Merging algorithms for Meta-Search

Peng-Fei Li

A thesis submitted for the degree of
Master with Honours in Computing of
The Australian National University

March, 2013

Declaration

To be done. [2]

Mickey M. Mouse
October, 2004

Acknowledgements

I would like to thank my lucky stars, and the cat, for not eating me.

Abstract

This thesis tells a great story about what I achieved in my research project. The abstract is short, but informative. it makes clear the general area in which I worked, and what I achieved.

Contents

List of Figures

Chapter 1

Figures and References

Here's an example of including a figure. Figures need to have size information in them. The only ones I know how to make work are encapsulated postscript files, or eps files. If your figures are not in eps form you may need to find a graphics program which can convert formats. Notice how I can refer to the figure, Fig. 1.1.

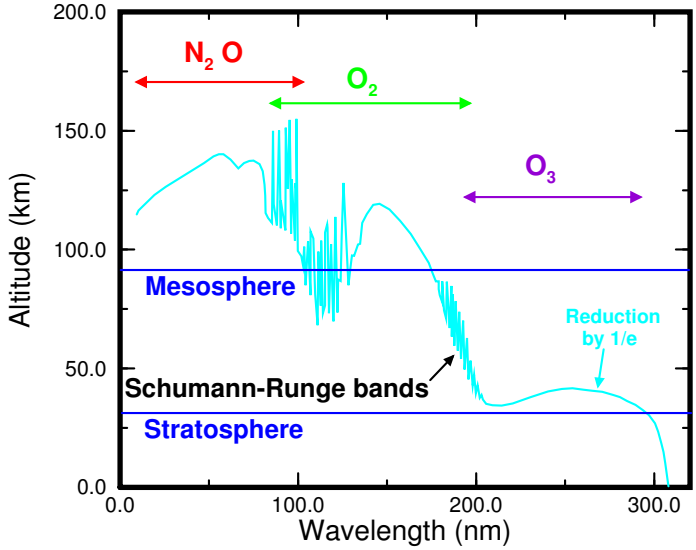

Figure 1.1: This is an example of how to include an eps (encapsulated postscript) figure. Maths is OK in the caption $\epsilon^2 = \Delta$.

Chapter 2

Literature Review

The Internet has grown to be anocean with vast amount of data since its born. Discovering useful information among the massive amount of resources existing on the Internet has been a hot topic for years, which brought the rise of the field of information retrieval. Index centralised searching engines like Google is typical solution to this task. Traditional searching engines use crawlers or spiders to crawl information among the pages and then index the fetched pages. However, this procedure has a disadvantage due to its inability to retrieval information from Deep Web [2]. But as a matter of fact, most of the sites existing on the Internet nowadays are generated dynamically by fetching data from its background data, these data are deep hidden from the surface web that index centralised searching engines can analysis. The size of deep web is much larger than surface web, as pointed out by Bergman [2], and is growing rapidly.

2.1 Meta-Search

An alternative approach is to provide an integrated search engine that can send users' query to different search engines and then return the merged result. This method leaves the searching job to each component search engine, built into the document collection, which can retrieval its documents through its internal database. This approach is named distributed search [4], sometimes named federated search [9] or metasearch. A typical meta-search engine grabs users' query request, do some query expansion, which is optional, and then sends the query to different component search engines, finally the engine collection the results returned by all the collections and merging the resulting in a suitable ranking to display to users (Figure 2.1).

