Ranking Function Optimization Based on OKAPI and K-Means

Jun Lu, L.T. Guo and T.F. Zhang

ABSTRACT

This paper presents the Re-ranking algorithm in the Medical information retrieval based on K-Means and OKAPI. A program is implemented to discover multiple relevant aspects of each search, group the results, and re-rank them. The Re-ranking algorithm is to be created diverse results within each query and to output the best results in comparison to the Gold Standard. In the pre-processing side, K-Means clustering algorithm is active on Information Retrieval system. The results indicate that re-ranking algorithm and K-Means clustering made the same difference of original OKAPI ranking score. We propose an explanation for these results that is based on an analysis of the specifics of the clustering algorithms and the nature of document data.

KEYWORD: Ranking Function; Re-ranking algorithm; K-Means; OKAPI; Hypotheses; WEKA

INTRODUCTION AND MOTIVATION

Type Area

The one of the most searching activities is looking for some information by using the search engines (Contentive, 2015). But it is not always very satisfactory that the engines capability of finding the exact results. The search query has to update many times and the engines need to track a couple of documents for seeking a few of words. Evaluation studies in (Gordon M. and Pathak P., 1999) show that the current state-of-the-art search engines have not well done for getting the results. In fact, current search engines and tools are very effective in certain types of queries, such as name finding, homepage finding, or finding a popular topic, but not very effective for a comprehensive search about a specific topic. Their performance results in terms of precision and recall remain to be improved.

Jun Lu, T.F. Zhang. Hubei Engineering University, Xiaogan 432100, China.
L.T. Guo, T.F. Zhang. Department of Automatic Control, Northwestern Polytechnical University, Xi’an 710072, China.
Many factors could affect the search performance: query representation, indexing, controlled vocabulary, stemming, stopping words, etc. (F.W. Lancaster, Warner A. J., 1993) But ultimately, it is affected by the ranking function, which is used to rank documents according to its match with a user’s query. There are several ranking functions available for searching (Fan Weiguo, Gordon M.D., Pathak P. 2004):

- Content-based ranking, such as OKAPI (Robertson S.E., Walker S., et al. 1996), Pivoted TFIDF (Singhal A., Salton G., Mitra M., Buckley C. 1996). These ranking function make extensive usage of many lexical/syntactical statistics of words in a document collection tf, df, document length, etc.
- Link-based ranking, utilize web interconnection information to help boost the ranking performance. Two of the most successful ranking functions are Page Rank (Brin S., Page L. 1998), HITS (Kleinberg J.M. 1999). These link-based ranking functions are especially useful to identify those authoritative pages, which are highly endorsed by others, on popular topics.
- Structure-based ranking assign weights to words appearing in different structural position, such as Title, Header, Anchor and use those weighting heuristics to improve ranking performance.

There are a comparative study on three content-based ranking functions OKAPI, ptf idf, and In query. These three ranking functions are well known in information retrieval field. Details of these three ranking functions can be found in. The study result is that OKAPI is clear winner among the three.

A research question can be made from Medical information retrieval evaluations on these ranking functions:

A program has been carried out in order to discover multiple relevant aspects of each search, group the results, and re-rank them. The goal of this Re-ranking is to create diverse results within each query and to output the best results in comparison to the Gold Standard. In addition, we aim to devise an algorithm, using pre-existing knowledge in weighting and relevance in information retrieval, which will assign a new score to the paragraph based on its previous score in addition to any other parameters that need to be introduced.

In the pre-processing side, document clustering (Steinbach M., Karypis G., Kumar V. 2000) algorithm is more efficient in performing the clustering by considering each document as initial centroid and then merges those documents into a cluster by considering the relevancy of contents, until all documents in a cluster have similar feature. The most common document clustering techniques is K-Means clustering. It is interesting that how K-Means clustering algorithm active on Information Retrieval system. The following questions is attempted to address:

- Would K-Means clustering make the difference of original OKAPI ranking score?
- Are K-Means clustering algorithm absolutely critical to the IR system ranking score?
- If yes, how much difference would it be? Would it be a significant increment? If no, how much decrease would it be?

The next section, the paper describes the K-Means algorithm and OKAPI concepts were presented. Section III discusses the experimental methods. Two levels of hypotheses were shown and the Re-ranking algorithm is introduced. Section IV
presents the experimental results and evaluates the hypotheses. Section V concludes this paper and point out future research directions.

