Measure Based Metasearch

ABSTRACT

In this short paper we propose a simple method for turning any query retrieval performance measure into a metasearch algorithm. While motivation is mainly a better exploration, understanding, evaluation and/or comparison of the IR measures rather than designing a new metasearch algorithms, we also demonstrate good metasearch results in comparison with popular metasearch techniques. In particular, we consider the most common reported measures: precision at standard cutoffs (PC), R-precision (RP) and average precision (AP).

1. INTRODUCTION

The basic idea of our work is that IR measures evaluate performances of runs by "looking" into the list returned (in different ways) for relevant documents returned. The better method is, the more is "looking" at the right ranks and therefore should be able to identify more relevant documents. Therefore any given method, in a multi run setting (like TREC) , can be used for metasearch by identifying the documents at the [averaged] ranks believed to be relevant. The metasearch performance (measured also with AP) will give an insight of the quality of the measure used.

Metasearch is the well-studied process of fusing the ranked lists of documents returned by a collection of systems in response to a given user query in order to obtain a combined list whose quality equals or exceeds that of any of the underlying lists. Many metasearch techniques have been proposed and studied [3, 5, 9, 2, 7]. In this work, we consider two benchmark techniques: the first is based on combining the normalized scores given to each document by the underlying systems (CombMNZ [4, 6]), and the second is based on viewing the metasearch problem as a multi-candidate election where the documents are candidates and the systems are voters expressing preferential rankings among the candidates (Condorcet [8]).

2. METHOD

We now break the method into subparts. R denotes the number of relevant documents in a given query and Z denotes the length of a given retrieved list of documents to be evaluated.

rank weighting scheme. Most IR measures for ranked lists induce a "rank importance" , formally *rank weighting scheme*, when assess the quality of a given list. For example, the PC at cutoff c=10 takes into account only the 10 top ranks with a uniform weighting; therefore it induces equal weights on those 10 ranks and weight=0 for the rest.

Consider now any measure used for evaluating a ranked list of documents. If we write the mathematical formula that produces the measurement, will try to look at this formula as *weighed average* (Z=length of the list measured)

$$measurement = \sum_{i=1}^{Z} w_i \cdot r_i$$

where the vector w is "rank importance" and r has to do with relevance judgments at ranks. Now this may easy for some measures (RP) but definitely challenging for measures that operate not on individual ranks but rather on pairs of ranks or other subsets(AP).

Metasearch. Every document has associated a set of weights (one weight per run), corresponding to the ranks the document is retrieved at in every run. On a document by document basis, the sum (or average) of these weights gives an overall weight of the document, for *that particular weighting scheme* (associated with an IR measure). Ranking the documents by this overall weight we obtain a metasearch algorithm, by outputing top Z (think Z=1000) documents.

We believe this idea can be generalized to most of the measures. Next are details of rank weights for each of the three measures PC —PC, RP and AP— for a generic given list (a run).

precision at cutoffs. Lets fix a cutoff c and denote PC as "precision at cutoff c". Because no recall is involved, it is easy to see that essentially PC puts a uniform distribution of weights over the top c ranks and 0 for all other ranks in the list.

R precision. RP is very similar with PC (use c=R = number of relevant documents in the query), except R is unknown as no relevance judgment is revealed. In this paper we assume that R is "magically" given; that is because, as stated above, our purpose is to investigate the measures and the corelations between them; in an implementation for metasearch purposes, R needs to be estimated or guessed from data.
There is a second fundamental difference between RP and PC. In standard IR settings , like TREC, the performance of anything is usually averaged over many (50) queries. In that case c= PC-cutoff level is a constant over queries while R varies, making the RP-cutoff appropriate for each query

average precision. AP formula is significantly more difficult to see as weighted average across ranks because it operates on pairs of documents (at ranks) . We have

$$AP = \frac{1}{R} \sum_{i \le Z, rel(i)=1} Prec_i =$$
$$\frac{1}{R} \sum_{i \le Z} rel(i) \cdot \frac{1}{i} \sum_{j \le i} rel(j) =$$
$$\frac{1}{R} \sum_{1 \le j \le i \le Z} \frac{1}{i} \cdot rel(i) \cdot rel(j)$$

