

Relevance Feedback for XML Retrieval

Yosi Mass and Matan Mandelbrod

IBM Research Lab,
Haifa, 31905, Israel
{yosimass, matan}@il.ibm.com

Abstract. Relevance Feedback (RF) algorithms were studied in the context of traditional IR systems where the returned unit is an entire document. In this paper we describe a component ranking algorithm for XML retrieval and show how to apply known RF algorithms from traditional IR on top of it to achieve Relevance Feedback for XML. We then give two examples of known RF algorithms and show results of applying them to our XML retrieval system in the INEX'04 RF Track.

1 Introduction

Relevance Feedback (RF) was studied e.g. in [6, 7] in the context of traditional IR engines that return full documents for a user query. The idea is to have an iterative process where results returned by the search engine are marked as relevant or not relevant by the user and this information is then fed back to the search engine to refine the query. RF algorithms can be also used for Automatic Query Refinement (AQR) by applying an automatic process that marks the top results returned by the search engine as relevant for use by subsequent iterations.

In [5] we described an algorithm for XML component ranking and showed how to apply existing AQR algorithms to run on top of it. The Component ranking algorithm is based on detecting the most informative component types in the collection and creating separate indices for each such type. For example in the INEX[3] collection the most informative components are articles, sections and paragraphs. Given a query Q we run the query on each index separately and results from the different indices are merged into a single result set with components sorted by their relevance score to the given query Q . We showed then in [5] that we can take advantage of the separate indices and apply existing AQR methods from traditional IR on each index separately without any modification to the AQR algorithm.

In this paper we show how further to exploit the separate indices to achieve Relevance Feedback for XML by applying existing RF algorithms with real user feedback to the base component ranking algorithm. Why is it different than applying Automatic Relevance Feedback? The difference lies in the data on which the relevance feedback input is given. In AQR the process is automatic hence it can be done on the top N results (N is a given constant) of each index separately before merging the results. In a real user feedback scenario the user does the feedback on the merged results and assuming assessment of at most the top N results then we have feedback on N results

from all indices together. It can happen that most of the feedback came from few indices but still we would like to use the feedback to improve results from other indices as well. This is explained in details in section 2 below.

The paper is organized as follows: In section 2 we describe a general method to apply existing RF algorithms on top of the component ranking algorithm. Then in section 3 we discuss two specific RF algorithms and demonstrate their usage to XML retrieval using the method from section 2. In section 4 we report experiments with those algorithms for the INEX RF track and we conclude in section 5 with summary and future directions.

2 Relevance Feedback for XML Retrieval

In [5] we described an algorithm for XML component ranking and showed how to apply existing AQR algorithms on top of it. The base algorithm is described in Fig. 1 below.

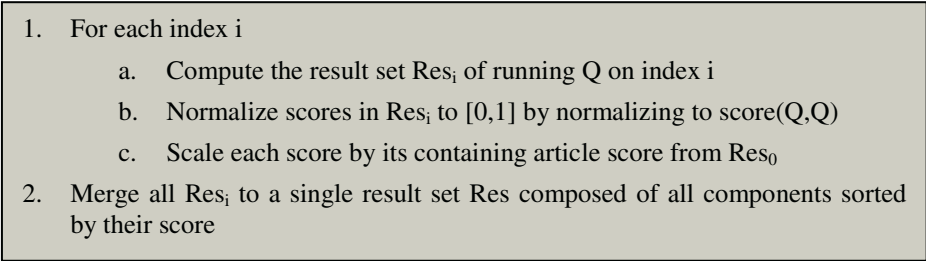
- 
1. For each index i
 - a. Compute the result set Res_i of running Q on index i
 - b. Normalize scores in Res_i to $[0,1]$ by normalizing to $score(Q,Q)$
 - c. Scale each score by its containing article score from Res_0
 2. Merge all Res_i to a single result set Res composed of all components sorted by their score

Fig. 1. XML Component ranking

We brief here the algorithm steps while full details can be found in [5]. In step 1 we run the given query on each index separately (step a) and then normalize result scores in each index to be able to compare scores from different indices (step b). We then apply a DocPivot scaling (step c) and finally in step 2 we merge the results to a single sorted result set of all components.

Since we have separate indices for different component granularities we showed in [5] that we can modify the above algorithm with an AQR procedure by adding a new step between 1.a and 1.b in which we apply the given AQR algorithm on each index separately.

Applying real user feedback is somewhat more complicated than applying AQR since the feedback is given on the merged result set and not on results from each index separately. Assuming that the user can assess at most N results then we have feedback on the top N results from all indices together. Those N results can all come from a single index or from few indices but we still want to use this feedback to improve results from all indices. Moreover we would like to do it without modifying the RF algorithms. We do this by applying the base component ranking algorithm as in Fig. 1 above and then continue with the algorithm in Fig. 2 below.

