Diversity based re-ranking of search results using Affinity Graph

Manuela Bergau
s4543645

Shima Yousefi Roudbordeh
s1039415

Evgeniia Martynova
s1038931

ABSTRACT

In this project, we address the problem of redundancy of search results. We implement a re-ranking algorithm which combines a concept of document affinity with ideas from link analysis research field [10] and compare its performance to BM25 baseline retrieval system. The results in the document collection gained by exploiting Wikipedia disambiguation pages demonstrate that our re-ranking algorithm could not improve the quality of the top ten results.

1 INTRODUCTION

Modern search systems have reached the impressive results in terms of document relevance to a query, however, their architecture and the current state of web documents collection lead to usability problems. One of them is the redundancy of search results when the content of the top-ranked documents overlaps to a large extent. Since the documents shown on the first page gain the most user attention, the utility of search results list containing many similar documents on top is little for the users.

Interpretation of a search query depends on many factors which are hidden from the search engine. Users might express different information needs with the same query, thus it is valid to assume that the search engine output, which contains more diverse results on top, would improve overall user experience. In this project, we examine how the re-ranking algorithm suggested in [10] can help to reach this goal. The authors of this study introduce two novel concepts:

- **diversity**, which measures the variety of topics in a group of documents
- **information richness**, which measures the coverage of different topics in a single document.

Both these metrics are query-independent and based only on the content of documents in the collection. To compute these metrics and exploit them for document re-ranking they develop an Affinity Ranking Framework. Experimental results in Yahoo! Directory, ODP Data, and Newsgroup data demonstrate that their proposed ranking algorithm significantly improves the search performance.

We want to replicate the results of the paper, therefore we implement the proposed algorithm and evaluate the performance by comparing nDCG and precision metrics of re-ranked search results to results of a BM25 baseline retrieval system.

Conference'17, July 2017, Washington, DC, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 AFFINITY RANKING FRAMEWORK

The Affinity Ranking algorithm is based on the ideas from the link analysis research field. Traditional approaches to the mining of the links structure between documents (e.g. PageRank and HITS algorithms [5]) make use of hyperlinks between documents and tracking of user behaviour to obtain the information about implicit links between web pages. The drawback of these methods is that they based on the subjective human judgements and do not necessarily reflect the similarity of information in documents objectively. The main idea of Affinity Ranking is to construct the Affinity Graph (AG) of documents based on the objective information measures and use it to re-rank the original search results. The algorithm consists of four steps:

2.1 Affinity Graph Construction

The AG is built based on the asymmetric similarity measure which reflects the significance of similarity between documents d_i and d_j and called **affinity**. Let $\vec{d_i}$ be a representation of a document d_i according to the traditional vector space model [8], then the affinity of $\vec{d_j}$ to $\vec{d_i}$ is defined as

$$aff(d_i, d_j) = \frac{\vec{d_i} \cdot \vec{d_j}}{||\vec{d_i}||} \quad (1)$$

Mathematically it is just a projection of vector $\vec{d_j}$ to $\vec{d_i}$. From this formula it follows that larger documents have higher affinity to other documents, which reflects the idea that large documents are more informative than short ones. After calculating affinity between all documents the AG is constructed by considering the documents as nodes and adding the directed links from d_i to d_j with weights $aff(d_i, d_j)$ if $aff(d_i, d_j) > aff_t \ \forall i \neq j$, where aff_t is a threshold.

2.2 Information Richness Computation

Information Richness is computed for each document based on the affinity graph by link analysis algorithm similar to PageRank. To describe the AG adjacency matrix **M** is used:

$$M_{i,j} = \begin{cases} aff(d_i, d_j) & \text{if } aff(d_i, d_j) > aff_t \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

M is then normalized such that the sum of the rows equals to 1 and normalized adjacency matrix $\tilde{\mathbf{M}}$ is used in further calculations.

With the assumptions that the more neighbors a document has and the more informative document's neighbors are, the more informative the document itself is, the authors propose the following formula to calculate information richness score of each document:

$$InfoRich(d_i) = c \cdot \sum_{all j \neq i} InfoRich(d_j) \cdot \widetilde{M}_{j,i} + \frac{(1-c)}{n} \quad (3)$$

And in matrix from:

$$\lambda = c\tilde{\mathbf{M}}^{\mathbf{T}}\lambda + \frac{(1-c)}{n}\vec{e} \quad (4)$$

Where $\vec{e}$ is a unit vector and c is a dumping factor similar to the random jumping factor in PageRank. In the paper c is set to be 0.85 value in the experiments [10] and we use the same value in our project. The authors explain the information richness computation as a derivation from random information flow model which is an adaptation of the random surfer model [5].