2.2 Collection Selection

An efficiency issue may arise when the meta-search engine have to search across massive amount of collections. As a matter of fact, because some collection are mainly focused on a particular topic, many of the collections will contain limited number of documents that are relevant to the users' query, or even no documents will return in some extreme cases. Due to the limitation of memory, bandwidth, time and other resources, it would be a good idea to ignore some collections that are likely to be the last ones to have relevant documents with respect to a particular query or topic in semantic aspect. That problems has been defined as collection selection problems and have attracted many attentions by scientists in this area. Early meta-search engines apply manually cluster collections into different themes or topics and then assign query to different clusters according to its topic,

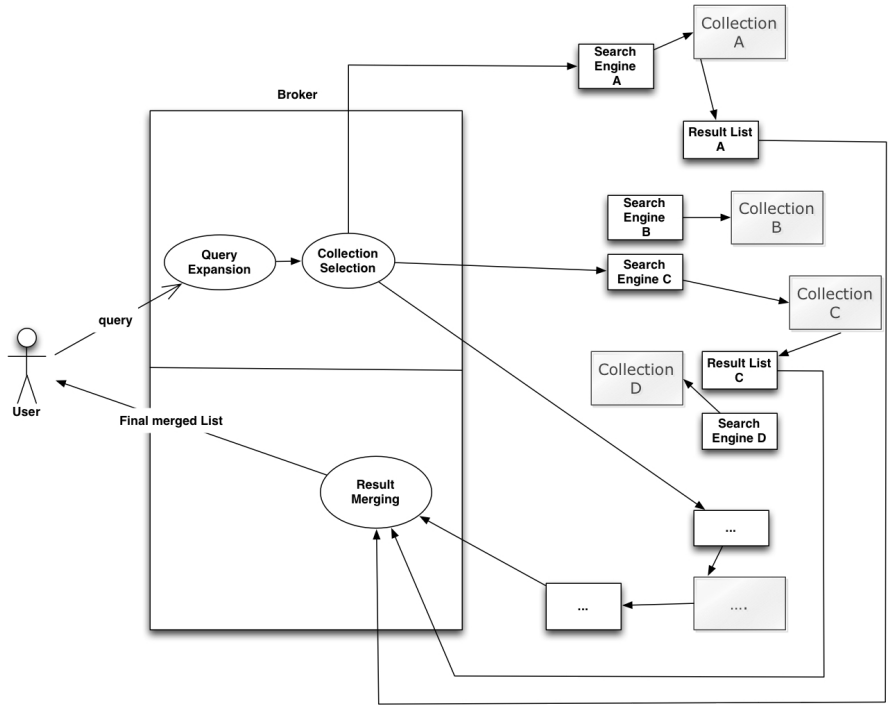

Figure 2.1: A typical architecture of metasearch, the users' requests are fetched by the main component, which is named broker, some query expansion may be executed and the processed query will send to different collections. The collections execute searching on the query and return the retrieval results to the broker and then the

it has already been tried to group collection automatically [6]. Later attempts have been made to treat each collection as a big document and then calculate the similarity between the collection and the query using the bag of words model. In this model, each collection is treated as a big document and other statistic information is stored for later use. A vector space model can be build using the big bag words model, in which all collections as well as queries are built into vectors. Each element into the vector represent the weight of a term to a collection or a query. The weight is often set by the TF-IDF weight which maybe obtained from the cooperative collection statistics or estimation and is used to calculate the similarity of documents or queries. Cosine similarity and Okapi BM25 [14] are widely used to get the similarity of a collection and a query, which is also regard as the collection score in these algorithms. There are also many algorithms based on the lexicon information of the collection. For example, CORI [3, 4] is developed based on an information retrieval system named INQUERY [5], which builds an inference network to search information in a collection. The CORI system calculate the belief of a collection associated to a query and rank them according to the score. Besides all these lexicon based algorithms, document surrogate methods [16] [15] [18] and machine learning approaches [21]have also come into use in this area.

