Merging algorithms for enterprise search

PengFei (Vincent) Li
Australian National University
u4959060@anu.edu.au

Paul Thomas
CSIRO and
Australian National University
paul.thomas@csiro.au

David Hawking
Funnelback Pty Ltd and
Australian National University
david.hawking@acm.org

ABSTRACT

Effective enterprise search must draw on a number of sources—for example web pages, telephone directories, and databases. Doing this means we need a way to make a single sorted list from results of very different types.

Many merging algorithms have been proposed but none have been applied to this, realistic, application. We report the results of an experiment which simulates heterogeneous enterprise retrieval, in a university setting, and uses multi-grade expert judgements to compare merging algorithms. Merging algorithms considered include several variants of round-robin, several methods proposed by Rasolofo et al. in the Current News Metasearcher, and four novel variations including a learned multi-weight method.

We find that the round-robin methods and one of the Rasolofo methods perform significantly worse than others. The GDS_TS method of Rasolofo achieves the highest average NDCG@10[1] score but the differences between it and the other GDS_ methods, local reranking, and the multi-weight method were not significant.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and software—*performance evaluation (efficiency and effectiveness), distributed systems*.

Keywords

Federated search; information retrieval; results merging

1. INTRODUCTION

Enterprise search provides information access across an enterprise—for example a company, government body, or academic institution. Although relatively little-studied, this is a significant problem: a large majority of enterprises have ineffective search of their own online information [9, 17] and it contributes to significant wasted time—one to two hours per day for white-collar workers—as well as financial loss [17].

Most search tools, including large-scale web search engines, crawl a document collection and create a single index, which is then used to respond to queries. However, a typical enterprise will have information spread across a large number of sources: contact information might be in an address book, exposed via LDAP or over the web; public web pages will describe some activities or products; internal web pages may discuss policies and procedures; information on customer engagements may live in its own specialised repository; calendars and network drives may be available to share; the enterprise may make use of external resources such as subscription databases or hosted software; and so forth.

In many cases these sources will not be accessible to a crawler as there will be no way to enumerate all records (e.g. databases with search, but no browse, interfaces); or access will be restricted by agreement (e.g. commercial software with licence restrictions or undocumented data formats); or for practical reasons (e.g. pay-per-access databases run by third parties). This rules out the conventional gather-and-index model and requires that we use techniques from *federated search* ("metasearch", "distributed search").

Rather than use a single index, federated search aggregates several independent sources behind a single search interface. Each source has an associated search engine, which is typically considered a black box: at query time, each search engine is interrogated independently and the result sets combined before being returned to the user.

This alleviates problems of distribution, heterogeneous data, and access restrictions. However, a federated system must (at a minimum) consider server discovery, how to find the servers to include; server characterisation, how to represent the holdings and extent of each server; server selection, how to identify the appropriate servers for each query; translation to and from each server's API or other interface; and result merging, how to combine the results from independent servers into one ranked list. For an overview of federated algorithms see e.g. Shokouhi and Si [14]. In this paper we concentrate on the problem of merging.

1.1 Result merging

Result merging is possibly the most important outstanding problem in federated search. The problem can be summarised thus: *given a set of results from each of a set of*

[1]Normalised Discounted Cumulative Gain to rank 10. See [4].

ADCS '13, December 05 - 06 2013, Brisbane, QLD, Australia
Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM 978-1-4503-2524-0/13/12 ...$15.00
http://dx.doi.org/10.1145/2537734.2537750.

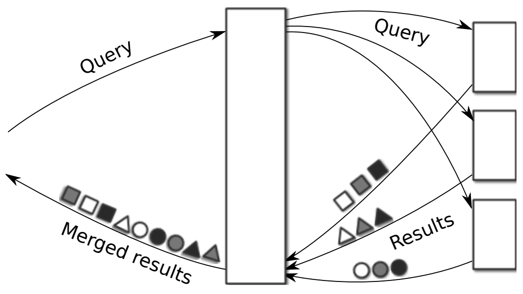


Figure 1: An enterprise search engine must merge results from different sources to produce a single ranked list. The results may be heterogeneous, and cooperation from the sources may be minimal or non-existent.

servers, produce a single ranked list (Figure 1).[2] Note that this formulation allows many variants. We need not respect the original ordering from each server (we can rank a server's second result ahead of its first, in our final output), but we may take account of the query, the result presentations (snippets, URLs, scores, ranks), the referenced documents (full text), and/or server characteristics (scale, scope, effectiveness). Importantly too we cannot rely on cooperation from servers: that is, we cannot assume anything more than a query-result interface. This limited interface means we cannot make use of language models or size estimates, unless we use sampling techniques [16], and if servers return document scores then we cannot assume they are on the same scale.

