Microsoft Research at TREC 2009

Web and Relevance Feedback Tracks

Nick Craswell, Dennis Fetterly, Marc Najork, Stephen Robertson, Emine Yilmaz

Microsoft Research

Abstract

We took part in the Web and Relevance Feedback tracks, using the ClueWeb09 corpus. To process the corpus, we developed a parallel processing pipeline which avoids the generation of an inverted file. We describe the components of the parallel architecture and the pipeline and how we ran the TREC experiments, and we present effectiveness results.

1 Introduction

This report is from a transatlantic team from Microsoft Research, comprising members of the Silicon Valley and Cambridge UK labs. We processed the billion-page ClueWeb09 corpus using a framework for data-parallel computation running on a cluster of 240 machines, as described in section. 2. Features used in our runs included BM25 [7], BM25F [10], SALSA-SETR [5], matching anchor counts, tf-idf, various language model scores, PageRank [6], and inter-domain in-degree; further details are provided in section 3. Some tuning was done using home-grown evaluation data, described in sections 4. We used the same data-processing setup to take part in the Web track (section 5) and the Relevance Feedback track (section 6), although in the latter case we were unable to complete any significant work in time for the deadlines.

2 Computational Infrastructure

We processed the corpus using DryadLINQ [9], a platform for data-parallel computation that empowers programmers to utilize large clusters of computers without having to worry about the intricacies of distributed systems. DryadLINQ programs are written in C# utilizing LINQ (Language INtegrated Queries), a declarative query language in the SQL tradition. The bulk of a typical program consists of straightforward, sequential C# statements, with LINQ expressions embedded to process relational data. As the program is executed, the DryadLINQ system ships each LINQ expression to all machines in the cluster, where the query is processed in parallel, and the results are being sent back to the machine running the sequential C# program. The following C# method for computing document frequencies for all query terms provides a flavor of how we used DryadLINQ:

```
static void DFs(HashSet<string> queryTerms, string docsURL, string freqsURL) {
  PartitionedTable<TextDocument> docs = PartitionedTable.Get<TextDocument>(docsURL);
  var words = from doc in docs
              from word in doc.words.Distinct()
              where queryTerms.Contains(word)
              select word;
  var docFreqs = from term in words.Merge()
                 group term by term into g
                 select new Pair<string, int>(g.Key, g.Count());
  docFreqs.ToPartitionedTable(freqsURL);
}
```

To compute certain link-based features (most notably SALSA), we used the Scalable Hyperlink Store [4], a special-purpose storage system for web graphs. SHS keeps a compressed version of the web graph in main memory, distributed over a cluster of machines. It provides a simple API to its clients that hides the details of data distribution, fault tolerance, etc. We created an SHS database over 12 servers containing the inter-domain web graph induced by the ClueWeb09 corpus, totaling 4.8 billion nodes and 5.2 billion edges.

Finally, we computed PageRank scores using a custom distributed program. We could have used DryadLINQ as well, but we happened to have a pre-existing implementation that could handle the ClueWeb09 corpus quite easily.

3 Processing pipeline

Our processing pipeline consisted of the following stages:

1. Parsing: We used DryadLINQ to parse the HTML web pages, tokenizing them into individual words and hyperlinks. We associated each title and content word occurrence with the document containing it; in addition, we associated each anchor word occurrence with the document referenced by the anchor's hyperlink.

2. Document Frequencies: We computed DFs for the union of all terms of the 172 (training and test) queries we were considering.

3. Initial scoring: Still in DryadLINQ, we computed a single score for each query and each document in the collection, retaining the top 5000 results per query. For the Web track, the single score was BM25F and the process was run on the entire category A collection. For the Relevance Feedback track, the single score was plain BM25 and the process was run only on the restricted category B set; on the same pass, however, tf-idf and language model scores were also calculated.

4. Inter-domain in-degree (IDID): We used SHS to compute the inter-domain in-degree (IDID) for each of the top-5000 (according to BM25F) results of each query, that is, the number of pages in different domains linking to the result.

5. SALSA: Furthermore, we used SHS to run SALSA-SETR [5] on the top-5000 results of each query. SALSA-SETR is a variant of Lempel & Moran's SALSA algorithm [3], which in turn is a variant of Kleinberg's HITS algorithm [2], one of the best-known link-based ranking algorithms.

6. PageRank: We computed PageRank scores for each page in the ClueWeb09 corpus, using a special-purpose distributed program.

7. Matching anchor count (MAC): We resolved all the symbolic hostnames in ClueWeb09 URLs/links into IP addresses. Then using DryadLINQ we identified unique $\langle s, t, a \rangle$ triples, where s is the IP address of a document, t is the target URL of a link within that document and a is the anchor text of that link. Then we built the MAC ranking feature for each query-target pair $\langle q, t \rangle$, counting the number of source IPs s that link to target t with anchor $a = q$.

