https://doi.org/10.31449/inf.v43i2.2132 Informatica **43** (2019) 235–241 **235**

New Re-Ranking Approach in Merging Search Results

Vo Trung Hung
University of Technology and Education - The University of Danang
48 Cao Thang, Danang, Vietnam
E-mail: vthung@ute.udn.vn, http://www.udn.vn/english

Keywords: search engine, algorithm, merging, re-ranking, search result list

Received: January 3, 2018

When merging query results from various information sources or from different search engines, popular methods based on available documents scores or on order ranks in returned lists, its can ensure fast response, but results are often inconsistent. Another approach is downloading contents of top documents for re-indexing and re-ranking to create final ranked result list. This method guarantees better quality but is resource-consuming. In this paper, we compare two methods of merging search results: a) applying formulas to re-evaluate document based on different combinations of returned order ranks, documents titles and snippets; b) Top-Down Re-ranking algorithm (TDR) gradually downloads, calculates scores and adds top documents from each source into the final list. We propose also a new way to re-rank search results based on genetic programming and re-ranking learning. Experimental result shows that the proposed method is better than traditional methods in terms of both quality and time.

Povzetek: V prispevkih sta primerjana dva pristopa pri združevanju zadetkov iskanja: z enačbo in z algoritmom TDR, nato pa je primerjana še izvirna metoda.

1 Introduction

In the Internet, search engines like Google, Bing, Yahoo provide a convenient mechanism for users to search and exploit information on the Web. According to statistics of "Surface Web" in 2017[1], it shows that Google indexes about 50 billion web pages, Bing about 5 billion pages.

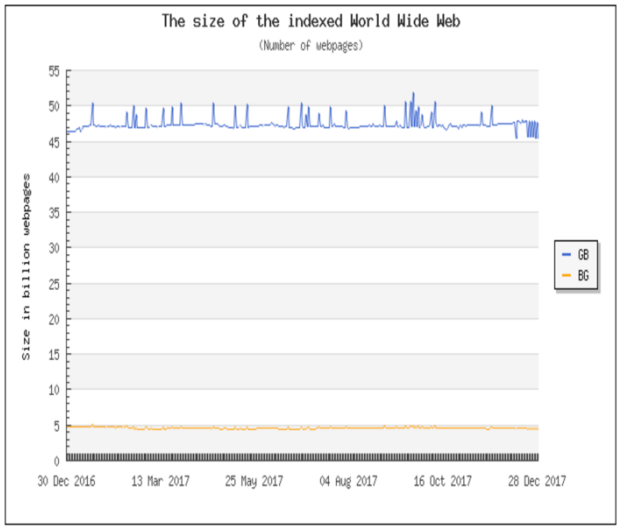

Figure 1: Size of the indexed webpages.

The "Surface Web" is only about 1% of the "Deep Web" - which is not indexed by popular search engines. Many websites do not allow search engines to crawl, instead offering themselves a separate query system such as PubMed or the US Census Bureau.

However, when searching on search engines such as Google, Yahoo or Bing users are not satisfied for two reasons. Firstly, each search engine has different corpus, searching and ranking methods so the returned results will be different. Secondly, search engines now perform monolingual searches (search only on the corresponding language for search keywords), so users can not find webpages in other languages.

To help users exploit the information effectively, there are some tools that combine search results from various sources. We can improve search results based on the available search engines by building a Meta Search Engines [1]. The nature of Meta Search Engines is to use techniques to exploit existing search engines and to process the results obtained from these search engines to generate a new search result that better matches user requirements. A Meta Search Engine needs to handle a variety of issues such as query processing, search on available search engines, processing returned results, re-ranking results found, and display results for users. In this study, we focused solely on re-ranking the results found by the search engines available.

There are two approaches to solve the problem. The first is to mix the search results (duplicate documents) of different search engines on the same information space. This method is often applied to "Surface Web". The second is to combine search results from independent sources (Federated Information Retrieval - FIR) [2], more in line with the exploitation of "Deep Web" information.