Many merging algorithms have been described. However, the evaluations to date may not tell us anything about their effectiveness in an enterprise setting. Performance of many algorithms varies considerably between collections, but more importantly none of the standard testbeds accurately represents enterprise contexts: typical evaluations have relied on one or more of the TREC ad-hoc or web test collections. These testbeds are homogeneous, in that they include only one document type (semi-structured text or HTML); often have subcollections or "servers" of uniform size; and do not treat typical enterprise content or information needs.

"Aggregated" search—as seen in e.g. some web search engines [5]—does deal with differing result types, but merging is still poorly understood: to the best of our knowledge there is no shared testbed, servers are typically completely cooperative (they are run by the same people), and documents and information needs on the public web will be different to those in an enterprise.

1.2 A university example

We illustrate these ideas with reference to search at the Australian National University (ANU). Like many large organisations, the ANU publishes information over a number of channels, for a number of audiences, and in a number of formats including front-ends to structured databases (such as the telephone directory and library catalogue), calendars, and semi-structured pages from more than 500 different web servers. Individual units in the university add email lists, file shares, local web servers, and other local resources; and the university also makes use of third-party services such as Twitter, Facebook, and YouTube to reach a wider audience or for specialised hosting.

These sources are very different in size (from quite small to more than a million documents), subject matter, language, intended audience (staff, potential or current students, researchers, funders, the public), interface, search capability (ranked or Boolean searches of different quality and with varying, or no API), and control (the ANU runs its own web search, but must rely on third parties to search e.g. YouTube). This is, however, a realistic case of enterprise search: certainly people at or interested in the ANU would like to search these sources, and we argue that other enterprises will look similar at least in their diversity.

Again, the range of sources and the restrictions on access make it impossible to build a central index; and yet a "single search box" would be useful, especially for users not familiar with each source. So if we were to build a federated search system to cover all these sources, how should we handle result merging? This paper describes an experiment to evaluate algorithms and understand our choices.

2. METHODS

We examined the merging problem with an ad-hoc-style test collection based on the university example above. Ten merging methods were considered and evaluated on NDCG and precision at 10.

2.1 Test collection

Our test collection comprises 44 queries which might be issued to a university search engine; eleven sources, with varied characteristics; and a total of 3000 documents judged on a five-point scale.

Queries. Queries were created by the authors, with reference to common queries on the existing ANU search services. They are intended to cover a range of information needs and include searches for:

- people at, or connected with, the ANU (13 queries such as "H.C. Coombs" or "Brian Schmidt");
- links with other organisations (8 queries including "NICTA" and "NHMRC");
- material for a press release (7 queries across varied topics);
- regional connections (5 queries including "South Korea" and "China");
- potential for research collaboration (7 queries across varied topics); and
- university departments and facilities (4 queries such as "ANU library" and "fenner school").

Table 1 shows a sample of the queries used. The average length across the whole set was 2.15 words, ranging from one to three words, with a single outlier of nine words.

We believe these are a realistic representation of enterprise search needs at a university. Some are best answered by a single document, from a single source (e.g. staff names, "student admin"); others are more complex and require synthesis across several sources (e.g. "big data", "indigenous languages").

[2]This corresponds to Shokouhi and Si's "federated search merging" or "metasearch merging", not "fusion" [14].

```
afghanistan
South Korea
Student Admin
murray darling basin
Alistair Rendell
climate change
Sir Mark Oliphant
South Korea
NICTA
Sir Howard Florey
China
murray darling basin
```

Table 1: Twelve queries chosen at random from the set of 44.

Internal:	
ANU web	University web pages
ANU contacts	Searchable staff list and contact details
ANU map	Building index and department locations
ANU library	Complete library catalogue
StudyAt	Course catalogue
ANU research (×3)	Three services: lists of researchers, research projects, and research publications
D-SPACE	Digital publications
External:	
Twitter (external)	ANU Twitter feed
YouTube (external)	ANU YouTube channel

Table 2: The sources included in this testbed cover a range of size, data type, audience, search interface, and publishing method. Note that we are unable to provide accurate counts of the number of documents indexed at each source, since we are accessing sources only via their search interface. We understand that the ANU web includes approximately one million documents, that the ANU library catalogue includes several million items and that the other sources are much smaller.

Sources. Table 2 lists the eleven sources used in our experiment. The majority are in-house search engines, which cover a range of document types and search interfaces, but the university also publishes material on YouTube and Twitter where it must rely on the search engines provided. The "ANU web" source combines more than 500 web sites run by the central administration or by individual departments, research groups, or staff using a wide variety of publishing technology.