Let d_i be a document at which information flow stops at current iteration and the $A(d_i) = \{d_j | \forall j \neq i, aff(d_i, d_j) > aff_t\}$ is the set of documents which d_i links. From this state information can either move to any document from $A(d_i)$ with the probability proportional to $aff(d_i, d_j)$ or randomly move to any document in the collection. The latter option is an analogue of a teleport operation in the random surfer model. If a document does not contain outgoing links, the teleport is always performed and for a document with out-links a standard information flow operation occurs with the probability c and the teleport with probability $1 - c$. With this reasoning, the authors reduce the problem of information richness computation to a Markov chain process, where the states are given by the documents and the transition matrix is given by $c\tilde{\mathbf{M}}^{\mathbf{T}} + \frac{(1-c)}{n}\mathbf{U}$, where $\mathbf{U} = [\frac{1}{n}]_{n \times n}$. Markov's transition probability matrix is known to have a principal left eigenvector which corresponds to the stationary probability distribution of each state, which is equivalent to Equation 4. Therefore, the coordinates of the principal eigenvector represent affinity score of each document in the collection.

2.3 Diversity Penalty

Information richness measure helps to present the most informative documents at the top of search results, however, it does not solve the redundancy problem. To increase the coverage of topics in the top search results the authors propose the greedy algorithm to impose redundancy penalty to the score of each document. The brief description of the algorithm:

1. Initialize two sets, $A = \emptyset$ and $B = \{d_i | \forall i\}$ and assign each document its affinity rank equals to information richness rank $AR_i = InfoRich(d_i)$.
2. Sort documents in B by their current score in descending order.
3. Move the top ranked document d_i from B to A then impose a penalty to the score of each document which has a link to d_i: $AR_j = AR_j - \tilde{M_{i,j}} \cdot InfoRich(d_i)$
4. Repeat steps 2 and 3 until all the documents are re-ranked or the iteration reaches a predefined maximum count.

At step 3 the penalty for each document, which links d_i, if proportional to its affinity to d_i. Thus, after applying this algorithm we get the final affinity ranking in which both information richness and diversity of the results are considered.

2.4 Re-ranking

The re-ranking is implemented as a linear combination of query similarity obtained with BM25 ranking and affinity scores. Since the two scores are always on a different order of magnitudes and their raw values vary in a different range, normalization should be applied before combining the scores. In the original paper, they perform average normalization of query similarity score and log average normalization of affinity score. However, we did not find the log average normalization of affinity rank meaningful and we do average normalization as well. First of all, the diversity penalty algorithm is very likely to give negative scores in the last half of the list. This can be fixed by shifting the scores by a constant so that all the scores have positive values and the relation between ranking is unchanged. We assume that the authors wanted to soften the decrease of the first top K affinity scores which we discovered to be quite rapid in our experiments (see Figure 6 in Appendix). Still, the result of the division of logs of two values is close to the division of unchanged values.

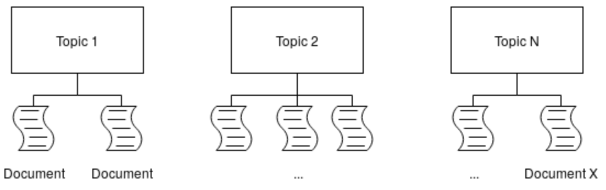

Figure 1: Illustration of the data structure

Therefore, the re-ranking formula we use is the following:

$$Score(q, d_i) = \alpha \cdot \frac{Sim(q, d_i)}{\overline{Sim}_\Theta(q)} + \beta \cdot \frac{AR(d_i)}{\overline{AR}_\Theta}, \forall d_i \in \Theta \tag{5}$$

Where α and β are tuneable constants that regulate which ranking gains more weight and satisfy the constraint $\alpha + \beta = 1$. $\overline{Sim}_\Theta(q) = Max_{\forall d_i \in \Theta} Sim(q, d_i)$ and $\overline{AR}_\Theta = Max_{\forall d_i \in \Theta} AR_i$, where Θ is the set of the documents returned for a query.