The research and development of a combination of search results from multiple sources focused on three main

[1] http://www.worldwidewebsize.com

issues: server description, server selection, and merging [1]. Server description is intended to estimate general information about the original search server such as the number of documents, terms; Frequency of search results returned, ... Server selection is made based on the server description information to determine the most suitable server to send the query. Mixed results are the main work of combining search results from multiple sources, evaluating, rearranging documents, creating final list of results returned to the user.

Merging techniques can be distinguished based on the types of information used for evaluating, re-ranking search results from sources [3]: server information search (total number of documents, results returned); Statistical information: the rank order of the document, the rating provided by the originator; basic information (title, abstract); or the content of the document itself. Research is aimed at improving the evaluation criteria such as accuracy, recall, data usage savings, response speed and bandwidth usage.

The innovation in this paper is using machine learning techniques and basic information returned from the original search engine for re-ranking. We propose solution of sequential mixing to balance the speed and quality of the results.

The rest of this paper is organized as follows. In the Session 2, we present an overview of re-ranking and focus on previous efforts on techniques of re-ranking as well as our analysis and remarks on pervious methods. Details of our proposal in using genetic programming for the re-ranking are presented in Section 3 and the experiment is presented in Session 4. We conclude important points in Section 5.

2 Overview on re-ranking

2.1 Ranking and re-ranking

In the information query, the ranking is usually done by calculating the score of fit between the document and the query, serving the goal of creating a list of documents in decreasing order of the score (shows the degree of suitability for user requirements).

After executing the initial query and receiving the results from a search engine, the data can be extracted including the query content itself, the text list, the ranking points corresponding to the text (some may be hidden from the user), some basic content for each text, such as title, abstract. On an interactive system, the search is performed repeatedly, and the system can store and analyse the contents of executed queries, found documents, read texts, declarations or manipulations by users. The above information may be exploited by the system to re-rank the result list in a variety of ways, distinguished by the type of data used as using the information of the available search engines, rating, or considering to user information.

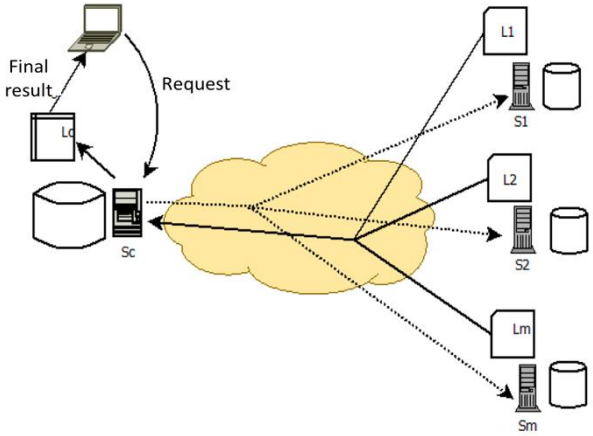

Figure 2: Mix model for search results.

Merging search results from multiple sources has the following process (Figure 2): The central server S_c receives the query from the user, sends the query to search servers from S_1 to S_m. From each S_i server, the list of L_i contains N best results created and returned to the central server. S_c re-evaluates the documents based on the content returned from the original search servers or the content themselves to create the final result list returned to the user.

2.2 Techniques of re-ranking

2.2.1 Combination available rating

The simplest method to merge ranking results is Raw-Score, which directly uses the rankings in each of the original search result listings [4]. The CombSUM method proposed by Fox and Shaw, takes the total score of the document in the various search engines to determine the CombSUM score for a document.

$$CombSUM = \sum_{i \in IR\ Servers} score_i$$

with IR Servers as the set of search engines, $score_i$ is the point of the document assigned by the i^{th} search engine.