This sum is a weighted average between $1/i$ vector and $rel(i) \cdot rel(j)$ vector but two vectors are associated with *pairs of ranks* , and not with ranks as we need. For that, we will build a joint distribution(JD) over pairs (i, j) using the $1/i$ vector and then use marginalization to obtain the weights-per-rank vector W. After marginalization, we have

$$W(r) = \frac{1}{2Z}(1 + \frac{1}{r} + \frac{1}{r+1} + ... \frac{1}{Z})$$

3. EXPERIMENTAL RESULTS

We test the metasearch associated with AP,RP and PC for standard cutoffs (5,10,15,20,30,50,100,200,500,1000) using TREC 5,6,7,8,9 (Table 1). We also include the performances of CombMNZ and Condorcet techniques. CombMNZ and Condorcet produce quality ranked lists of documents by fusing the ranked lists provided by a collection of underlying systems. Given ranked lists produced by possibly correlated systems of varying performance, these metasearch techniques will most often produce fused lists whose performance exceeds that of the "average" underlying list but which rarely exceeds that of the best underlying list. We included those two methods performance for a baseline comparison.

First we see once again that AP and RP are strongly correlated. This argument, based on metasearch performance just confirms the correlation, well known by statistical, algebraic and geometrical means.

As for the metasearch performance, both AP and RP demonstrate good results in comparison with previous methods.

While the metasearch for AP is a valid and easy-to implement method, the one coresponding to RP *is not*. Thats simply because R is unknown and critical for this algorithm to work. While the results in the table are useful for analyzing the measures, for practical metasearch purposes one has to come up with a guess for R, for each query.

We also show (Table 2) statistics on R over the TREC queries. Because of the mean at almost all TRECs (except TREC9) is around 100 it is expected that PC with cutoff c=100 does the best of all cutoffs, being relatively close to AP or RP numbers. For PC to work reasonably compared to RP, even at the best c, here c=100, it is necessary that the R values over the queries have a relatively small variation (or standard deviation). In particular, for TREC 9 we see that the ratio $std/mean$ is significantly high , therefore the drop in performance.

TREC	R mean	R std
5	109.9	137.7
6	92.1	103.1
7	93.4	85.1
8	94.5	80.0
9	52.3	84.1

Table 2: R stats over 50 queries

4. CONCLUSIONS AND FUTURE WORK

We showed how to do metasearch based on any given measure for ranked lists. Besides the good metaseach performances, it is a usefull tool in analyzing the measure.

In the near future we plan to test this idea on other measures like b-pref [**?**] or F-measure

5. REFERENCES

[1] *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, Louisiana, USA, Sept. 2001. ACM Press, New York.

[2] J. A. Aslam and M. Montague. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [1], pages 276–284.

[3] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In W. B. Croft and C. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 173–181, Dublin, Ireland, July 1994. Springer-Verlag, London.

[4] E. A. Fox and J. A. Shaw. Combination of multiple searches. In D. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 243–249, Gaithersburg, MD, USA, Mar. 1994. U.S. Government Printing Office, Washington D.C.

[5] J. H. Lee. Combining multiple evidence from different properties of weighting schemes. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 180–188, 1995.

[6] J. H. Lee. Analyses of multiple evidence combination. In N. J. Belkin, A. D. Narasimhalu, and P. Willett, editors, *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–275, Philadelphia, Pennsylvania, USA, July 1997. ACM Press, New York.

[7] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [1], pages 267–275.

[8] M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In K. Kalpakis, N. Goharian, and D. Grossman, editors, *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 538–548. ACM Press, November 2002.

[9] C. C. Vogt. How much more is better? Characterizing the effects of adding more IR systems to a combination. In *Content-Based Multimedia Information Access (RIAO)*, pages 457–475, Paris, France, Apr. 2000.

TREC	MNZ	COND	AP	RP	c5	c10	c15	c20	c30	c50	c100	c200	c500	c1000
5	.294	.307	.300	.308	.254	.265	.275	.280	.284	.288	.294	.285	.258	.237
6	.341	.315	.344	.357	.271	.292	.305	.315	.322	.335	.341	.334	.319	.312
7	.320	.308	.333	.334	.283	.295	.304	.311	.319	.327	.331	.321	.305	.301
8	.350	.343	.370	.366	.254	.286	.304	.313	.330	.351	.357	.343	.323	.305
9	.351	.348	.345	.353	.266	.309	.314	.303	.316	.320	.323	.310	.294	.259

Table 1: comparison of metasearch performances (average precision)