Improving Similar Document Retrieval Using a Recursive Pseudo Relevance Feedback Strategy

Kyle Williams[‡], C. Lee Giles[†‡]
[‡]Information Sciences and Technology, [†]Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802, USA
kwilliams@psu.edu, giles@ist.psu.edu

ABSTRACT

We present a recursive pseudo relevance feedback strategy for improving retrieval performance in similarity search. The strategy recursively searches on search results returned for a given query and produces a tree that is used for ranking. Experiments on the Reuters 21578 and WebKB datasets show how the strategy leads to a significant improvement in similarity search performance.

1. INTRODUCTION

Finding similar files is a common use case for search in digital libraries, as in research paper recommendation [1] or near duplicate detection. One of the requirements in similarity search is that the query documents have features in common with the authored documents, which may not always be the case even when documents are semantically similar [2]. This problem is commonly known as feature mismatch and relevance feedback methods based on user feedback or the top k results have been developed for query reformulation. We present a pseudo relevance feedback search strategy for similarity search. Unlike most work in relevance feedback where query reformulation is used based on a set of documents, we instead generate a set of new queries, where each new query is based on one of the top k returned search results. We perform this recursively on each search result and the output of our search is a tree, which we use for ranking.

2. RECURSIVE SEARCH AND RANKING

For an initial query Q that returns a set of results $\mathbb{R}$, the strategy involves recursively searching on the top $k \in \mathbb{R}$ for some recursive depth d and then combining and ranking the results of all searches. The output of the recursive search process is a tree as shown in Figure 1. In the tree, a directed edge from a node n_1 to a node n_2 represents the fact that n_1 was used as a query to retrieve n_2. For instance, the query Q was used to retrieve $r_{1,1}$ and $r_{1,2}$; $r_{1,1}$ was used to retrieve $r_{2,1}$ and $r_{2,2}$; $r_{2,1}$ to retrieve $r_{3,1}$, etc. We use this tree for ranking based on the intuition that if a document

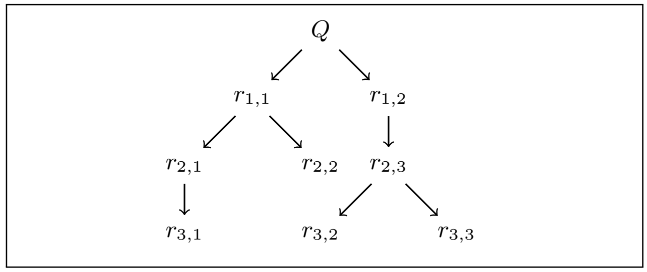

Figure 1: An example of a search result tree produced by the recursive search strategy.

was retrieved using the recursive feedback mechanism and not by the initial query, then it does not have features in common with the initial query and thus its score should be penalized. We propose to penalize search results based on their tree distance from the query document and, by doing this, we aim to account for the problem of performing blind relevance feedback on search results that are not relevant.

Assume a scoring function exists $\varphi(\cdot)$ exists that calculates the similarity between a query document q and a search result r. We then define a set of ranking formulas $\Psi(\varphi, T)$ that assign scores to documents based on both the similarity score φ and the search result tree T produced through the recursive search. Equations 1-5 represent a few simple formulas that are used in this study.

$$\psi_{Flat}(\varphi(q,r),T) = \varphi(q,r)^{\frac{T.depth(r)}{T.depth(r)}} \tag{1}$$

$$\psi_{Power}(\varphi(q,r),T) = \varphi(q,r)^{T.depth(r)} \tag{2}$$

$$\psi_{Decay}(\varphi(q,r),T) = \varphi(q,r)^{1+(1-\frac{1}{T.depth(r)})} \tag{3}$$

$$\psi_{Log}(\varphi(q,r),T) = \varphi(q,r)^{1+log_{10}T.depth(r)} \tag{4}$$

$$\psi_{Div}(\varphi(q,r),T) = \frac{\varphi(q,r)}{T.depth(r)} \tag{5}$$

The design of the ranking functions above is based on our intuition that the tree distance from the query document to a search result is correlated with relevance and that search results should be penalized for having higher tree distances. Thus, all of our ranking functions are designed to capture this in various ways by considering $T.depth(r)$, which is the depth in the tree T at which r occurs. The first ranking function, ψ_{Flat} (Equation 1), does not actually consider the

JCDL '16 June 19-23, 2016, Newark, NJ, USA

ACM ISBN 978-1-4503-4229-2/16/06.
DOI: http://dx.doi.org/10.1145/2910896.2925468

Table 1: Retrieval performance on Reuters and WebKB datasets. d is the recursive depth. $*$ and † represent statistically better performance than Baselines 1 and 2, respectively.