The score assigned by a search engine can be normalized to a NormalizedScore score to avoid differences in searcher norms:

$$NormalizedScore = \frac{score - MinScore}{MaxScore - MinScore}$$

with *MinScore* and *MaxScore* being the smallest and largest values in the score of all documents assigned by the search engine.

The weakness of this method is the difference of search engines quality on ranking quality, scoring, presentation methods, ... To overcome the limitation, we can add a weighting for search engines. The WeightedCombSUM score for a document is calculated by the formula:

$$WeightedCombSUM = \sum_{i \in IR\ Servers} w_i \times NormalizedScore_i$$

Here, w_i is the weight assigned to the search engine i in the set of search engines IR Servers; $NormalizedScore_i$ is the normalization of being assigned by server i to the document as in the formula of *NormalizedScore.*

Similarly, some studies [5] suggest a linear function combining the ratings of search engines of the form:

$$M(d,q) = \sum_{i=1}^{n} \beta_i \times s_i(d,q)$$

Here $M(d, q)$ is the final ranking point, $s_i(d, q)$ is the ranking (normalized) of the search engine i, β_i is the weight assigned to the search engine i. The limitation of these methods lies in the need to identify values β_i by manual methods or based on observation of training data.

2.2.2 Ranking order information

The second solutions group uses ranking order information in the original search list. The Round Robin method [6] is the simplest method of mixing, which is performed as follows: We have the result list which is returned from L_1, L_2, ..., L_m; Firstly, we get the m first result as R_1 from the list of L_i, then take the m second result is R_2 from the list of L_i and so on. The final result of the mixing process is in the form of L_{1R1}, ..., L_{mR1}, L_{1R2}, ..., L_{mR2}, ... This is the right solution to ensure search speed when the source of quality information equivalent.

Borda mixing method [7] uses expert judgment scores. Each expert ranked a number of *c* documents. For each expert, the top document is *c*, the second document is *c-1* and so on. If there are some unrated documents, the remainder is divided equally among all unrated papers. Finally, the materials are ranked according to the total number of points assigned. Blending methods use useful ranking information in the absence of information about the search engine rankings. However, studies show that this method of mixing is not as effective as the combination of scores.

The LMS method (using result Length to calculate Merging Score) introduces the original search server counting formula based on the number of returned documents, then identifies new points for documents by multiplying the server point by original point [8].

2.2.3 Ranking learning

In a local search system, documents can be indexed in a variety of ways such as VSM, LSI, LMIR, ... The score of a document versus a query in different ways can be considered as different attributes of the document. Current information query systems tend to apply machine learning techniques to model or create ranking formulas based on these attributes.

The learning process consists of two steps: training and testing. The training input is *D* consisting of the set *{<q, d, r>}*, where *q* is the query, *d* is the document represented by the list of attributes $\{f_1, f_2, ..., f_m\}$, *r* is the relevancy of the document *d* versus the query *q*. The training step involves the construction of an *F* rating model, based on a training database that determines the relationship between the attributes of the document and the relevance of the document to the query. At the test step, the ranking model applied to the *T-dataset* is made up of the set $\{<q_{test}, d_{test}, r_{test}>\}$, the $r_{predict}$ value is the d_{test} document relevancy for the q_{test} query. - calculated by the *F-rating* model - will be compared to the r_{test} value for the rating quality of the rating model. Data for training *D* and experimental data *T* are usually generated by editing the search results in practice, and then manually evaluated by experts.

Ranking methods generally have the same approach by optimizing the objective function: find the maximum value of the gain function or find the minimum value of the loss function.

Ranking techniques are divided into three groups: *point-wise*, *pair-wise* and *list-wise* [9]. With a *point-wise* approach, each training object corresponds to an assigned document attached to the rating value. The learning process involves finding a model that maps each object to a rating close to its actual value. The *pair-wise* approach utilizes pairs of documents that are associated with rank order (before or after) as training subjects. In the *list-wise* approach, the training object is itself the list of ranked documents corresponding to the query.