Each query was issued to each source, without any advanced features or any modification. We recorded the reported size of the total result set ("n results found") as well as the first page of results.

Documents. Unsurprisingly given the range of retrieval methods, APIs, and holdings, the result sets vary widely in size and quality. The ANU contacts list for example returned no results to a majority of queries, and never more than 14; whereas the research publications source has a very liberal search engine and reported 3758 results on average (and up to 48,330). There was also considerable variation across queries, with a mean 11,150 results per query but a range of 142 ("Rolf Zinkernagel") to 59,024 ("ANU Library"). There were no results for 124 of the 484 query × source combinations.

We formed the final pools by downloading the top ten results, from each source, to each query (or downloading fewer if there were fewer than ten results). This gave us a pool of 44 queries and 3000 documents from 11 sources.

There is minimal overlap between these collections. The ANU web indexes a small subset of the three research collections, but otherwise documents are almost completely partitioned. On our test queries, in 27 of 44 cases there was no overlap in the pool; on average there was only one overlapping document.

Relevant documents were distributed unevenly across the collections, from an average 0.1 relevant documents per query in the top ten (the ANU map) up to an average of 7.3 (the library).

Participants and judgements. Relevance judgements were provided by 20 participants recruited by word of mouth at the ANU. Since we are simulating enterprise search for the university, the participants have relevant background knowledge and are at least "silver standard" judges by the standard of Bailey et al. [1].

Participants were shown titles of all results for a query and asked to mark each result on a five-point scale, with instructions as follows:

- "*Irrelevant* is the score given to documents that are totally irrelevant to the topic.
- *On-topic but useless* is the score given to documents that are related to the topic but are talking about some other aspect of the area.
- *Somewhat useful* is the score given to documents that are relevant to the topic, and the topics that the queries are intended to retrieve are mentioned in the documents.
- *Useful* is the score given to documents where most, or at least some part of the documents are talking about the intended information.
- *Comprehensively useful* is the score given to the documents that are talking exactly about the topic."

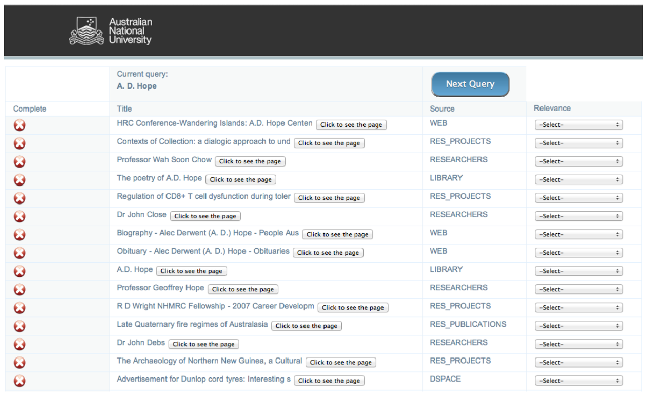
The full text of each result document was available if needed, and each query was judged by a single participant. Figure 2 illustrates the judging interface.

2.2 Measures

We report two measures of each algorithm: NDCG [4] and precision, both calculated at rank 10. Gains for NDCG were 0 for "irrelevant", then 1, 2, 3, to 4 for "comprehensively useful". Precision requires binary judgements and we present these results for different thresholds of relevance.

3. ALGORITHMS

We considered ten merging algorithms: six algorithms from the literature, and four novel variations. Several of



Australian National University

Current query:
A. D. Hope

Complete	Title	Source	Relevance
✗	HRC Conference-Wandering Islands: A.D. Hope Centen [Click to see the page]	WEB	-Select-
✗	Contexts of Collection: a dialogic approach to und [Click to see the page]	RES_PROJECTS	-Select-
✗	Professor Wah Soon Chow [Click to see the page]	RESEARCHERS	-Select-
✗	The poetry of A.D. Hope [Click to see the page]	LIBRARY	-Select-
✗	Regulation of CD8+ T cell dysfunction during toler [Click to see the page]	RES_PROJECTS	-Select-
✗	Dr John Close [Click to see the page]	RESEARCHERS	-Select-
✗	Biography - Alec Derwent (A. D.) Hope - People Aus [Click to see the page]	WEB	-Select-
✗	Obituary - Alec Derwent (A. D.) Hope - Obituaries [Click to see the page]	WEB	-Select-
✗	A.D. Hope [Click to see the page]	LIBRARY	-Select-
✗	Professor Geoffrey Hope [Click to see the page]	RESEARCHERS	-Select-
✗	R D Wright NHMRC Fellowship - 2007 Career Developm [Click to see the page]	RES_PROJECTS	-Select-
✗	Late Quaternary fire regimes of Australasia [Click to see the page]	RES_PUBLICATIONS	-Select-
✗	Dr John Debs [Click to see the page]	RESEARCHERS	-Select-
✗	The Archaeology of Northern New Guinea, a Cultural [Click to see the page]	RES_PROJECTS	-Select-
✗	Advertisement for Dunlop cord tyres: Interesting s [Click to see the page]	DSPACE	-Select-

Figure 2: The interface used to collect relevance judgements. Participants were asked to judge all documents for a query, and documents were presented in a random order. Full text was available if needed, via the "see the page" button.

the best-performing merging algorithms from past work are unfortunately not appropriate in a non-cooperative, heterogenous setting.