3. Take the top N results from Res and given their assessments extract R (Relevants) and the NR (Not relevants) from the top N.
4. For each index i
 - a. Apply the RF algorithm on (R, NR, Res_i) with any other needed RF specific params and refine Q to Q'
 - b. Compute the result set Res'_i of running Q' on index i
 - c. Normalize scores in Res'_i to [0,1] by normalizing to score(Q',Q)
 - d. Scale each score by its containing article score from Res'₀
5. Merge all Res'_i to a single result set Res' composed of all components sorted by their score
6. Freeze the original top N from Res as the top N in Res'

Fig. 2. XML Component ranking with RF

The algorithm in Fig. 2 describes a single iteration of applying an RF algorithm on our component ranking algorithm. The algorithm works as follows; In step 3 we use the top N results from the merged results and based on the user feedback we select the subset of relevant (R) and non relevant (NR) components. Note that in a traditional RF algorithm the R and NR components are of same type as the collection (namely, full documents) while here they come from different component types so for each index some of them may be of different granularities than components in that index. We claim that the fact that a component is relevant or not relevant for the query can be used in a typical RF algorithm regardless to its granularity. In the next section we demonstrate two specific RF algorithms and show that at least for them the above claim holds.

So in step 4 we just apply the existing RF algorithm on each of our indices separately where we give it the R and NR components as if they came from that index. The result is a refined query Q' (step 4.a) and then similar to the AQR case the new query is used to generate a new result set Res_i for each index. Results are then scaled by the DocPivot as described in [5] and finally the different result sets are merged (step 5) to a single result set of all component types.

To be able to measure the contribution of an RF iteration over the original query we take in step 6 the seen top N components and put them back as the top N in the final merged result. We then remove them from rest of Res' if they appear there again. In the next section we demonstrate two example RF algorithms that we applied on our XML component ranking using the algorithm from Fig. 2 above.

3 Example Usages of RF Algorithms for XML

In this section we describe two RF algorithms known from IR and we show how we applied them on top of our XML component ranking algorithm.

3.1 Rocchio for XML

The Rocchio algorithm [6] is the first RF algorithm that was proposed for the Vector Space Model[8]. The idea in the Vector Space model is that both the documents and the query are represented as vectors in the space generated by all tokens in the collection (assuming any two tokens are independent). The similarity of a document to a query is then measured as some distance between two vectors, usually as the cosine of the angle between the two.

The Rocchio formula tries to find the optimal query; one that maximises the difference between the average of the relevant documents and the average of the non-relevant documents with respect to the query. The Rocchio equation is given in Fig. 3 below.

$$Q' = \alpha Q + \beta / n_1 \sum_{i=1}^{n_1} R_i - \gamma / n_2 \sum_{i=1}^{n_2} NR_i$$

Fig. 3. The Rocchio equation

Q is the initial query, Q' is the resulted refined query, $\{R_1, \dots, R_{n_1}\}$ are the set of relevant documents and $\{NR_1, \dots, NR_{n_2}\}$ are the non-relevant documents. Since Q , $\{R_i\}$, and $\{NR_i\}$ are all vectors then the above equation generates a new vector Q' that is close to the average of the relevant documents and far from the average of the non-relevant documents. The α , β , γ are tuning parameters that can be used to weight the effect of the original query and the effect of the relevant and the non-relevant documents. The Rocchio algorithm gets an additional parameter k which is the number of new query terms to add to the query.

Note that step 4.a in Fig. 2 above is composed in the Rocchio case from two sub steps; In the first sub step new terms are added to the query and then the original query terms are reweighed. We can therefore apply to the first sub step in the Rocchio case two embedding variants into our XML component ranking –

1. Compute the new query terms only in the main index¹ and use them for other indices as well.
2. Compute a different set of new terms to add for each index separately.

In section 4 we report experiments we did with the Rocchio algorithm in our XML component ranking algorithm.

3.2 LA Query Refinement for XML

In [5] we described a general method to modify our component ranking algorithm with Automatic Query Refinement and described results for an example such AQR algorithm based on [1]. The idea there is to add to the query Lexical Affinity (LA) terms that best separate the relevant from the non relevant documents with respect to

¹ We always assume the the first index contains the full documents.

a query Q . A Lexical Affinity is a pair of terms that appear close to each other in some relevant documents such that exactly one of the terms appears in the query. We summarize here the various parameters that are used in the LA Refinement procedure while full details can be found in [5]. The procedure gets 4 parameters (M, N, K, α) where M denotes the number of highly ranked documents to use for constructing a list of candidate LAs. N ($N \gg M$) is the number of highly rank documents to be used for selecting the best K LAs (among the candidate LAs) which have the highest Information Gain. Those LAs are then added to the query Q and their contribution to $score(Q, d)$ is calculated as described in details in [5]. Since they don't actually appear in the query Q we take their TF_Q to be the given parameter α .