LITERATURE

OKAPI Algorithm

OKAPI BM 25 is a function used by search engines to rank documents according to their relevance to a search query (see in Table I). It uses term frequency, document frequency, and document length as parameters to determine an IR score for each document. This method differs from \( tf \), \( qtf \) in that it also considers document length when assigning a weight (Witten Ian H., Frank Eibe.2005). In the project, the given data sets have already been assigned a score by the OKAPI information retrieval system, and these scores will also be used in the Re-ranking portion of the project as well.

\[
\omega = \frac{(k_1+1) \times tf}{K+tf} \times \log \frac{N-n+0.5}{n+0.5} \times \frac{k_1+1}{k_1+qtf} \times \frac{(advl-dl)}{(advl+dl)}
\]

BM25 Variables References see Table 1.

| Variable | Description |
|----------|-------------|
| \( N \)  | total number of indexed documents in the collection. |
| \( n \)  | number of documents containing the term \( t \). \((n \leq N)\) |
| \( tf \) | within document term frequency of \( t \) in \( d \). |
| \( qtf \) | within query term frequency of \( t \) in \( q \). |
| \( nq \) | number of query term(query length). |
| \( dl \) | length of document \( d \). |
| \( avdl \) | average length of all indexed documents in the collection. |
| \( k \)  | \( k_1 \times \left(1 - b \times \frac{dl}{avdl}\right) \) |
| \( k_1,k_2,k_3,b \) | constants, used to tune the system. |

functionDirect = \( k \) - Means\( ( ) \)

Initialize \( k \) prototypes \( \{\omega_1 , \ldots , \omega_k\} \) such that \( \omega_j = i_j, j \in \{1,\ldots,k\}, i \in \{1,\ldots,n\} \)

Each cluster \( C_i \) is associated with prototype \( \omega_j \)

Repeat

foreach input vector \( i_l \), where \( l \in \{1,\ldots,n\}, \) do

Assign \( i_l \) to the \( C_j \), cluster with nearest prototype \( \omega_j \), \( \{ i_e , |i_l - \omega_j| \leq |i_l - \omega_e|, j \in \{1,\ldots,k\} \} \)

foreach cluster \( C_j \),where \( j \in \{1,\ldots,k\}, \) do

Update the prototype \( \omega_j \) to be the centroid of all samples currently in \( C_j \), so that \( \omega_j = \sum_{i \in C_j} i_l \left| C_j \right| \)

Compute the error function :

\[ E = \sum_{j=1}^{k} \sum_{i \in C_j} \left| i_l - \omega_j \right|^2 \]

Until \( E \) does not change significantly or cluster membership no longer changes

Figure 1. Direct K-MEANS Clustering.
K-Means Algorithm

In the organization of Medical data, there are many situations in which the data collected has no class attributes (Fan Weiguo, Gordon Michael D., Pathak Praveen. 2004). Clustering is the process of organizing data into groups where the members are similar in some way. Basic K-Means algorithm is show in Figure 1.

APPROACH

Hypotheses

We are presenting the hypotheses in this section. The hypotheses reflect the expectation that there is a difference between the OKAPI system and our developed Project system. We expect our Project system with clustering function could improve the performance on Information Retrieval system, and improvement could be statistically significant. There are two levels of hypotheses: one that designs at aspect level, and the other one compares at passage level.

1) \( H_0 \) – No significant difference in score between none K-Means clustering Re-ranking IR system and K-Means clustering Re-ranking IR system with aspect category.

2) \( H_0 \) – No significant difference in score between none K-Means clustering Re-ranking IR system and K-Means clustering Re-ranking IR system with passage category.

In addition, we want to state four alternative hypotheses to support all stated hypotheses:

3) \( H_1 \) – The K-Means clustering Re-ranking IR system get better scores than none clustering IR system with aspect category.

4) \( H_2 \) – None K-Means clustering Re-ranking IR system get better scores than the clustering IR system with aspect category.

5) \( H_1 \) – The K-Means clustering Re-ranking IR system get better scores than none clustering IR system with passage category.

6) \( H_2 \) – None K-Means clustering Re-ranking IR system get better scores than the clustering IR system with passage category.

These hypotheses will allow us to make some conclusions about the ranking between the two IR systems, although the data does not allow us to reject all null-hypotheses or support all the alternative hypotheses (Pitkow J., Schutze H., Cass T. et al. 2002).