The scoring functions here fall into two families. The first assigns a score to each document, allowing us to rank them directly. The second scores collections instead, and then ranks documents based on the collection they came from.

3.1 Document scores

Rank. The simplest indication of a document's quality is the rank assigned by its original search engine: if the engine is effective, good documents will be at earlier ranks. To use this to generate a score in the range $[0, 1]$ we score documents with $1-(\text{rank}/1000)$: that is, the first (best) document scores 0.999, the next 0.998, then 0.997 etc. (Note that we never retrieve more than a few tens of documents in practice.)

Generic document scores. The "generic document scoring" functions of Rasolofo et al. [12] make use of document characteristics as presented by each search engine. They are attractive since they do not rely on any particular information beyond a list of documents, and in particular they do not rely on scores or full text.

Each document from collection c is represented by some field, d. We may use for example its title, or summary, or whatever else is available. The the score for this document for query q is then

$$\text{GDS}_{cd} = \frac{|q \cap d|}{\sqrt{|q|^2 + |d|^2}}$$

where $|q|$ is the number of words in the query, and $|d|$ the number of words in the field representing the document. GDS_{cd} ranges from 0 (no overlap) to $\sqrt{2}$ (complete overlap).

Following Rasolofo et al. [12] we use "TS" to mean the score based on a document's title field. If there is no title, or there are no query terms in the title, as a fallback a document receives 1/10th of its rank score (that is, 0.0999, 0.0998, etc). This ensures that any document with a query term in its title is ranked above any document with no query terms in its title, that is that title scores are always higher than rank scores; but that documents with no query terms in their title retain their original ordering. "SS" is the same, using snippets or summaries instead of titles. Details of the GDS methods are summarised in Section 3.4.

3.2 Collection scores

Document scores do not themselves account for the quality of a collection, but documents returned by a "good" server may well deserve a higher rank than those from a "poor" server. Collection scores allow this. In the experiments below we use two such scores, prioritising scale and accuracy respectively.

LMS. The LMS collection score of Rasolofo et al. [11] scores engines based on the reported size of their result sets: this prioritises engines with a higher number of responsive documents. The score for collection c is

$$\text{LMS}_c = \log\left(1 + \frac{l_c K}{\sum_{j=1\ldots n} l_j}\right) , \qquad (1)$$

where l_c is the reported length of the result set from c, K is a scaling constant set to 600, and there are n servers.

Similarity score. LMS will rank an engine highly if it reports many results, even if the results themselves are poor. To prioritise engines with more effective search, we also consider scoring engines by their mean DTSS score (a linear combination of scores from title and summary, falling back to ranks only if both are zero. See Section 3.4).

3.3 Round-robin algorithms

We are now in a situation to score and rank documents: that is, to carry out the merge.

Several past systems have used simple round-robin methods to merge result lists. Round-robin methods simply interleave the lists from each constituent engine, in order: so the first-ranked documents are returned from each engine, then the second-ranked, and so on.

The question is then how to order the engines themselves, or equivalently how to order the documents in each tier (note that round-robin requires that the ordering of engines is the same for each tier). The simplest approach is to order engines randomly, which we refer to as "SRR", and we use this as a baseline.

SRR can only be effective if all engines are equally effective, and relevant documents are evenly distributed [11, 12]. This is extremely unlikely in practice, so we considered two novel alternatives.

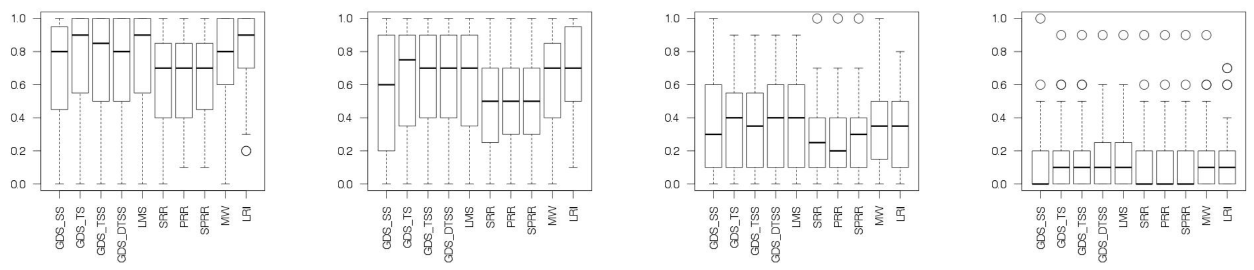
"PRR" orders engines according to the reported size of their result sets, so the first document in the merged list is the first document from the biggest set: this prioritises engines with a higher number of responsive documents. We do this by ranking collections according to LMS (Equation 1), then doing a round-robin merge.