2.3 Result Merging

Although collection is a significant issue in many systems and has drawn many attentions recent years, it is still a prior procedure of metasearch. It is quite important factor with related to the speed of the system as well as precision of the final result. However, in some

cases when the number of collection is not very big, it may be a fact that the collection selection procedure may decrease the efficiency as well as precision since its incomplete searching. As the results are returned from different sub-components, it becomes another issue how to put them together into an integrated result list. Documents in these results lists are distributed differently and there is limited information about the result document in most cases. This problem has been defined as result merging or ranking fusion problems. A formal definition has been given to define ranking fusion problems by [7] and [13]. Given a set of items denoted by U, τ is an ordered list of the elements in a subset of U, which is denoted by S, for example, τ=[$x_1 \geq x_2 \geq \cdots \geq x_k$], where $x_i \in S$. If τ contains all the elements of U, tau is said to be a $fulllist$. However, full lists are not possible in most cases because the searching engines will not return all the items in the database as a limitation of resources. Assume a set of rank lists R=$\{\tau_1,\tau_2,\tau_3,\cdots,\tau_n\}$, the result merging or rank fusion problem is to find out a new rank list, denoted by $\hat{\tau}$, which is the result of a rank fusion method applied to the rank lists in R.

2.3.1 Round-Robin

Previous meta-search merge the different lists into a unified one in a simple Round-Robin manner [12]. The round-robin algorithm is based on the assumption that the number of relevant documents is approximately the same and distributed evenly in all the collections [12] [11]. But the assumption cannot hold in a real world, where the number of relevant documents varies and the distribution is extraordinarily messy. Rasolofo et.al. [12]also proved that the round-robin method perform various randomly. It's may be a better idea to give weight to different collection to determine the order of collections when doing round-robin, which can make the method more stable. So they added some features to each collection to form a biased round-robin algorithm that different collections have collection scores showing their preference in each round-robin circle [12] and saw some precision increase.

2.3.2 Score-based Methods

It may be an advantage of knowing score information of each document in a collection, document score can be easily reused to get them into the right place into the universal list. Because document scores in different collections are always not comparable, a new score function should be generated to return comparable scores. Normalisation is approach used by many algorithms to generate the final score, for example, [13] introduced two basic normalisation function, $Scorenormalisation$(euqation2.1) and $Z-scorenormalisation$(equation 2.2) .

$$w^{\tau}(i) = \frac{s^{\tau}(i) - min_{j\in\tau} s^{\tau}(j)}{max_{j\in\tau} s^{\tau}(j) - min_{j\in\tau} s^{\tau}(j)} \tag{2.1}$$

$w^{\tau}(i)$ indicates the normalised weight of item $i \in \tau$, and the score of an item assigned by τ is denoted by $s^{\tau}(i)$

$$w^{\tau}(i) = \frac{s^{\tau}(i) - \mu_{s^{\tau}}}{\sigma_{s^{\tau}}} \tag{2.2}$$

$\mu_{s^{\tau}}$ denotes the means of scores and $\sigma_{s^{\tau}}$ is the standard deviation. In most cases, the score of each document is not available to meta-search systems. An alternative way is to estimate the score according to accessible information. [12] utilised various document fields to estimate the document score. They generate the document score using a generic

document scoring function(Equation 2.3).

$$w_{ij} = \frac{NQW_i}{\sqrt{L_q^2 + LF_i^2}} \tag{2.3}$$

NQW_i is the number of query words appearing in the processed field of the document i, L_q is the length (number of words) of the query, and LF_i is the length of the processed field of the document i. Then they use the score as the bases for their two new algorithms, namely SM-XX and RR-XX. The SM-XX algorithm use the estimated document score as the unified document score and merge all the documents in the order of the score. The other algorithm, use the score to re-rank the results in each collection and use the round-robin method to merge the new result lists. The "XX" stands for the document fields such as document title, document summary. Although it shows that the final result of RR-XX is better than original round-robin algorithm, it still leave the distribution of documents in each collection as a problem.