8. Extraction: For each query we identified a pool of documents and collated our ranking features. The resulting "extraction" file could be used for training or for generating submitted runs.

9. Training: Using an extraction of training queries and some surrogate qrels (see Section 4), we trained a single-layer LambdaRank model [1].

10. Runs: Using an extraction of the 50 test queries, we ran our trained model to produce our submitted runs.

4 Ranking Model

Various free parameters of our methods, including the BM25F parameters, those used in SALSA-SETR, and the various weighted linear combinations) needed to be tuned in some way. The basic approach to this was two-fold. We selected 118 queries from a search engine log as a training set. We ran these queries against the Bing search engine, and used Bing ranks to define a set of relevance labels, on the assumption that the Bing results were good. Rather than using the entire top-5000 BM25F as our training documents, we first take the intersection between top-5000 and the Bing top-10 ('Good') documents. We then add random ('Bad') documents to get a pool of 10 documents. This is further augmented by adding top-5000 documents that are in the top-4 on one of several features: BM25F, SALSA-SETR, PageRank and Anchor We then manually assessed all the pooled query-result pairs, using a 3-point scale. When Bing ranks are used to assign relevance labels to documents, documents retrieved at rank 1 were assumed to be *perfect*, ranks 2-3 *excellent*, 4-5 *good*, 6-10 *fair*, and documents retrieved lower down are assumed to be nonrelevant.

We used both the Bing-induced labels and the manual assessments to guide the tuning of the free parameters. We trained a single layer LambdaRank model for 300 epochs and we used a separate validation set to pick the best performing epoch. When the manual labels are used for training, the Bing-induced labels are used as the validation set and vice versa.

5 Web track submissions

We submitted three runs to the diversity task:

MSDiv1 A linear combination of BM25F, IDID, SALSA and PageRank. Manual labels are used as the training set and the Bing-induced labels are used as the validation set.

MSDiv2 A linear combination of BM25F, IDID, SALSA, PageRank and MAC. Manual labels are used as the training set and the Bing-induced labels are used as the validation set.

MSDiv3 A linear combination of BM25F, PageRank and MAC. We attempted to perform "host collapsing" – including only the two highest-scoring results from each host – so as to increase result diversity.

We also submitted MSDiv1 as MS1 and MSDiv2 as MS2 to the ad-hoc task. Table 1 shows the performance of our submitted runs. MSDiv2 and MSDiv3 (which both include the MAC feature) clearly outperform MSDiv1 (which does not include MAC), and their relative ordering varies depending on which performance measure is considered.

Run-ID	αnDCG@5	αnDCG@10	αnDCG@20	IA-P@5	IA-P@10	IA-P@20
MSDiv1/MS1	0.178	0.234	0.275	0.083	0.098	0.091
MSDiv2/MS2	0.267	0.306	0.352	0.130	0.109	0.102
MSDiv3	0.268	0.309	0.346	0.127	0.117	0.105

Table 1: Performance of of web track submissions

After the submission and while performing additional analysis for the final version of this paper, we discovered that our "host collapse" implementation used in preparing the MSDiv3 submission contained a bug – it excluded many more results than it should have. Table 2 shows how a bug-free implementation would have performed. The four rows in the table illustrate the effect of including the top one, two, three, or all results for a given host. We can draw three conclusions from this data: first, fixing the bug in the host collapse implementation improved performance; second, showing only two results per host produces only a mild improvement under some measures (*e.g.* αnDCG@5) and is detrimental under others; and third, the ranking function that produced MSDiv3 without host collapse (which used only three features) did better than the five-feature function that produced MSDiv2.

Run-ID	αnDCG@5	αnDCG@10	αnDCG@20	IA-P@5	IA-P@10	IA-P@20
MSDiv3(1)	0.279	0.317	0.352	0.129	0.117	0.104
MSDiv3(2)	0.285	0.321	0.359	0.136	0.121	0.109
MSDiv3(3)	0.283	0.321	0.359	0.137	0.120	0.110
MSDiv3(∞)	0.283	0.320	0.363	0.139	0.118	0.109

Table 2: Performance of MSDiv3, after fixing bug in "host-collapse" code

In order to understand why a ranking function incorporating fewer features did better, we measured the performance of each of the five features in isolation. Table 3 shows the results. According to the table, the matching anchor count feature in isolation performs about as well as any of the ranking functions in Table 2 for low document cut-off values, and not much worse for higher cut-offs. BM25F also performed quite well, but worse than MAC. Among the link-based features, inter-domain in-indegree outperformed SALSA and PageRank – which greatly surprised us, since earlier work suggested that SALSA should significantly outperform in-degree and PageRank [5].