Procedure

The procedure is presented in Figure 2.
In the first stage, we begin with processing the original “top-passages-york07ga1.txt”. From that single file, we have 36 directories generated based on the 36 topics. Within each of these directories, a single CSV file is generated for each paragraph entry found in the topic, resulting in 1000 CSV files for each topic. The CSV files are then processed for stop words and stemming, and finally all documents within the directory are merged into a single CSV file that represents the topic. Formatting of this CSV file is done by means of the WEKA API in order to ensure its compatibility to the clustering tools.

In the next stage, clustering is carried out using the WEKA Simple K-Means class. For this project, each topic has been assigned a total of 10 clusters to be generated. Since the project of centroids is random, there can be any number of members within each cluster (Zheng Lei, Cox I.J. 2009).

After clustering, the implementation carries out the task of assigning a new score to the members within each cluster. To assign this new score, we use a formula in an attempt to improve upon the ones provided that has been assigned by OKAPI. This algorithm will be elaborated on in the code implementation section of the report. This process is repeated for the clusters within each topic.

When each of the members in each topic have been reassigned a score, the topic is then re-ranked based on this. Each topic’s new re-ranked list is then appended to a new retrieval results file based on the format provided by “output-format-york07ga1.txt”. The results are then fed through to the TREC Python evaluation program to evaluate the effectiveness of the ranking algorithm.

The experiment is conducted on OKAPI system. We have implemented a new Project system using WEKA API which clusters the data into several groups and our own idea of Re-ranking algorithm. And then, we compare evaluation scores on two systems by using the TREC Python evaluation program. This experiment addresses these three controlled factors:

1) Our data set is randomly cut from a big medical data set. We use the same data set for two compared systems. The data set is 14,795 KB, 36 topics and all text formats.

2) We used same evaluation tool which is the TREC Python evaluation program to generate a scalable score for each system on each topic.

3) We used the same computer to result this experiment in order to get fair compare result.

We would like to build a new score based on original OKAPI score by giving a K-Means clustering algorithm and our own Re-ranking formula “treatment”. We have used two group of K-Means separately set on two categories OKAPI original ranking scores and need four groups of data sets need to be analysis and observation.

Re-ranking Algorithm

During the score reassignment phase, the current member of a cluster calls the following function to have its score re-calculated. To begin with, the algorithm requires the max score from one of the clusters generated by the clustering phase. This is retrieved by using the cluster ID to get the appropriate score from the maxs array. The maxs array stores the double value of the highest scores in each cluster. If the max score is greater than the old score assigned by OKAPI and the old score is greater than the given cluster’s average, a new score is computed based on the old score divided by
the max score multiplied by 30. This is then added to log10 of the given cluster’s size divided by 10 and multiplied by 12 (see in Figure 3). The new score is then assigned to the member of the cluster that has called the function. In the case of the maximum score being less than the old one and the old score less than the cluster average, the old score is just reassigned to the member.

```java
//get max score
double maxScore = maxs[item.getClusterId()];
//if the old score is not too high and not too low,
//increase it a little bit by normalize it and plus weight of its cluster
//'-3' here is arbitrary decide, change to automatically decide is better
if (((maxScore - 3) > oldScore) && (oldScore > (clusterAverage - 3)))
{
    //the new score has two parts: the old score and size of its cluster
    //weight are '30' and '12'. change to automatically decide is better
    newScore = oldScore / maxScore * 30 +
    Math.log10(item.getClusterSize() / 10) * 12; }
else
{
    // if the old score is too large or too small, just leave it
    newScore = oldScore; }
```

Figure 3. Re-Ranking Algorithm.

The first if statement sets the condition that the maximum score of the current item is not too big or not too small, and the constants set were determined through some experimentation. Ideally, these two parameters would be calculated dynamically based on the range of scores of the given cluster. The ranking algorithm itself is made up of two parts: the old score, that is, the one assigned by OKAPI, and a normalized score based on the cluster size. Again, the constants here were determined through experimentation and would have changed in subsequent versions of this implementation. In the else case scores that were too big or too small were considered out of range and no better than the old score. The items would be re-arranged in descending order – highest score being ranked first and so on.

**RESULT**

On average, the implementation takes about 10 minutes to process the initial data set and convert into the desired formats for processing. A majority of the running time is attributed to the clustering. In total, the running time can be around 40 or more minutes, about 30 of it for clustering.