Linear combination is another approach to get the final score of documents, for example, Fox and Shaw [8] developed 6 methods to normalise the overall similarity across all the individual searching systems based on SMART by combining the scores across all the collections. These 6 algorithms(Table 2.1), uses aggregation methods to estimate the

Table 2.1: Comb Algorithms

Name	Similarity
CombMAX	MAX(Individual Similarities)
CombMIN	MIN(Individual Similarities)
CombSUM	SUM(Individual Similarities)
CombANZ	Number of Nonzero Similarities
CombMNZ	SUM(Individual Similarities)* Number of Nonzero Similarities
CombMED	MED(Individual Similarities)

location of a document among all the item sets. As discussed by the authors, is trying to eliminate the two types of primary errors in ranking methods, namely assigning a relatively high rank to a non-relevant document and assigning a relatively low rank to a relevant document. However, it's an obvious fact that both CombMIN and CombMAX combination method bring opposite effects. For CombMIN method, it can eliminate the possibility that a non-relevant document being assigned a high rank, however, the problem that CombMAX trying to fix has been amplified. The same effect exits when applying CombMIN method. And they also conduct some experiments all the combination algorithms and found that CombMNZ is the performs best among all the others. These score based algorithms can achieve great efficiency in precision and recall when the overlap between collection is very large and has been used as a baseline algorithm for many other papers. However, in real world application, the result returned by different collections are seldom having duplication, which means many of the documents will appear once in all the collections. The fact will bring very challenging difficulties to the combination methods. [19] also proposed a linear combination model named LC Model, which takes the weight of collection into consideration.

2.3.3 Rank-based Methods

Although in most cases, the score information of each document are not given in most scenarios, or cost too much complexity to estimate, ranking information of a document in a collection is presented in an obvious way, like is the order in the result list. Many algorithms are based on the ranking score of document, for example, Borda-fuse [1] was a voting procedure popularly used in Election scenario, and was modified to apply to our metasearch systems. In this model, servers are acting as the role of voter, and each document is just like the candidate. Each server or retrieval system holds its own preference of ranking on all the documents. And then, the top ranked documents are assigned c points for candidates whose size is c, and c-1 points for the next, etc. Then we sum the score of each document and rank them according to the score. Due to the facts that different server may have different weight on different topics, they assigned weights to each server and then modified the method to a weighted borda-fuse algorithm. This model, works quite well for the system, which has many overlap in the documents. However, in reality, the repositories may vary quite differently. In that case, each document may just appears once in each rank list, which means they may distributed evenly among all the searching systems and has a lot of documents with the same scores. So the model cannot work well in the situation when there is little intersection within documents. And as a fact, their experiments showed that the new algorithm cannot outperform the baseline algorithm, but in most case, can perform better than the best-input system. [13] also proved that Borda-fuse algorithm is not competitive with score-based algorithms. [10] [7] to be described...

2.3.4 Collection weight

It is a fact that different collection have a different weight on different queries or topics, for example, an electronic online shop is more likely to contain information about a PC than a bookshop website. The metasearch system should be a bias decider on different collection both on semantic relationship and the volume of results they returned. Many of the previous algorithms have weighted versions. CORI [3, 4] uses a simple linear combination method to normalise the collection score(equation 2.4),

$$C' = \frac{C - C_{min}}{C_{max} - C_{min}} \tag{2.4}$$

, the normalised score is then used by the raw score based function to estimate the document score by equation 2.5:

$$D' = \frac{D + 0.4 \times D \times C'}{1.4} \tag{2.5}$$

Rasolofo et.al. [11]proposed a new approach to rank the list from different collections names LMS(using result Length to calculate Merging Score). They estimate the collection score by the The algorithm uses the result length, namely the number of documents retrieved by the collection using equation 2.6. l_i is the number of documents returned by a collection and K is a constant set to 600. C is the number of collections.

$$S_i = \log 1 + \frac{l_i \times K}{\sum_{j=1}^{|C|} l_i} \tag{2.6}$$

The collection score was then used to generate a collection weight.

$$w_i = 1 + [(s_i - \bar{s})/\bar{s}] \tag{2.7}$$

where s_i is the ith collection score calculated by the previous formula and $\hat{s}$ is the mean collection score. This approach is like a simplified version of CORI, which just take the size of relevant documents into account. So more weight will be put on the collections with more result documents. This is not always true because bad collections can also return many irrelevant documents. So it is not accurate to use the size of returned documents as the only criteria.