3 EXPERIMENTAL SETUP

3.1 Data

For a meaningful evaluation of search results diversity, we needed an annotated collection in which mapping of the documents to topics and subtopics covered by each document is known. TREC Interaction track data set's annotation is a good example of such mapping however, it was impossible to obtain it. The solution we found was to make use of Wikipedia disambiguation pages. We can see that the structure of Wikipedia disambiguation pages is similar to topics/subtopic mapping to documents which we were seeking. The information from Wikipedia serves all our purposes: it allows to construct queries based on page titles, get relevance judgments as well as annotation of documents with topics and subtopics they cover. We consider a document relevant to a query if the disambiguation page, based on which the query was constructed, has a link to this document (Figure 1). The titles of disambiguation pages are topics and each document covers one subtopic of a topic. With this approach, each document is annotated only with one topic and one subtopic, whereas in fact documents often cover multiple topics and subtopics. Also, it is likely that the obtained relevance judgements and topics/subtopic annotation do not include all documents a human would consider relevant. These problems make an evaluation of the results questionable. However, this is the best collection we could get under the project constraints.

We collected 120,594 documents from 18143 disambiguation pages. The disambiguation pages were selected randomly from the set of all links to Wikipedia disambiguation pages. We excluded disambiguation pages for which it was possible to download less than two documents due to broken links or any other exceptions.

3.2 Implementation

We implemented the algorithm described in Section 2 with Python programming language. The source code can be found in the GitHub repository [1]. The essential difference between our implementation and the original algorithm is that the affinity graph is constructed only for the set of the documents extracted for a query. This makes implementation more feasible and independent on the documents collection to some extent, however, it introduces a performance bottleneck and might distort the results of the algorithm. Given the time constraints of the project, we think that it is a fine trade-off between feasibility and replication of the original paper results. Clearly, this approach can not be used in live systems.

We chose to use Lucene search engine and conduct the experiments with Anserini IR toolkit [9]. Since Anserini does not provide an out of the box support of custom document collections, we had to convert our data set into the TREC format. Then we indexed the collection with Anserini and used Pyserini (python interface to essential Anserini objects) to extract the search results for re-ranking. A few implementation-wise decisions we made:

1) To reduce the dimensionality of the vector space model and make affinity reflect actual documents content we removed the stop words during transformation of the documents to vectors. We decided to use the NLTK's stop words list because scikit-learn's built-in stop words list is known to have a few issues [6].

2) For the computation of principal eigenvector, we used SciPy library. The implementation of SciPy's function to find eigenvalues and eigenvectors has some optimizations for better numerical stability and thus it is more robust than similar functions from standard Python math package.

3) The least trivial task was to choose the value of the affinity threshold. The authors of the original paper do not mention the threshold they used or any criteria to choose it. Therefore, we decided to find the threshold empirically based on the knowledge that the affinity graph should be sparse and we want to add edges between the documents with significant affinity only. However, for affinities calculated with equation 1 it is unclear which values should be considered significant, because it does not have an upper boundary. Partially this problem was addressed by stop words removal, which helped to substantially decrease the difference between the smallest and largest affinities. Further, to make the choice more intuitive we normalized the affinity matrix by division on the maximum value so that all its item were in a [0, 1] interval. Then we run the affinity ranking algorithm with different affinity thresholds for the queries selected for experiments and checked the number of edges added to affinity graph and the median value of documents affinities to get an idea of resulting graph structure. Based on the observations we selected $aff_t = 0.1$. For this value, the average percentage of added edges out of all the possible edges is ≈ 3.6, which we found satisfactory.

3.3 Evaluation

3.3.1 Queries and Relevancy. To evaluate the implementation we use the names of the aspect topics as queries and consider all documents linked as relevant. We considered only aspect topics that have more than 20 documents linked.

Query	
St. Mary's Church	Union Station
James Brown	Army nursing
Falcon	Masonic Temple
Thomas Smith	Isis
Odyssey	Spider
Vector	Spirit
Franklin	Charles Brown
Wake Up	Island
Hesar, Iran	Skelton
Gypsy	Discovery

Figure 2: List of queries used for evaluation

3.3.2 Evaluation. We compared the resulted ranking to the original BM25 ranking of search results. Our measures are nDCG and precision. We were thinking of using α-nDCG which is an evaluation measure for diversified search. It is defied as

$$C_i(r) = \sum_{k=1}^{r} I_i(k) \tag{6}$$

where r is the rank $I_i(k) = 1$ if the document at rank r is relevant to intent i and $I_i(k) = 0$ otherwise [7]. Due to the structure of our collection the usage of α-nDCG will result in the same value for both lists.
For precision we only consider the top 10 results, as the top documents are the most likely to be viewed by the user. We define precision as followed:

$$precision = \frac{H_{\text{relevant}}}{10} \tag{7}$$

Where H_{relevant} is the number of relevant documents in the top 10.