The characteristic of the *point-wise* solution group is *PRank* introduced in [10] using a regression analysis.

In the *pair-wise* group, they constructed the *RankSVM* ranking algorithm with the aim of minimizing bias in the list of sorted pairs. This method is often referred to in studies as a basis for comparison. Freund applies boosting and introduces the *RankBoost* algorithm [11]. The advantage of this approach is that it is easy to deploy and can run in parallel for testing. Another example is *FRank* based on the probability ranking model.

In the *ListNet* method of the *list-wise* group, the document list itself is considered a training subject. The authors use a probability method to calculate the loss function for the list, which is determined by the difference between the expected sorting list and the correct sorting list. Neural network models and gradient descent are used in deployment algorithms to determine the ranking model.

While the presented methods may apply to mixing results from multiple search engines, the ranking learning methods apply to the case of the search system. Kits and documents are indexed in different ways. According to Yu-Ting and colleagues [12], ranking methods with training data (referred to as supervised ranking) were evaluated more effectively than others one (may be considered non-supervisor ranking).

2.2.4 Using user information

By default, traditional web search engines perform keyword-based queries. However, two different users, with different interests, can use same keywords with different search goals. In order to better meet the individual user's search needs, the user's declaration of behaviour and habits of the user during the search operation has become a research object. personalized ranking results or cooperative ratings [13].

Personalization of rankings results in querying and ranking results for users based on individual user interests and is carried out through two processes: (1) The

information that describes the user's interest and (2) the data collection reasoning to predict the content is close to the user's desires.

Initial data collection solutions require the user to disclose the information interest through the registration table, and the user may change this information [14]. The problem with this solution is that the user does not want to, or has difficulty in providing feedback about their search results as well as their concerns. Another direction, more popular, perform "learning", create user profiles through search history to classify, create groups of topics of interest to users with the aim of providing more information for the ranking. Based on the collected data, the authors build a model that describes and exploits relationships between users, queries, and Web pages, and serves search results matching the needs of user. In terms of characteristics, models may be limited to the exploitation of "two-way data" that exploits the user's interest in information topics, or "data in three directions" (three-way data) incorporates more information about the site.

In addition to the user-identified information solution, a number of solutions for exploiting user group information, created through the analysis of the already-searched content of the set User groups have the same characteristics (geographic location, occupation, interest) or have common search habits, such as Collaborative Filtering (CF). Web sites that meet a person's profile will be considered appropriate for others in the same group.

Due to the sparseness of the data sparsity, the latent semantic indexing algorithm is widely used as the primary technique for data modelling to optimize the layout as well as volume calculation [15].

2.3 Remarks

In the re-evaluation methods based on the rating of the original search engines, raw-score is the simplest method, which will compare directly the origin of the documents to the final result list. *CombSUM* is taking the total score of the document in the various search engines to determine the ranking in the final list. This score is standardized to avoid differences in the norms of each search engine, or to supplement the corresponding original server quality parameters in the Weighted *CombSUM*.

The second solutions group uses ranking order information in the original search list. This is the right solution to ensure search speed when the source of quality information equivalent.

The third solutions group uses the basic information (such as headings, excerpts, ...) of the original results in the scoring of documents. It compares the query with the title or footnote of the document, then applies the scoring formula based on ranking factors, title points, point lengths, lengths of title, and excerpts. In the news search system "News MetaSearcher" [16], in addition to the above factors, the time to update the document is also included in the rating formula.

The fourth solutions group performs the loading of the entire contents of the documents present in the original search result listings, then uses the indexing and scoring mechanism at the central server to perform the sorting, re-ordering the materials. It reviews the entire document to ensure a stable end result list, but takes a lot of time and bandwidth to load data from multiple servers.

The methods in the two first groups rely on the statistical information returned from the query (score, rank order) to perform calculations, so ensure a quick response to the final ranking result. However, some of the factors that make the quality of the endorsements are not good: Firstly, the search engines have large differences in data size, ranking algorithms that make the scoring formula based only on statistical information is not really relevant; Second, in reality the search server usually does not provide information about the document review point.