2.3.5 Machine Learning Approaches

Apart from these score-based or rank-based algorithms, machine learning approaches has been considered to be a possible way to solve the problem. [1] proposed algorithm based on the Bayes's theorem. They built a probabilistic model which is used to estimate the probability of the relevance of a document to a query. Given a document d, $r_i(d)$ is the rank assigned by the ith collection to a query. So the probabilities of a document is relevant and irrelevant given the ranking r_1,r_2,$\cdots r_n$ are $P_{irr} = Pr[rel|r_1, r_2, r_3 \cdots, r_n]$ and $P_{irr} = Pr[irr|r_1, r_2, r_3 \cdots, r_4]$. Odds(equation 2.8) of relevance is widely used as a measurement to compute the possibility.

$$O_{rel} = P_{rel}/P_{irr} \tag{2.8}$$

By Byes rules, we can get

$$\begin{aligned} P_{rel} &= \frac{Pr[r_1, r_2, r_3 \cdots, r_n|rel] \cdot Pr[rel]}{Pr[r_1, r_2, r_3 \cdots, r_n]} \\ & \qquad\qquad and \\ P_{rel} &= \frac{Pr[r_1, r_2, r_3 \cdots, r_n|irr] \cdot Pr[irr]}{Pr[r_1, r_2, r_3 \cdots, r_n]} \end{aligned} \tag{2.9}$$

, so we can compute the odds by

$$\begin{aligned} O_rel &= \frac{Pr[r_1, r_2, r_3 \cdots, r_n|rel] \cdot Pr[rel]}{Pr[r_1, r_2, r_3 \cdots, r_n|irr] \cdot Pr[irr]} \\ &= \frac{\prod_i Pr[r_i|rel] \cdot Pr[rel]}{\prod_i Pr[r_i|irr] \cdot Pr[irr]} \end{aligned} \tag{2.10}$$

taking log on both side and we can get a new formula:

$$\log O_{rel} = \sum_i \log \frac{Pr[r_i]|rel}{Pr[r_i|irr]} + \log \frac{Pr[rel]}{Pr[irr]} \tag{2.11}$$

because $\frac{Pr[rel]}{Pr[irr]}$ is a common term for a document, which takes the same value so we can drop it and get the final formula

$$rel(d) = \sum_i \log \frac{Pr[r_i(d)|rel]}{Pr[r_i(d)|irr]} \tag{2.12}$$

$Pr[r_i(d)|rel]$ represent the probability that a relevant document would be ranked at level r_i by system i, while the $Pr[r_i(d)|irr]$ is the probability that an irrelevant document would be ranked at level r_i by system i. The model sounds reasonable in theory but hard to implement in reality. In the paper, the author used the trec_eval to calculate these probabilities by human, which definitely cannot be applied to real world scenario. It may be practical to use training data to get these probabilities in real world. But, as the collections keep changes all the time and the training will take a vast mount of time, its hard to guarantee the efficiency and precision. Another example of machine learning approach is done by [21]. :KNN,CLUSTERING(To be continued). [13]:Markov chain(to be continue)

2.4 Evaluation

As a matter of fact, it is a tough task to judge the performance of an information retrieval system. Many meta-searchers have trying to persuade others that their algorithms are better, but none of these algorithms can perform stably in every situation. And others are trying to find good ways to do these judgments on these algorithms. For lots of algorithms developed before, it is a common way to test the performance based on the Test Collection like TREC [20]. However, as discussed by [17], the test collection approach is lack of private data and will not evolve as a real data source. Another approach is to judge the performance based on search log like click-through data. This approach is based on the hypothesis that high-ranked documents tend to have more clicks. However, the hypothesis does not always hold in reality. On the other hand, the search log is not easy to achieve in an un-cooperative environment. Other methods like Human experimentation in the lab as well as naturalistic observation are widely used in the practice and they both have their drawbacks. [17] also implemented a tool to evaluate the performance based on embedded comparison and log analysis.