The "SPRR" alternative orders engines according to the average DTSS score, that is the average similarity between the query and the documents in each engine's result set: this prioritises engines with more effective search.

In either case, the round-robin principle enforces diversity in the head of the final merged list.

3.4 Generic document ranking algorithms

Using the generic scoring function above gives four further



(a) (b) (c) (d)

Figure 3: Overall comparison of the methods on precision at 10 documents retrieved (prec@10) for different relevance judgment cutoffs: (a) "on topic", (b) "somewhat useful", (c) "useful", and (d) "comprehensively useful". Boxes mark 1st/3rd quartiles, bars mark median.

merging algorithms. These are based entirely on document characteristics:

- "GDS_TS" uses TS, that is the score from the title field with the rank score as a fallback.
- "GDS_SS" is similar, but uses the snippet for each result.
- "GDS_TSS" uses first the title, then the snippet, then the rank. This has been useful in informal experiments with other heterogeneous environments.
- "GDS_DTSS" uses a linear combination of scores from title (with weight 0.9) and summary (0.1), falling back to ranks only if both are zero.

3.5 Length-based algorithm

The algorithms above make use of only document-based or only collection-based scores. A novel method which combines the two is inspired by the LMS and GDS functions, as well as the CORI selection and merging method [2]. The "LMS" method combines LMS_c and DTSS_{cd} scores with CORI-like normalisation, with a score for document d from collection c of

$$\text{LMS}_{cd} = \frac{(\text{DTSS}_{cd} + 0.4 \times \text{LMS}_c \times \text{DTSS}_{cd})}{1.4}$$

where DTSS_{cd} and LMS_c are defined as above, the constant 0.4 follows CORI, and the constant 1.4 is for normalisation. This score affords some weight to the size of each result set (a proxy for collection quality) as well as the similarity between document and queries. As with the other GDS-based methods, it does not make any assumptions about engine features or result format besides the availability of titles or snippets and an estimated result size.

3.6 Multiple weights algorithm

The final GDS-based method in our experiments, also novel, is the most general algorithm we consider here. It uses a linear combination of the scores due to DTSS_{cd} (without rank component), LMS_c, rank, and the mean DTSS score of all documents returned by the server. This multiple weights method ("MW" below) generalises the GDS family.

In our experiments the weights on each component are learned with linear regression on the five-level relevance score, using four-fold cross-validation. On this training set the weights were 0.79 for mean DTSS, 0.48 for DTSS, 0.26 for rank and 0.04 for LMS. This is a strong bias towards similarity measures, and away from simple size of the result set; but the largest component (mean DTSS) is also a query-by-query effectiveness measure for each server.

3.7 Local re-indexing

The algorithms above rely only on information presented in a typical result list—titles, summaries, and sizes. We also experimented with downloading the full text of each document, via the URLs provided by each search engine, indexing them locally, and re-ranking the documents using this index. This overcomes the problems of incompatible scores, different collection statistics, and search engines of differing effectiveness; however it does represent significant overhead, especially as many more documents will be retrieved than will finally be in the head of the merged list.

We used version 4.4 of the open source Lucene retrieval system[3] with default settings for local reindexing and ranking.

One local re-indexing approach was used in the Inquirus metasearcher by Lawrence and Giles [6] and evaluated along with a number of local re-indexing alternatives by Craswell et al. [3]. Craswell et al. found that local reranking methods were able to outperform those based on scores and ranks, even when documents were only partially downloaded, though there was a big range of performance across the local re-indexing methods. It is an attractive option in our case, although ranking wildly different objects from the same index, such as our web pages and telephone directory, presents a serious hurdle.

3.8 Alternatives

Several further merging algorithms have been suggested in the literature, but are not evaluated here. These rely on features which are not generally available, and are certainly not available from many of the servers in our testbed: document scores, well-behaved scoring functions, and training data.