The third solutions group is usually chosen in practice because of its advantages in both speed and search quality compared to the two first groups. The final solution group has a stable ranking quality, but requires a lot of time for downloading the full content of the candidate materials as well as computational time for indexing and re-rating.

From here the requirement for a solution is guaranteed to make the most out of the basic information from the return lists, on the other hand requires the content of the documents in the final list to be consistent with the query and satisfactoriness on time and bandwidth costs.

3 Proposal solution

3.1 Idea

We propose a new solution to re-rank search results in using genetic programming.

Genetic Programming (GP) was first introduced by Angeline [17], based on genetic algorithms. In GP, each potential solution as a function is called an individual in the population set. GPs operate through the loop mechanism: at each generation, the dominant individual selectivity in the population is based on the content of the price; Perform hybrid, mutant, and spawn operations to create better individuals for later generations.

From randomness and irrelevance to the algorithmic principle of individual formation, in many cases genetic programming helps to overcome localized optimization errors. Although there is no assurance that the results identified by genetic programming are optimal, experimentation in different areas indicates that this result is generally better than the application of algorithms defined by the expert, in many cases, this result is close to the optimal solution [17].

An important element in the implementation of genetic programming is the definition of the individual, on the basis of which the content is determined, ensuring that the measurement accurately determines the quality of the solution. In addition, the complexity of the content, the number of individuals in the population, the rate of hybridization and mutation, the number of generations to be tested should be well defined to balance the ability to create a good solution, eliminate solutions that are not suitable for the calculation volume and time to solve the problem.

Previously, the practice of ranking methods was conducted independently, on different sets of data. This does not allow comparison of methods and hinders research. In 2007, Microsoft introduced the LETOR (LEARNING TO Rank) data set for the study of techniques in text search. In version 3.0 [18], the OHSUMED collection is edited from MEDLINE - a database of medical publications - for academic rankings. From the data of 106 queries, three files are created: the trainset contains 63 queries, the validation set contains 21 queries, and the testing set contains 22 queries. Each file contains records in the following format:

<lb> qid:<q> 1:<v1> 2:<v2> . ... 45:<v45>

where *<lb>* is the value of relevance; *<q>* is the query number; *<v1>, ... <v45>* are values that correspond to the features of the documents, which are calculated on the basis of common rankings for search. Some examples of attributes used include:

ID	Formula
1	$\sum_{q_i \in q \cap d} c(q_i, d)$ in the titles
5	$\sum_{q_i \in q \cap d} \log(\frac{C}{df(q_i)})$ in the titles
11	BM25 of the title
14	LMIR.JM of the title
16	$\sum_{q_i \in q \cap d} c(q_i, d)$ in the compendium
26	BM25 of the compendium
28	LMIR.JM of the compendium

Table 1: Example attribute of the OHSUMED collection.

In the above formulas, q_i is the query keyword i[th] in the query q, d is the document, $c(q_i, d)$ is the number of occurrences of q_i in the document d; C is the total number of documents in the corpus, $df(q_i)$ is the number of documents containing the keyword q_i. The BM25 and LMIR.JM scores are documented using the BM25 rating model and the Jelinek - Mercer smoothing language model [19].

3.2 Modelling application of genetic programming

The GP application solution for rating learning is as following model:

- **Input 1**: Training data set D with recording records in the form of the OHSUMED collection;

- **Input 2**: Parameters N_g is the number of generations, N_p is the number of individuals per generation, N_c is the hybrid speed, N_m is the speed of the mutation.

- **Output**: The rank function $F(q, d)$, which sets the value to a real number, corresponds to the relevance of the document d to the query q.