This LA Refinement algorithm can be used with real user feedback and can be plugged as a RF module in our component ranking algorithm as in Fig. 2 above. In section 4 we describe experiments we did with that algorithm for XML retrieval.

4 Runs Description

We describe now the runs we submitted for the RF track - one for the Rocchio implementation and one for the LA refinement.

4.1 Rocchio Runs

As discussed above the Rocchio algorithm is based on the Vector Space scoring model. Since our component ranking algorithm is also based on that model then we could easily plug the Rocchio algorithm into our component ranking as in Fig. 2 above. In our implementation a document d and the query Q are represented as vectors as in Fig. 4 below.

$$\begin{aligned}
 d &= (w_d(t_1), \dots, w_d(t_n)), w_d(t_i) = tf_d(t_i) * idf(t_i) \\
 Q &= (w_Q(t_1), \dots, w_Q(t_n)), w_Q(t_i) = tf_Q(t_i) * idf(t_i)
 \end{aligned}$$

Fig. 4. Document and Query vectors

Where $tf_Q(t)$ is a function of the number of occurrences of t in Q , $tf_d(t)$ is a function of the number of occurrences of t in d and $idf(t)$ is a function of the inverse document frequency of t in the index (exact functions details are described in [5]).

Given α, β, γ we define the Gain of a token t as

$$G(t) = \frac{\beta}{n_1} \sum_{i=1}^{n_1} w_{R_i}(t) - \frac{\gamma}{n_2} \sum_{i=1}^{n_2} w_{NR_i}(t)$$

Fig. 5. Gain of a token

where $w_{R_i}(t)$ are the weights of t in each relevant component R_i and $w_{NR_i}(t)$ are the weights of t in each non-relevant component NR_i as defined in Fig. 4. It is easy to see that tokens with the highest Gain are the ones that maximize the optimal query Q' as defined by the Rocchio equation in Fig. 3 above.

So in the RF step (4.a) in Fig. 2 above we compute $G(t)$ for each new token t in the top N components that is not already in Q . We then select the k tokens with the maximal Gain as the terms to be added to the query Q .

We run a single Rocchio iteration where we compute the new tokens to add only on the main index (first variant from sec 3.1 above). We tried with $N = 20$, $\alpha = 1$, $\beta = \{0.1, \dots, 0.9\}$, $\gamma = \{0.1, \dots, 0.9\}$ and $k = 3$ on our base CO algorithm. The Mean Average Precision (MAP) values we got using the `inex_eval` aggregate metric for 100^2 results are summarized in Fig. 6.

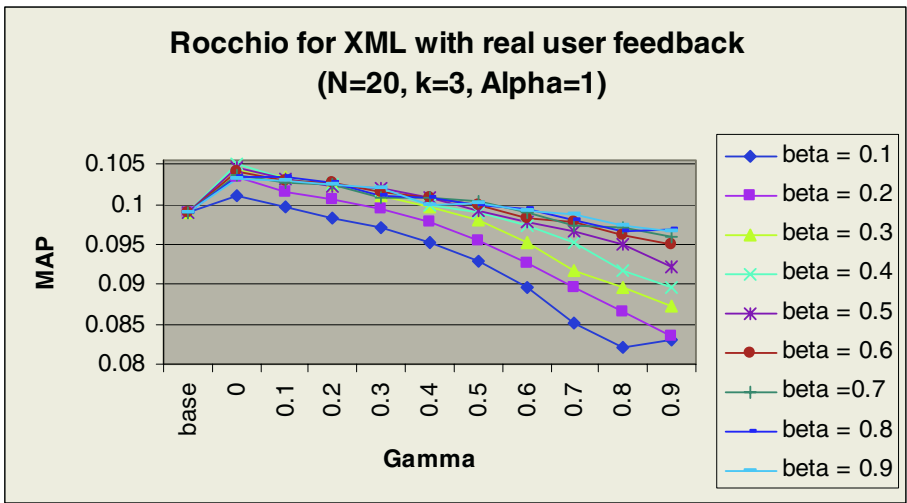


Fig. 6. Rocchio for XML

The Figure shows graphs for the different values. We can see that the value for $\gamma = 0.1$ gives best results for all γ values. The best MAP was achieved at $\gamma = 0.8$ which is what we sent for the RF track.