A number of proposals use document scores, as reported by constituent search engines. As these are unlikely to be on the same scale, they are commonly normalised linearly to the range $(0, 1)$, to z scores, or by fitting separate functions

[3]`lucene.apache.org/`

for relevant and non-relevant documents [7, 8, 10, 13]. Some of these methods are simple and perform relatively well, but do require more information from each engine. They are also naïve to differences in scoring approaches: for example, scores from each engine may be distributed differently; scores from one engine may be bunched together at one end of the range since all returned documents are equally good; or one ineffective engine may routinely assign inaccurate scores.

More sophisticated alternatives include the popular CORI algorithm [2], which combines document scores with a score for each collection, or the later SSL algorithm [15], which uses training data to learn how to transform each server's scores and therefore can account for differing server effectiveness. Again, neither algorithm is usable in our scenario since we do not have document scores and nor do we have training data for SSL.

4. RESULTS AND DISCUSSION

4.1 Methods compared

We compared the methods on one metric (prec@10) based on binary relevance levels and another (NDCG@10) taking advantage of the graded judgements. Figure 3 presents the prec@10 results for different score cutoffs, averaged across the 44 topics. Figure 4 presents the comparable NDCG scores.

The NDCG@10 plot shows a pattern in which the three round-robin methods (SRR, PRR, and SPRR) methods are inferior to the others and that GDS_SS is only slightly better. NDCG@10 conveys more useful information than prec@10 because it makes use of the grades of relevance, rewards methods which rank highly relevant documents above less relevant ones, and does not depress scores on topics for which no documents achieve the highest grades.

Naturally, the performance levels on prec@10 deteriorate as the threshold of relevance increases from left to right in the figure. Performance of the methods with a loose criterion of relevance shows that it is relatively easy to fill the top ten places in each merged list with results which match the query to at least a minimal degree, while the rightmost plot shows that some methods nearly always fail to include any highly relevant documents in the top ten.

However, the patterns apparent in the NDCG@10 plot are also visible in all of the prec@10 plots.

Considering NDCG@10, the "whiskers" in Figure 4 indicate a very high variance in scores achieved. The GDS_SS method is an extreme example with a proportion of topics achieving a zero score and another proportion achieving 1.0.

We summarise our observations on the NDCG@10 plots and confirm them with paired t-tests:

- Round-robin methods are generally inferior to the GDS methods. For example, GDS_TS > SPRR ($p < 0.01$). Even with an oracle prioritising the servers (not shown) the results are inferior to GDS_TS. Although diversity in result sets is often desirable, the diversity enforced by round-robin approaches is more hinderance than help in this setting.

- GDS_SS is worse than the other GDS methods. For example, GDS_TS > GDS_SS ($p < 0.01$).

- MW performs no better than the other GDS methods. The difference between GDS_TS and MW is not significant at the 0.05 level.

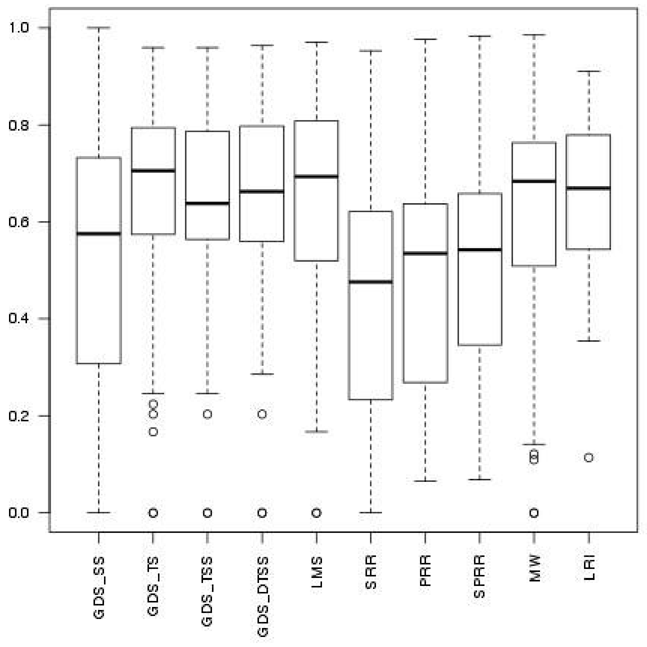


Figure 4: Overall comparison of the methods on NDCG@10.

- LRI is competitive with the GDS methods. The difference between GDS_TS and LRI is not significant at the 0.05 level.

- LRI is more consistent than other methods—the best queries perform a little worse than with other methods, but the worst perform better.

The poor performance of prioritised round-robin is unsurprising given that if all ten or more sources supply answers to a query, no server can contribute more than one result, even if one can supply many highly relevant documents and others return low value results.