3.4 Results

3.4.1 α tuning. As it can be seen in equation 5 the ranking includes two parameters that can be tuned. These parameters control to which extend the original ranking and the affinity graph is taken into account. Due to the fact that $\alpha + \beta = 1$ we only set α. Table 3 shows the effect of the parameters on the nDCG and precision. Note that $\alpha = 1$ is equal to the original ranking. For $\alpha = 0.25$ and $\alpha = 0.5$ the values of the used metrics are almost the same as for the documents order based on affinity rank only. This might be caused by the typical trend of affinity scores which decrease sharply for a first few documents and then reaches a plateau (see Figure 6 in Appendix). As a result, affinity rank tends to dominate query similarity rank. Based on these results we chose for $\alpha = 0.75$ for our more detailed tests.

α	0	0.25	0.5	0.75	original ranking
nDCG	5.61	5.61	5.61	5.66	6.05
precision	0.52	0.53	0.52	0.56	0.6

Figure 3: Change of the mean values depending on α

3.4.2 *nDCG.*

According to the nDCG mean of 5.66, our implementation slightly worsens the overall results. When examining the individual results of all test queries (see Figure 4) we see that in most of the cases the nDCG changes are only minimal higher or lower than the original. In four cases the nDCG is significantly worse. A possible reason can be, that the construction of the affinity graph only for the documents extracted for the query could affect algorithm results. Also, incomplete relevance judgements we have might make evaluation with precision and nDCG metrics less reliable. Furthermore, we could not measure diversity as our document set does not have enough annotations of different subtopics.

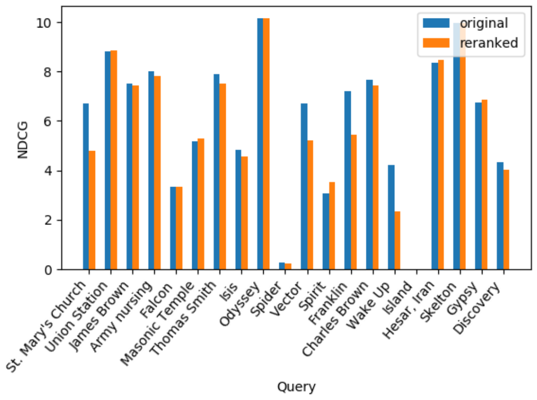

Figure 4: results of the nDCG measure

3.4.3 *Top 10 search results.*

Similar to nDCG, the precision values for our selected 20 queries in the top 10 re-ranked search results are not as good as the original search result. As we can see in Figure 5, for nine queries we have the same precision value as the original search result and for another nine queries our implementation resulted in lower values. But in two cases our algorithm acts better and the precision value is higher than the original search result.
Although we used topics with more than 20 related documents as our query set, we think that the precision value for our top 10 results is not satisfactory for evaluation. One of the reasons can be, that we do not have many documents with overlapping content.

4 RELATED WORK

So far several attempts were made to improve the diversity of search results. Carbonell et al[2] presented a re-ranking method based on Maximal Marginal Relevance (MMR) which criterion strives to reduce redundancy while maintaining query relevance in re-ranking retrieved documents. The results indicate some benefits of MMR diversity ranking in document retrieval. This approach does not include a direct criterion of diversity evaluation to ensure that the group of documents with low redundancy can achieve large topic coverage[10].
Huang et al[3] proposed a Bayesian learning approach to promoting diversity for information retrieval in biomedicine and a re-ranking model to improve retrieval performance in the biomedical domain.

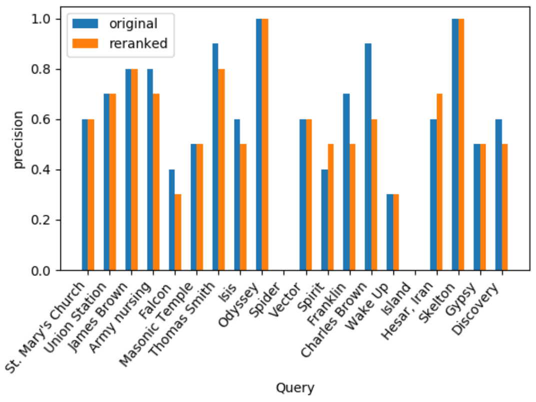

Figure 5: results of the precision measure

The experimental results show the effectiveness of Bayesian learning for promoting diversity in ranking for biomedical information retrieval on four years TREC data sets. As this method was applied to a specific data domain, we are not sure if it can be generalised to other domains.
A comprehensive survey by Matevž Kunaver and Tomaž Požrl[4] provides an overview of research done on this topic starting at one of the first mentions of diversity in 2001 until 2015. The articles explain the work that has been done in the field of recommendation diversification: the definition and evaluation of diversity; the impact of diversification on the quality of recommendation results and the development of diversification algorithms themselves.