The training process consists of five steps as follows:

- *Step 1*: Randomly identify first generation individuals;

- *Step 2*: Determine the value of the content for each individual;

- *Step 3*: Perform hybrid and mutation operations;

- *Step 4*: Create a new generation and repeat steps from 2 to 4 until you have enough N_g;

- *Step 5*: Choose the best individual result.

Each individual (gene) is defined as a function $f(q, d)$ that measures the relevance of the document to the query, with the following options:

- Option 1: The linear function uses 45 attributes:

$$TF - AF = a_1 \times f_1 + a_2 \times f_2 + \cdots + a_{45} \times f_{45}$$

- Option 2: Linear function, using only a selective random attribute:

$$TF - RF = a_{i1} \times f_{i1} + a_{i2} \times f_{i2} + \cdots + a_{in} \times f_{in}$$

- Option 3: Apply function to attributes. Limit the use of functions x, $1/x$, $sin(x)$, $log(x)$, and $1/(1+e^x)$.

$$TF - FF = a_1 \times h_1(f_1) + a_2 \times h_2(f_2) + \cdots + a_{45} \times h_{45}(f_{45})$$

- Option 4: Create a *TF-GF* function similar to the one presented in [20], but retain the evaluation of non-linear functions. The function is binary tree, with inner vertices being operators, leaf vertices are constants or variables.

In the formulas, a_i are the parameters, f_i are the attribute values of the document, h_i are the function.

In options 1, 2 and 3, to hybridize two individuals $f_1(q, d)$ and $f_2(q, d)$, a random list of parameters has the same index of functions to be exchanged. The mutation operation for the individual, $f(q,d)$, is performed by swapping two random parameters of the function $f(q, d)$.

Comparison of search and ranking solutions is usually based on the measures *P@k, MAP, NDCG@k* [20] that is used to determine the value of the content. Here, we test the fitness function corresponding to the MAP value.

In the first two options, N_g, N_p, N_c, N_m are respectively 100, 100, 0.9, 0.1. For option 3, N_g, N_p are defined as 200,400. In option 4, N_g, N_p, N_c, N_m are respectively 1000, 100, 0.9 and 0.2. These values are determined by experiment. The N_g value, given in alternatives 3 and 4, is greater due to the complexity and diversity of individuals - the ranking function.

4 Experiment

The *TF-Ranking* experimental software, built on the basis of the *PyEvolve* library, was developed by Christian S. Perone[2], which enables the development of a genetic algorithm for development in the Python language.

In the OHSUMED collection, the data is divided into five directories, each containing the *train.txt*, *vali.txt* and *test.txt* files for training, re-evaluation, and experimentation. According to each directory, the training and experiment steps are as follows:

- The training module reads data from *train.txt* for best *pbest* selection, applying the scoring function to the text in *test.txt*.

- Microsoft's Eval-Score-3.0.pl tool is used to generate *P@k, MAP, NDCG@k* values ($k = 1,2,5,100$), evaluating the effect of the generated point function.

For each option, the mean value for each of the *P@k, MAP, NDCG@k* scores of the five directories was taken as the scores for the experimental option. The implementation of training and experiment was done 5

[2] http://pyevolve.sourceforge.net (access on 15/01/2016)

times, the average value for comparison and evaluation of results.

Table 2, Table 3 and Table 4 compare *MAP*, *P@k* and *NDCG@k* (with *k = 1, 2, 5, 10*) of the proposed solution against the baseline method, published in website of the LETOR[3] assessment data set. Bold cells contain the highest values in the corresponding column.