The poor performance of GDS_SS is probably because some servers supply results with poor or absent snippets. This is true even of servers in our relatively small testbed and suggests that a general enterprise search engine will have to use some sort of fallback, as in GDS_TSS or the weighting methods.

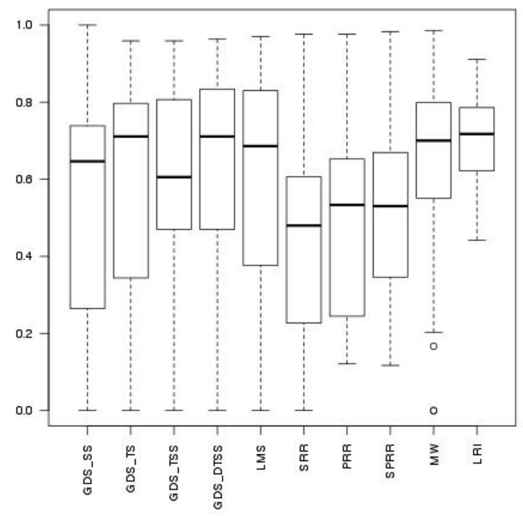
The MW method requires training to learn the appropriate weights. It is possible that with more training data it would be able to out-perform the other GDS methods. Another consequence of the need for training is that the evaluation method for MW (averaging across folds) is different from the method used to evaluate the other methods.

Craswell et al.[3] found considerable differences in retrieval effectiveness for different ranking schemes in local re-indexing. Further research is needed to determine whether other local ranking functions would be able to outperform GDS_TS.

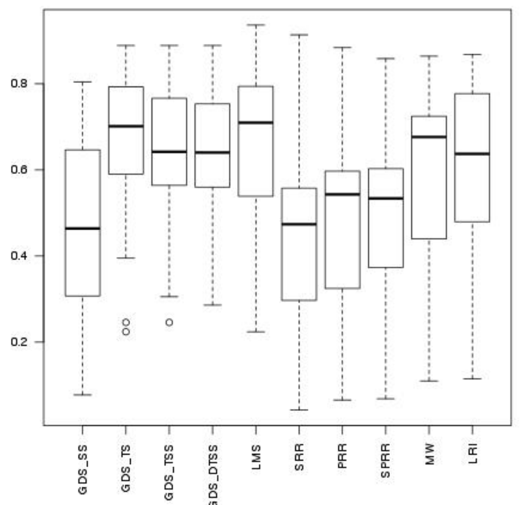
Finally, let us note that we were unable to compare performance against a centralised index since we have no access to the documents in some of the sources (e.g. ANU contacts and ANU library). The sources are genuinely non-cooperating.

4.2 Query characteristics

It is quite likely that merging methods may perform differently across different families of query. There are many



(a) NDCG@10 scores for 21 queries relating to names of people or organisations (NPO).



(b) NDCG@10 scores for the remaining 23 queries (OTHER).

Figure 5: Differences in performance for different families of query.

possible ways to divide up the query set. We present in Figure 5 a performance comparison for queries which relate to the names of people or organisations (NPO, 21 queries) with the rest (OTHER, 23).

As can be seen:

- The median performance of GDS_SS is substantially better on the NPO set, though variance is still very high. GDS_TSS also appears to perform better.
- Local re-indexing (LRI) is also more effective for the NPO queries.
- There is quite a large difference in performance between GDS_TSS and GDS_DTSS on the NPO queries but almost none on the others.
- LMS performs similarly to GDS_TSS for NPO queries but slightly better for OTHER. This may be because some sources have fewer good results for specific people or organisations, but those few results are highly useful: for example, the telephone directory may return only one result for a person (that person's details) but many results for other queries (anyone with a similar-sounding name or affiliation). In these cases, LMS will demote small sources which should in fact be promoted.

Otherwise, observations made on the full query set seem to also generally apply for each of the families shown here.

Future work may identify query characteristics tending to lead to better or worse performance of particular methods.

4.3 Correlation

Given the similar results in Figures 3 and 4, it is possible that the various algorithms are actually doing the same thing: that is, they might be using different methods but ranking documents the same way. We investigated this possibility by looking at the correlation between rankings.

Table 3 gives, for each pair of algorithms, the degree of correlation between their rankings. The figures are the mean correlation (Kendall's τ) across all documents for all 44 queries.

Algorithms which always rank all documents the same way will have mean correlation of 1, while pairs of algorithms which rank documents independently will have mean correlation of 0. As expected the GDS_ methods correlate well, particularly those which use the title field, while the various round-robin methods correlate surprisingly well with each other. The multiple weights and especially the local re-indexing algorithms produce rankings which are consistently different to all other algorithms (LRI has $\tau = 0.28$ against MW, and less than that in all other cases).