5 DISCUSSION

In this paper, we used diversity and information richness metrics, for document re-ranking. Additionally, we applied Affinity Ranking algorithm to re-rank the search results extracted with Anserini IR toolkit. Our ultimate goal was to improve the diversity of the top ten results. As the results show, we did not succeed all the way. The main reason for this is, that there was no dataset available to us with annotated aspects and a relevance judgement. This made it hard to evaluate our implementations performance.
In addition to this our implementation has performance issues as we are constructing the affinity graph with each query. As a result, our implementation is not ready for real live deployment.
However, we found two possible flaws in the original paper, which are not addressed by the authors. The first one is uncertainty about the affinity threshold choice. We think that a clear way to set it, derived from appropriate mathematical concepts, should be devised. And the second one is related to the shape of the affinity scores curve. An interesting future work direction could be to examine why it changes this way, how it affects the re-ranking results and how the distribution of affinity scores can be smoothed.

REFERENCES

[1] Manuela Bergau, Shima Yousefi, and Evgeniia Martynova. 2019. Project source code. https://github.com/EvgeniyaMartynova/IR_project

[2] Jaime Carbinell and Jade Stewart. 2017. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. *ACM SIGIR Forum* 51 (08 2017), 209–210. https://doi.org/10.1145/3130348.3130369

[3] Xiangji Huang and Qinmin Hu. 2009. A Bayesian Learning Approach to Promoting Diversity in Ranking for Biomedical Information Retrieval. *Proc. of the 32nd ACM SIGIR* 51 (01 2009), 307–314. https://doi.org/10.1145/1571941.1571995

[4] Matevž Kunaver and Tomaz Pozrl. 2017. Diversity in Recommender Systems, A Survey. *Knowledge-Based Systems* 123 (02 2017). https://doi.org/10.1016/j.knosys.2017.02.009

[5] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schutze. [n.d.]. Introduction to Information Retrieval. Cambridge University Press 2008. *Ch* 21 ([n. d.]), 461–481.

[6] Joel Nothman, Hanmin Qin, and Roman Yurchak. 2018. Stop word lists in free open-source software packages. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. 7–12.

[7] Tetsuya Sakai. 2018. *α-nDCG*. Springer New York, New York, NY, 98–99. https://doi.org/10.1007/978-1-4614-8265-9_80619

[8] SK Michael Wong and Vijay V Raghavan. 1984. Vector space model of information retrieval: a reevaluation. In *Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*. British Computer Society, 167–185.

[9] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, New York, NY, USA, 1253–1256. https://doi.org/10.1145/3077136.3080721

[10] Benyu Zhang, Hua Li, Yi Liu, Lei Ji, Wensi Xi, Weiguo Fan, Zheng Chen, and Wei-Ying Ma. 2005. Improving Web Search Results Using Affinity Graph. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*. ACM, New York, NY, USA, 504–511. https://doi.org/10.1145/1076034.1076120

APPENDIX

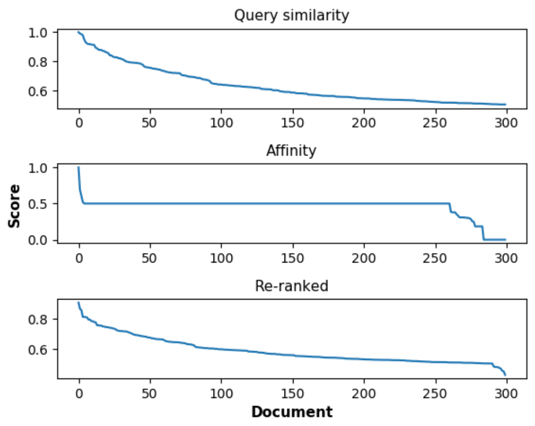

Figure 6: Typical changes of scores from top to low ranked documents for different types of ranking. All the scores are normalized.