We have pass different values into two parameters which are number of clusters and number of seeds by using K-Mean API. We got the results as you could see in below Table 2. With aspect category, it is increased 23.10% on average by using 2 clusters and 50 seeds. The average percentage of difference becomes larger when the clusters number increase to 6. The same expected result we got with passage category. We got 7.24% and 4.45% on average percentage of difference by using the combination of 6 clusters 50 seeds and the combination of 2 clusters 50 seeds.

| Category | Clusters No. | Seeds No. | OKAPI | PROJECT | Difference (AVG) |
|----------|--------------|-----------|-------|---------|------------------|
| Aspect #1 | 6            | 50        | 0.1017| 0.1293  | 27.14%           |
| Aspect #2 | 2            | 50        | 0.1017| 0.1252  | 23.10%           |
| Passage #1| 6            | 50        | 0.0373| 0.0400  | 7.24%            |
| Passage #2| 2            | 50        | 0.0373| 0.0390  | 4.45%            |
Similar to OKAPI, the scores that are assigned by the implementation to the paragraphs fall within the range of ~20 to ~40. This ensures that the scores are at least comparable with the addition of normalization. Perhaps the most important characteristic affecting the design and execution of the implementation is the number of parameters required for the implementation. From the outset, K-Means already requires 2 input parameters. The scoring function alone can require 6 or more parameters from within the solution itself. After finishing the tests, the summary of hypothesis tests exhibits the following as below Table 3.

| Test No. | Null Hypothesis | Test          | Sig.    | Decision       |
|---------|-----------------|---------------|---------|----------------|
| Aspect #1 | The distribution of Performs is the same across categories of Method | Independent Samples Mann-Whitney U Test | 0.496   | Retain the null hypothesis |
| Aspect #2 |                      |               | 0.620   |                |
| Passage#1 |                      |               | 0.702   |                |
| Passage#2 |                      |               | 0.901   |                |

We have done two tests for $H_{a1}$ to $H_{a2}$ as we see Table 3. Even though, we have seen an improvement on Project system, the difference is not significant enough for us to reject the null hypothesis $H_{a1}$. As we have seen the result from SPSS that we have to accept $H_{a1}$, that no significant difference in score between none K-Means clustering Re-ranking IR system and K-Means clustering Re-ranking IR system with aspect category. As we accepted $H_{a1}$, we need to verify the two related alternative hypotheses, we could see that K-Means clustering Re-ranking IR system get better scores, therefore, we have conformed to $H_{a1}$.

We have done two tests for $H_{a2}$, same as $H_{a1}$, we have seen an improvement on Project system, but the difference is not significant enough for us to reject the null hypothesis $H_{a2}$. As we have seen the result from SPSS that we have to accept $H_{a2}$ that no significant difference in score between none K-Means clustering Re-ranking IR system and K-Means clustering Re-ranking IR system with passage category.

Can examine the two related alternative hypotheses after have accepted $H_{a2}$, and we could see that K-Means clustering Re-ranking IR system get better scores than none clustering IR system with passage category; therefore, we have conformed to $H_{a2}$.

On top of the above observation, we can see that the difference between the two indicates that the first set of parameters is only slightly better than the second. In addition, it is also interesting to note that for both cases, topics 202 and 225 seem to have no OKAPI score, no re-assigned score, and therefore no difference. This is after running it through the TREC evaluation program, so perhaps the problem may lie in the evaluation program, the particular topic’s formatting, or a discrepancy with the golden standard.

From the analysis, we need to dismiss the null hypothesis $H_{a2}$ because the comparison results are not large enough. Consequently, we can observe the two related hypotheses, $H_{a1}$ and $H_{a2}$. The variation charts show a little that the Project ranking score is well again than the original OKAPI scores. This let us to validate of the hypothesis $H_{a1}$ that the K-Means clustering Re-ranking IR system get better scores
than none clustering IR system with aspect category. And also this allows us to reject the counter-hypothesis $H_{22}$ that none K-Means clustering Re-ranking IR system get better scores than the clustering IR system with aspect category. The experiment also compares OKAPI, OKCA and Pivoted TFIDF method and result show on Figure3 and Figure4.

As the number of documents increases, the accuracy of continuous improvement.

