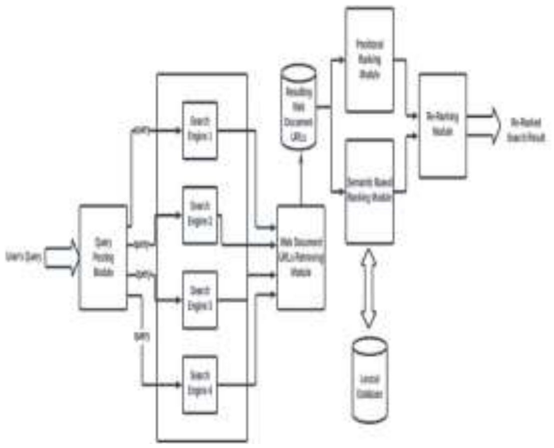


Figure 1: MSE Architecture

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.

III.I 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.

III.II 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 $SR_i$ do
    foreach $D_j$ in $SR_i$ 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

III.III 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 $SR_i$ do
    foreach $D_j$ in $SR_i$ do
        foreach word in $D_j$ do
            score := similarity_score(word,query)
            if score>0 do
                $SeR_{Di,j}$ := $SeR_{Di,j}$ + score
            end if
        end
    end
end
```

Figure3: Semantic-based Ranking Pseudo Code

III.IV 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.

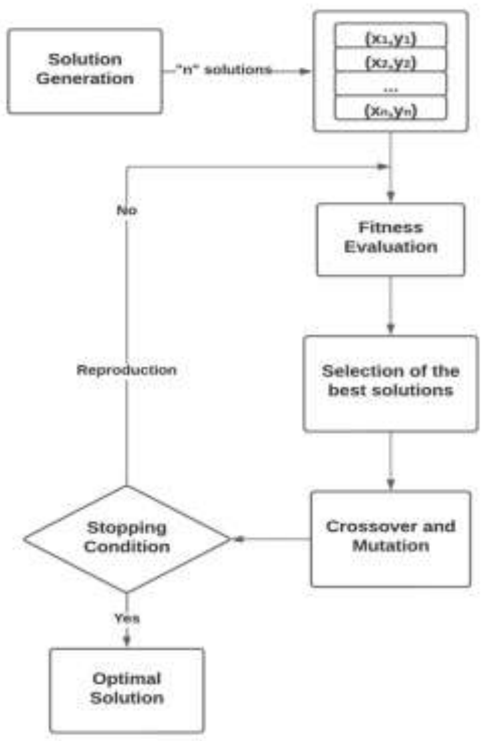
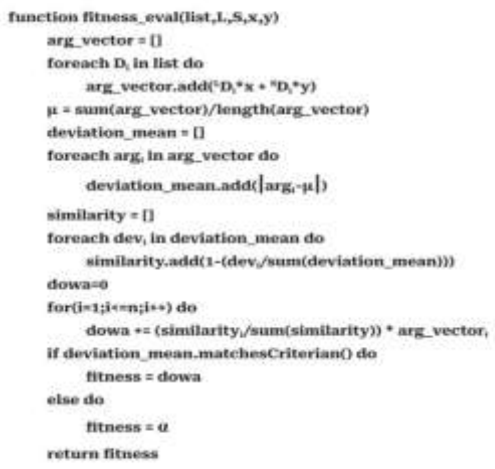


Figure 4: Flow of Events in Genetic Algorithm

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 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(^L D_i*x + ^S D_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
```

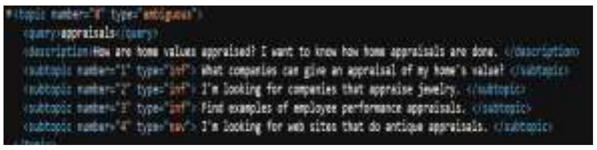
Figure 5: Weights Calculation Pseudo Code

where, **(x,y)** - solution tuple ;
list - list of documents
L - list of Positional ranks
S - list of Semantic based ranks
D_i- ith 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.

IV. 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="[illegible]" 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>
```

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 i^{th} 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.

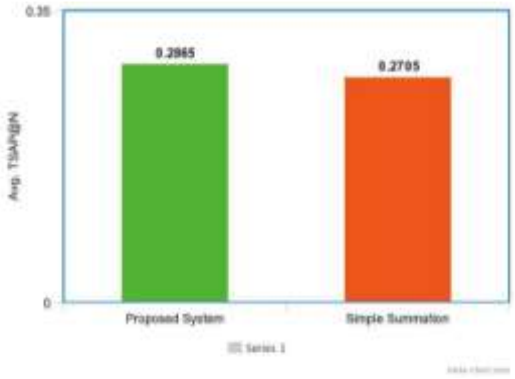
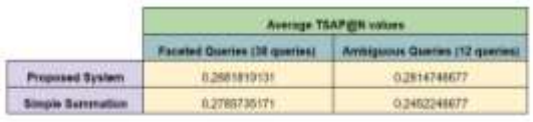


Figure 8: Proposed Approach vs Simple Summation

The comparison of the proposed approach with that of Simple summation is displayed in Figure 9. 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	0.2981819131	0.2814748677
Simple Summation	0.2788735171	0.2452248677

Figure 9: Faceted and Ambiguous Queries

V. 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.

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.