Bibliography

[1] Javed a. Aslam and Mark Montague. Models for metasearch. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '01*, pages 276–284, 2001.

[2] MK Bergman. The deep web: Surfacing hidden value. *Bright Planet*, pages 1–17, 2001.

[3] James P Callan, Zhihong Lu, and W Bruce Croft. Searching Distributed Collections With Inference Networks. *ACM SIGIR-95*, pages 21–28, 1995.

[4] Jamie Callan. Distributed information retrieval. In *Advances in Information Retrieval*, pages 127–150. 2000.

[5] JP Callan, WB Croft, and SM Harding. The INQUERY retrieval system. *Database and Expert Systems ...*, 1992.

[6] Peter B Danzig, Jongsuk Ahn, John Noll, and Katia Obraczka. Distributed Indexing : A Scalable Mechanism for Distributed Information Retrieval 1 Introduction and Motivation. pages 220–229, 1991.

[7] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the Web. *Proceedings of the tenth international conference on World Wide Web - WWW '01*, pages 613–622, 2001.

[8] Edward A Fox and Joseph A Shaw. Combination of Multiple Searches. In D K Harman, editor, *The 2nd Text Retrieval Conference TREC2 NIST SP 500215*, volume 500-215 of *NIST Special Publication*, pages 243–252. NIST, National Institute for Standards and Technology, NIST Special Publication 500215, 1994.

[9] Péter Jacsó. Thoughts about federated searching. *Information Today*, 21(October 2004):7–9, 2004.

[10] Joon Ho Lee. Analyses of multiple evidence combination. *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR 97*, 31(SI):267–276, 1997.

[11] Yves Rasolofo, F Abbaci, and J Savoy. Approaches to collection selection and results merging for distributed information retrieval. *... international conference on Information ...*, 2001.

[12] Yves Rasolofo, David Hawking, and Jacques Savoy. Result merging strategies for a current news metasearcher. *Information Processing & Management*, 39(4):581–609, July 2003.

[13] M Elena Renda, Via G Moruzzi, and Umberto Straccia. Web Metasearch : Rank vs . Score Based Rank Aggregation Methods Categories and Subject Descriptors. 2003.

[14] SE Robertson and S Walker. Okapi at TREC-3. *Proceedings of the Third Text REtrieval Conference (TREC 1994). Gaithersburg, USA, November 1994.*, 1994.

[15] Milad Shokouhi. Central-Rank-Based Collection Selection in Uncooperative Distributed Information Retrieval. *Advances in Information Retrieval*, pages 160–172, 2007.

[16] Luo Si and Jamie Callan. Relevant document distribution estimation method for resource selection. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval - SIGIR '03*, page 298, 2003.

[17] Paul Thomas and David Hawking. Evaluation by comparing result sets in context. *Proceedings of the 15th ACM international conference on Information and knowledge management - CIKM '06*, page 94, 2006.

[18] Paul Thomas and M Shokouhi. SUSHI: scoring scaled samples for server selection. *Proceedings of the 32nd international ACM ...*, 2009.

[19] Christopher C Vogt and Garrison W Cottrell. Fusion Via a Linear Combination of Scores. 27:1–27, 1998.

[20] E Voorhees and DK Harman. *TREC: Experiment and evaluation in information retrieval*. 2005.

[21] EM Voorhees, NK Gupta, and B Johnson-Laird. Learning Collection Fusion Strategies. *Proceedings of the 18th ...*, pages 172–179, 1995.