CONCLUSION

One of the potential improvements that we found to be clear from the start was to handle the "noise". One solution suggested to handle noise was to create a threshold or range for the number of documents in each cluster - i.e. clusters that do not fall within the criteria will be considered “less” in terms of scoring and ranking.

However, this might only work if the initial clustering is balanced/well done. In the case that the clustering is bad, the relevant documents may not be ranked appropriately making the entire process incorrect. In the implementation, we considered that perhaps using an alternative clustering method, such as EM to address this. In addition to removal of noise, we did not do anything like set a threshold for keywords in each paragraph. This would have served to reduce the number of keywords and possibly the execution time.

In conclusion, the project has succeeded in carrying out its objectives. The implementation was able to carry out the preprocessing and clustering portions of the project without any significant problems. The solution for re-assigning scores may not have been as good as OKAPI’s, but during the testing and analysis portion we were able to conclude that it slightly improved the OKAPI score. The implementation was also able to carry out Re-ranking based on this as well as evaluation of the final output.

ACKNOWLEDGMENT

This work was supported by the Program of Science Foundation of Hubei Province (2014CFB576) and Science Foundation of Hubei Province (z2013016, 2014CFB576), and Scientific research program Funded By Hubei Provincial Department Of Education 2016 and Scientific research program Funded By The public security Department of Hubei Province (hbst2014yyxc03).
REFERENCES

1. Contentive. 2015. www.searchenginewatch.com.
2. Gordon M., Pathak P., 1999. Finding information on the world wide web: the retrieval effectiveness of search engines, *Information Processing and Management*, vol. 35, no. 2: 141–180. Amsterdam: ELSEVIER.
3. F.W. Lancaster, Warner A.J., 1993. *Information Retrieval Today*. Arlington: Information Resources Press.
4. Fan Weiguo, Gordon M.D., Pathak P. 2004. Wensi Xi, Fox E.A., Ranking function optimization for effective Web search by genetic programming: an empirical study, *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004*, v 37: 1638-1690. Hawaii: Shidler College of Business.
5. Robertson S.E., Walker S., Jones S., Hancock-Beaulieu M.M., Gatford M. 1996. OKAPI at TREC-4, in *Proceedings of the Fourth Text Retrieval Conference*, D.K. Harman, Ed. NIST Special Publication,1996: 73–97. Gaithersburg: NIST.
6. Singhal A., Salton G., Mitra M., Buckley C. 1996, Document length normalization, *Information Processing and Management, Sep 1996*, v 32, n 5: 619-633. Amsterdam: ELSEVIER.
7. Brin S., Page L. 1998. The anatomy of a large-scale hyper textual Web search engine, *Computer Networks and ISDN Systems*, vol. 30, no. 1–7: 107–117. Amsterdam: ELSEVIER.
8. Kleinberg J. M.1999. Authoritative sources in a hyperlinked environment, *Journal of the ACM,1999*, vol. 46, no. 5: 604–632. New York: Assoc Computing Machinery.
9. Steinbach M., Karypis G., Kumar V.2000. A Comparison of Document Clustering Techniques, *Proc. KDD-2000 Workshop on Text Mining, Aug 20-23, 2000*. Boston: Assoc Computing Machinery.
10. Witten Ian H., Frank Eibe. 2005. *Data Mining: Practical machine learning tools and techniques*, 2005. San Francisco: Morgan Kaufmann.
11. Fan Weiguo, Gordon Michael D., Pathak Praveen. 2004. Discovery of Context-Specific Ranking Functions for Effective Information Retrieval Using Genetic Programming, *IEEE Transactions on Knowledge and Data Engineering, April 2004*, v 16, n 4: 523-527. New York: IEEE.
12. Fan Weiguo, Gordon Michael D., Pathak Praveen. 2004. A generic ranking function discovery framework by genetic programming for information retrieval. *Information Processing and Management, July 2004*, v 40, n 4: 587-602. Amsterdam: ELSEVIER.
13. Pitkow J., Schutze H., Cass T., Cooley R., Turnbull D., Edmonds A., Adar E., Breuel T.2002. Personalized search, *Communications of the ACM, September 2002*, vol. 45, no. 9: 50–55. New York: Assoc Computing Machinery.
14. Zheng Lei, Cox I.J. 2009. Document-Oriented Pruning of the Inverted Index in Information Retrieval Systems, *Proceedings - International Conference on Advanced Information Networking and Applications, 2009*: 697-702. Columbus: WAINA.