We see that we get very minor improvement (~5%) over the base run. A possible reason can be the Freezing method of the top N results which leave many possible Non-relevant components in the top N so the effect of RF is only for components at rank $N+1$ and lower.

² Note that the results here are for a run with 100 results so the MAP is lower than the 0.134 value we got in our official INEX submission for 1500 results. Note also that the MAP values in this graph are lower than those achieved in Fig. 7 for LA Refinement as those in Fig. 7 were calculated for 1500 results.

4.2 LA Refinement Runs

Fig. 7 summarizes our experiments with the LA procedure parameters (M, N, K, α). In all experiments we fixed $M = 20$ and we selected out of the top 20 components from our base run those that were assessed as relevant (highly exhaustive-highly specific). We then used those components for extracting the set of candidate LAs to add to the query. The figure below shows are results for fixing $K = 5$ namely we select the best 5 LAs to be added to the query. The figure shows 4 graphs for different values of N (100, 200, 300, 400) and each graph shows the MAP achieved for α values at the range 0.4-1.3.

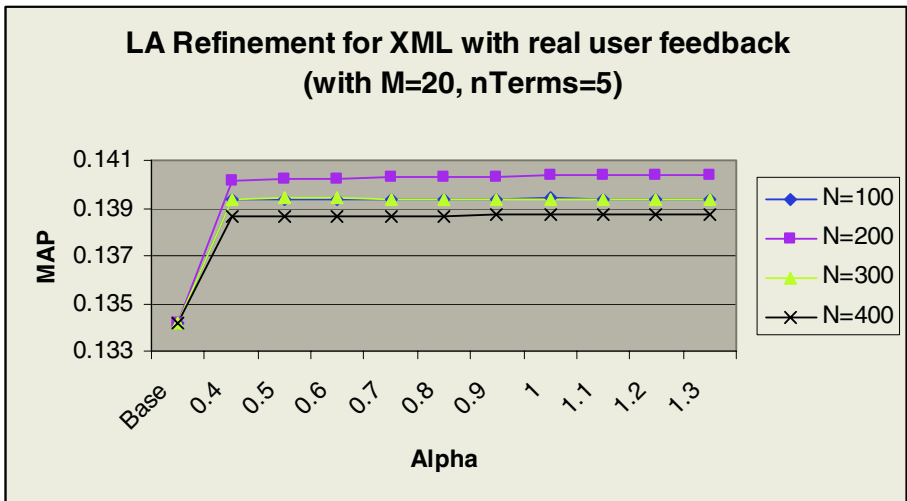


Fig. 7. LA runs with fixed $M=20, K=5$ and varying N, α

We can see that the base run with no RF got $MAP=0.134$ and actually all graphs achieved improvement over that base run. The best results were achieved for $N=200$ (N is the number of documents to use for selecting the best LAs) but it was not significantly different from other parameter combinations we have tried.

5 Discussion

We have presented an XML retrieval system and showed how to run RF algorithms on top of it to achieve relevance feedback for XML. We then demonstrated two example RF algorithms and reported their usage for the XML RF track. We got relatively small improvements in the Mean Average Precision (MAP) with those algorithms and we still need to explore if it's an algorithm limitations or a possible problem in the metrics used to calculate the MAP.

References

1. Carmel D., Farchi E., Petruschka Y., Soffer A.: Automatic Query Refinement using Lexical Affinities with Maximal Information Gain. Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002.
2. Carmel D., Maarek Y., Mandelbrod M., Mass Y., Soffer A.: Searching XML Documents via XML Fragments, SIGIR 2003, Toronto, Canada, Aug. 2003
3. INEX, Initiative for the Evaluation of XML Retrieval, <http://inex.is.informatik.uni-duisburg.de>
4. Mass Y., Mandelbrod M.: Retrieving the most relevant XML Component, Proceedings of the Second Workshop of the Initiative for The Evaluation of XML Retrieval (INEX), 15-17 December 2003, Schloss Dagstuhl, Germany, pg 53-58
5. Mass Y., Mandelbrod M. : Component Ranking and Automatic Query Refinement for XML retrieval, to appear in the Proceedings of the Third Workshop of the Initiative for The Evaluation of XML Retrieval (INEX), 6-8 December 2004, Schloss Dagstuhl, Germany
6. Rocchio J. J. : Relevance Feedback in information retrieval The SMART retrieval system – experiements in automatic document processing, (G. Salton ed.) Chapter 14 pg 313-323, 1971.
7. Ruthven I., Lalmas M. : A survey on the use of relevance feedback for information access systems, Knowledge Engineering Review, 18(1):2003.
8. Salton G. : Automatic Text Processing – The Transformation, Analysis and Retrieval of Information by Computer, Addison Wesley Publishing Company, Reading, MA, 1989.