Ranking	Precision@10		MAP	
	d = 2	d = 3	d = 2	d = 3
Reuters				
Baseline 1	0.8591	0.8591	0.2664	0.2644
Baseline 2	0.8677	0.8677	0.4494^{*}	0.4494^{*}
Flat	0.8660	0.8653	0.4567^{*}	0.4569^{*}
Power	$0.8703^{*,\dagger}$	0.8698	0.4545^{*}	0.4543^{*}
Decay	0.8694	0.8689	$0.4592^{*,\dagger}$	$0.4596^{*,\dagger}$
Log	0.8698	0.8688	$0.4573^{*,\dagger}$	$0.4576^{*,\dagger}$
Div	$\mathbf{0.8706}^{*}$	$\mathbf{0.8705}^{*}$	$\mathbf{0.4610}^{*,\dagger}$	$\mathbf{0.4611}^{*,\dagger}$
WebKB				
Baseline 1	0.5127	0.5127	0.0812	0.0812
Baseline 2	0.5161	0.5161	0.1416^{*}	0.1416^{*}
Flat	0.5162	0.5153	$0.1546^{*,\dagger}$	$0.1548^{*,\dagger}$
Power	0.5174	0.5160	$\mathbf{0.1573}^{*,\dagger}$	$\mathbf{0.1612}^{*,\dagger}$
Decay	0.5154	0.5134	$0.1568^{*,\dagger}$	$0.1589^{*,\dagger}$
Log	0.5156	0.5128	$0.1571^{*,\dagger}$	$0.1599^{*,\dagger}$
Div	$\mathbf{0.5182}^{*}$	$\mathbf{0.5167}^{*}$	$0.1543^{*,\dagger}$	$0.1579^{*,\dagger}$

tree structure but instead has the effect of *flattening* the tree and each result is simply assigned its similarity score. The remaining functions all penalize the scores of results based on their tree distance from the query. The difference between them is in the severity of the penalization.

3. EXPERIMENTS

3.1 Method

We use the Reuters 21578 and WebKB datasets and, following other studies [3], we define two documents as being similar if they belong to the same category. The Reuters dataset contains news articles and we use the 10 most popular categories from the Reuters dataset [3]. The WebKB dataset contains categorized University webpages and we use data from Cornell University for testing. Queries are constructed from the documents based on the top 10 TF-IDF ranked terms and cosine similarity is used as the scoring function $\varphi(\cdot)$. We measure Precision@10 and Mean Average Precision (MAP). We consider two baselines. **Baseline 1 - regular search:** regular search is performed with no relevance feedback. **Baseline 2 - query reformulation:** the query is reformulated based on the top 10 TF-IDF ranked terms among the top 10 results for the initial query.

3.2 Results

Results are shown in Table 1. As can be seen from the table, Baseline 1 performs worst overall. All search strategies achieve a higher Precision@10 than Baseline 1 at a recursive depth of 2 and 3. Similarly, all search strategies also achieve better MAP. For the Reuters dataset, all of the recursive search strategies except the *Flat* ranking strategy achieve better Precision@10 than Baseline 2. The reason for this is that the Flat ranking strategy does not consider the tree structure but instead flattens the tree. This provides evidence to support our intuition that penalizing search results that appear lower in the search result tree can lead to better ranking. The two ranking functions that achieve the highest Precision@10 are the Power and Div ranking functions. For the Power ranking function, the mean is significantly better than the mean for Baseline 2 at the 5% level based on a Wilcox signed-rank test. For this dataset, there is very little difference between the Precision@10 for a recursive depth of 2 or 3. We speculate that the reason is that too few results score highly enough to change the top 10 ranked documents.

For the WebKB dataset, the Power and Div ranking function have a higher Precision@10 compared to Baseline 2 and increasing to a depth of 3 leads to a decrease in Precision@10. In some instance, this decrease can be large as is the case for the Log ranking function where there is an over 2% decrease. As was the case with the Reuters dataset, the highest Precision@10 at depth 2 is achieved by the Div ranking method and was 0.52. It's interesting to note that, for both datasets, the Power and Div ranking functions achieve the highest Precision@10. This intuitively makes sense since these methods penalize results that appear lower in the tree more than the other methods. Therefore, the top k results are likely to be very similar to the case of when no recursive search is performed except still include additional documents that are highly similar to the original query document.

For the Reuters dataset, all recursive methods achieve higher MAP than Baseline 2. For the Div, Log and Decay ranking functions, the differences in MAP are significant at the 5% level. There is very little difference between the MAP achieved at a recursive depth of 2 and a recursive depth of 3, which was also observed for Precision@10. The highest MAP at depth 2 is achieved by the Div ranking function. For the WebKB dataset, all recursive methods also achieved higher MAP than Baseline 2. At a recursive depth of 2, all results are significant. For this dataset, increasing the recursive depth to 3 leads to a visible increase in MAP.

4. CONCLUSIONS

This paper presented a strategy for improving performance in similarity search and experiments showed how the recursive search strategies are effective in finding more relevant documents. Furthermore, for most of the experiments, the methods that take the tree structure into consideration perform best, indicating that the structure of the search result tree provides information that can be used to improve ranking. However, that being said, it is likely that the most appropriate ranking formula that takes the search tree into consideration is dependent on the dataset used. We leave investigating this for future work.

Acknowledgments

We gratefully acknowledge partial support by the National Science Foundation.

5. REFERENCES

[1] C. Nascimento, A. H. Laender, A. S. da Silva, and M. A. Gonçalves. A source independent framework for research paper recommendation. In *Proceeding of JCDL*, pages 297–306, 2011.

[2] J. Xu and B. Croft. Query Expansion Using Local and Global Document Analysis. In *Proceedings of SIGIR*, pages 4–11, 1996.

[3] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. *Proceeding of SIGIR*, pages 18–25, 2010.