Method	MAP
Regression	0.4220
RankSVM	0.4334
RankBoost	0.4411
ListNet	0.4457
FRank	0.4439
TF-AF	0.4456
TF-RF	0.4467
TF-FF	**0.4468**
TF-GF	0.4427

Table 2: Comparison of MAP values

Method	K=1	K=2	K=5	K=10
Regression	0.4456	0.4532	0.4278	0.4110
RankSVM	0.4958	0.4331	0.4164	0.4140
RankBoost	0.4632	0.4504	0.4494	0.4302
ListNet	0.5326	0.481	0.4432	0.441
FRank	0.5300	0.5008	0.4588	0.4433
TF-AF	0.5506	0.4789	0.4476	0.4348
TF-RF	**0.5545**	0.4835	**0.4633**	0.4404
TF-FF	0.5294	**0.4957**	0.4600	**0.4437**
TF-GF	0.4997	0.4760	0.4507	0.4372

Table 3: Comparison of NDCG@k values

Method	P@1	P@2	P@5	P@10
Regression	0.5965	0.6006	0.5337	0.4666
RankSVM	0.5974	0.5494	0.5319	0.4864
RankBoost	0.5576	0.5481	0.5447	0.4966
ListNet	0.6524	0.6093	0.5502	0.4975
FRank	0.6429	0.6195	0.5638	**0.5016**
TF-AF	**0.6691**	0.6167	0.5499	0.4955
TF-RF	0.6642	0.6020	**0.5653**	0.4954
TF-FF	0.6619	**0.6279**	0.5612	0.4983
TF-GF	0.6220	0.6058	0.5520	0.4969

Table 4: Comparison of P@k values

Experimental results show that the *TF-AF*, *TF-RF* alternatives are good. *MAP*, *NDCG @ k* and *P @ k* values outperformed the corresponding Regression, RankSVM, and RankBoost methods, which were equivalent and slightly better than the ListNet and FRank methods. The TF-GF method was not very good: Despite the good results on the training set, the results on the experimental set were just average, sign of overfitting.

One-time training for 5 directories with *TF-AF*, *TF-TF*, *TF-FF*, and *TF-GF* options takes 150 minutes, 70 minutes, 200 minutes and 10 hours respectively on a dual-CPU computer. Core 3.30 GHz, 4 GB RAM installed Windows 7.

This result shows that the use of linear functions for ranking assures efficiency, both in terms of experimental quality and duration of training.

5 Conclusion

The paper introduces an overview on re-ranking. It evaluates the application of methods of mixing information retrieval results from multiple sources by re-calculating the scores based on the basic information returned from the original search engine and proposing a re-ranking method. sequentially, progressively download the best documents to create the final result list.

The innovation of this proposal is applying the machine learning method in using genetic programming. We experimented proposal solution on the LETOR experimental data set to develop a new ranking system with the objective of evaluating the effectiveness of this learning methodology. Experimental results suggest that the proposed method is better than traditional methods in terms of both quality and time.

Our next research is to integrate this re-ranking tool in multi-language and cross-language search systems. The systems are intended to allow users to find documents in languages other than the language of the search keywords.

Acknowledgement

This research is funded by Funds for Science and Technology Development of the University of Danang under project number B2019-DN06-18.

References

[1] Kurt I. Munson (2000), Internet Search Engines: Understanding Their Design to Improve Information Retrieval, *Journal of Library Metadata*, Volume 2, p.p. 47-60. https://doi.org/10.1300/J141v02n03_04

[2] M. Shokouhi and L. Si (2011), Foundations and Trends® in Information Retrieval, *Federated Search*, Volume 5 (No. 1), p.p. 101-107. https://doi.org/10.1561/1500000010

[3] J. Callan (2002), Distributed information retrieval, *The Information Retrieval Series: Springer*, INRE, Volume 7, p.p. 127-150. https://doi.org/10.1007/0-306-47019-5_5

[4] S. Wu, F. Crestani, Y. Bi (2006), Evaluating Score Normalization Methods in Data Fusion, *Information Retrieval Technology*, Proceedings of 3rd Asia Information Retrieval Symposium, AIRS 2006, Singapore, p.p. 642-648. https://doi.org/10.1007/11880592_57

[5] W. Shengli, B. Yaxin, Z. Xiaoqin (2011), The linear combination data fusion method in information retrieval, *Lecture Notes in Computer Science book series* (LNCS, volume 6861), pp. 219–233. https://doi.org/10.1007/978-3-642-23091-2_20