5. CONCLUSIONS AND FUTURE WORK

Enterprise search is an important but under-studied problem, and methods from federated search have promise since they can combine any number of non-cooperative, heterogeneous sources behind a single interface. In many instances, it is the only viable technique as the gather-and-index model is not feasible or not possible. We evaluated algorithms in a setting which is realistic not only for the institution we studied but for many others.

Result merging remains one of the major outstanding problems in federated search. In particular, previous evaluations have not considered the range of sources common to enterprise scenarios. Here we show that local re-indexing methods and methods based on titles and summaries of results achieve reasonable performance and outperform round-robin methods over a disparate collection of sources including websites, staff directory and external social media content—although these methods do need to adapt to differing document types and presentations.

Excluding GDS_SS and the round-robin methods it seems that there is no real argument from effectiveness to choose one

	GDS_SS	GDS_TS	GDS_TSS	GDS_DTSS	LMS	SRR	PRR	SPRR	MW	LRI
GDS_SS	—	0.66	0.44	0.43	0.66	0.11	0.13	0.16	0.29	0.18
GDS_TS		—	0.60	0.61	0.98	0.12	0.13	0.17	0.38	0.17
GDS_TSS			—	0.98	0.58	0.42	0.43	0.47	0.61	0.22
GDS_DTSS				—	0.59	0.42	0.43	0.47	0.61	0.21
LMS					—	0.12	0.14	0.18	0.39	0.18
SRR						—	0.91	0.90	0.59	0.16
PRR							—	0.93	0.64	0.19
SPRR								—	0.64	0.18
MW									—	0.28
LRI										—

Table 3: Correlations between rankings from different algorithms. Shown are the mean τ statistics across 44 queries.

method over another. The multiple weights method needs training data and locally re-indexing needs to download text for each result; locally re-indexing is more consistent than other methods, but in this environment a method such as GDS_TSS or LMS seems to be a reasonable choice on the whole. Locally re-indexing may be a good option if users are sensitive to poor results more than they are to "average" results. However, it entails slower response and more resources, and that may exclude it from some applications.

Further work could lead to an extension of the machine-learned multiple weights method, taking into account query and source features. It is also possible that it might be possible to use different merging algorithms for different families of queries, especially if queries can be classified automatically.

Acknowledgements. We thank the participants who provided relevance judgements. The protocol of Section 2.1 was approved by the Australian National University's Human Research Ethics Committee.

6. REFERENCES

[1] P. Bailey, N. Craswell, I. Soboroff, P. Thomas, A. de Vries, and E. Yilmaz. Relevance assessment: Are judges exchangeable and does it matter? In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 666–674, 2008.

[2] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1995.

[3] N. Craswell, D. Hawking, and P. Thistlewaite. Merging results from isolated search engines. In *Proceedings of the 10th Australasian Database Conference*, pages 189–200. Springer-Verlag, 1999. `http://david-hawking.net/pubs/craswell_adc99.pdf`.

[4] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, Oct. 2002.

[5] M. Lalmas. Aggregated search. In *Advanced Topics in Information Retrieval*, volume 33 of *The Information Retrieval Series*, pages 109–123. Springer, 2011.

[6] S. Lawrence and C. L. Giles. Inquirus, the NECI meta search engine. In *Proceedings of WWW7*, pages 95–105, 1998.

[7] J.-H. Lee. Analyses of multiple evidence combination. In *Proc.. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 267–276, 1997.

[8] R. Manmatha and H. Sever. A formal approach to score normalization for meta-search. In *Proc. Human Language Technology Research*, pages 98–103, 2002.

[9] MindMetre Research. Mind the search gap. Research report, 2011.

[10] M. Montage and J. Aslam. Relevance score normalization for metasearch. In *Proc. ACM Conf. on Information and Knowledge Management*, pages 427–433, 2001.

[11] Y. Rasolofo, F. Abbaci, and J. Savoy. Approaches to collection selection and results merging for distributed information retrieval. In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 191–198, 2001.

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

[13] M. E. Renda and U. Straccia. Web metasearch: rank vs. score based rank aggregation methods. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 841–846, 2003.

[14] M. Shokouhi and L. Si. Federated search. *Foundations and Trends in Information Retrieval*, 5(1):1–102, 2011.

[15] L. Si and J. Callan. A semisupervised learning method to merge search engine results. *ACM Trans. Inf. Syst.*, 21(4):457–491, Oct. 2003.

[16] P. Thomas and D. Hawking. Evaluating sampling methods for uncooperative collections. In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 503–510, 2007.

[17] C. Vespi. From overload to impact: an industry scorecard on big data business challenges. Oracle Corporation report, July 2012.