[3] http://research.microsoft.com/

[6] S. Wu, S. McClean (2005), Data Fusion with Correlation Weights, *Lecture Notes in Computer Science*, Volume 3408/2005, p.p. 275-286. https://doi.org/10.1007/978-3-540-31865-1_20

[7] B. Xu, S. Luo, K. Sun (2012), Towards Multimodal Query in Web Service Search, *19th International Conference on Web Services*, IEEE. https://doi.org/10.1109/icws.2012.42

[8] Y. Rasolofo, F. Abbaci, J. Savoy (2001), Approaches to collection selection and results merging for distributed information retrieval, *CIKM'01 Proceedings of the 10th international conference on Information and knowledge management*, ACM, p.p. 191 - 198. https://doi.org/10.1145/502585.502618

[9] L. Hang (2011), Learning to Rank for Information Retrieval and Natural Language Processing, *Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers*, p.p. 1-113. https://doi.org/10.2200/s00348ed1v01y201104hlt012

[10] C. Koby, S. Yoram (2002), Pranking with Ranking, *Advances in Neural Information Processing Systems 14*, Volume 14, p.p. 641-647. https://doi.org/10.7551/mitpress/1120.003.0087

[11] M.R. Yousefi, T.M. Breuel (2012), Gated Boosting: Efficient Classifier Boosting and Combining, *Lecture Notes in Computer Science*, p.p. 262-265. https://doi.org/10.1007/978-3-642-33347-7_28

[12] L. Yu-Ting, L. Tie-Yan, Q. Tao, M. Zhi-Ming, L. Hang (2007), Supervised rank aggregation, *Proceedings of the 16th international conference on World Wide Web - WWW '07*, p.p. 481–490. https://doi.org/10.1145/1242572.1242638

[13] K. Veningston, R. Shanmugalakshmi (2012), Enhancing personalized web search re-ranking algorithm by incorporating user profile, *Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12)*. https://doi.org/10.1109/icccnt.2012.6396036

[14] P.A. Chirita, W. Nejdl, R. Paiu, C. Kohlschütter (2005), Using ODP metadata to personalize search, *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '05*, p.p. 178--185. https://doi.org/10.1145/1076034.1076067

[15] T. Nasrin, H. Faili (2016), Automatic Wordnet Development for Low-Resource Languages using Cross-Lingual WSD, *Journal of Artificial Intelligence Research*, Volume 56, p.p. 61–87. https://doi.org/10.1613/jair.4968

[16] Y. Rasolofo, D. Hawking, J. Savoy (2003), Result Merging Strategies for a Current News MetaSearcher, *Information Processing & Management*, No 39(4), p.p. 581–609. https://doi.org/10.1016/s0306-4573(02)00122-x

[17] P.J. Angeline (1994), Genetic programming: On the programming of computers by means of natural selection, *Biosystems, MIT Press Cambridge*, p.p. 69-73. https://doi.org/10.1016/0303-2647(94)90062-0

[18] Q. Tao, L.T. Yan, X. Jun, L. Hang (2010), LETOR: A benchmark collection for research on learning to rank for information retrieval, *Information Retrieval*, Volume 13, No. 4, p.p. 346–374. https://doi.org/10.1007/s10791-009-9123-y

[19] C. Zhai, J. Lafferty (2001), *A study of smoothing methods for language models applied to Ad Hoc information retrieval*, Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '01, p.p. 334–342. https://doi.org/10.1145/383952.384019

[20] T.G. Lam, T.H. Vo, C.P. Huynh (2015), Building Structured Query in Target Language for Vietnamese – English Cross Language Information Retrieval Systems, *International Journal of Engineering Research & Technology (IJERT)*, Volume 4, No. 04, p.p. 146–151. https://doi.org/10.17577/ijertv4is040317