Run-ID	αnDCG@5	αnDCG@10	αnDCG@20	IA-P@5	IA-P@10	IA-P@20
MAC	0.283	0.312	0.355	0.133	0.109	0.092
BM25F	0.175	0.218	0.251	0.083	0.082	0.072
IDID	0.131	0.162	0.181	0.053	0.055	0.045
SALSA	0.127	0.155	0.183	0.049	0.048	0.041
PageRank	0.095	0.118	0.136	0.042	0.042	0.034

Table 3: Performance of individual features

Inspection of scores and judgments of a few queries revealed why the link-based features performed counter to previous observations: Many of the results with high SALSA scores were not judged (the same is true for PageRank, albeit to a lesser degree). In order to account for this fact, we devised a simple measure that is inspired by the standard precision measure, but ignores unjudged results: we consider the twenty highest-scoring results and simply divide the number of relevant results by the number of judged results. Table 4 shows the number of judged and relevant results together with their ratio, for each individual feature. Under this measure, SALSA performs best, closely followed by matching anchor count. We hypothesize that SALSA performs differently under the relevant/judged measure versus standard measures because it is a fairly orthogonal feature: It surfaces many results that are not surfaced by other features. The judgment process was driven by the pooling of submitted runs, and we did not submit any runs using SALSA as the sole feature.

Run-ID	judged	relevant	relevant/judged
SALSA	6.96	2.50	0.359
MAC	15.78	5.58	0.354
BM25F	15.30	4.20	0.275
IDID	12.36	3.04	0.246
PageRank	9.72	2.32	0.239

Table 4: Performance of individual features, ignoring results without judgments

Finally, Table 5 shows the number of judged and relevant results together with their ratio, for the three submitted runs (bottom rows) as well as MSDiv3 with the host-collapse bug fixed (top rows). Under this measure, MSDiv3 slightly outperforms MSDiv2 and greatly outperforms MSDiv1 (consistent with Table 1), fixing the host-collapse bug improves performance (consistent with Table 2), and SALSA and MAC in isolation perform better than the best parametrization of MSDiv3. There are two plausible explanations for this: one, the relevant/judged measure may become more fragile as the fraction of unjudged results goes up; and two, we may have combined features in a less than optimal way in our submissions.

Run-ID	judged	relevant	relevant/judged
MSDiv3(3)	19.80	6.62	0.334
MSDiv3(2)	19.64	6.56	0.334
MSDiv3(∞)	19.86	6.62	0.333
MSDiv3(1)	19.34	6.28	0.325
MSDiv3	19.96	6.32	0.317
MSDiv2/MS2	19.98	6.24	0.312
MSDiv1/MS1	19.94	5.46	0.274

Table 5: Performance of our runs, ignoring results without judgments

In order to check whether we combined features in a suboptimal way, we performed two sets of experiments pertaining to combination of evidence. First, we experimented with weighted linear combinations of BM25F and MAC, i.e. $score = \text{BM25F} + w \cdot \text{MAC}$. We found that performance plateaued for $w \geq 2$, as shown in the top half of Table 6. We also tried applying a log-based transform to MAC, i.e. $score = \text{BM25F} + w \cdot \log \text{MAC}$, and found that performance plateaued for $w \geq 1$, as shown in the bottom half of Table 6. In other words, MAC is the dominant ranking signal, and BM25F serves as a “tie-breaker”.

w	αnDCG@5	αnDCG@10	αnDCG@20	IA-P@5	IA-P@10	IA-P@20
0.5	0.273	0.307	0.356	0.133	0.114	0.108
1	0.274	0.319	0.361	0.133	0.118	0.109
2	0.283	0.320	0.363	0.139	0.118	0.109
10	0.283	0.320	0.363	0.139	0.118	0.109
100	0.283	0.320	0.363	0.139	0.118	0.109
1000	0.283	0.320	0.363	0.139	0.118	0.109
10000	0.283	0.320	0.363	0.139	0.118	0.109
0.1	0.265	0.309	0.355	0.126	0.115	0.105
1	0.283	0.320	0.363	0.139	0.118	0.109
10	0.283	0.320	0.363	0.139	0.118	0.109
100	0.283	0.320	0.363	0.139	0.118	0.109
1000	0.283	0.320	0.363	0.139	0.118	0.109

Table 6: Combining BM25F and MAC

Next, we tried three-way combinations using the scoring function $score = \text{BM25F} + \text{MAC} + w \cdot \log L$, with L being one of the link-based features PageRank, IDID and SALSA. We found that the performance of each three-way combination showed a sharp peak for a particular value of w, unlike what we observed for the two-way combination of BM25F and MAC. Table 7 shows the results for the best-performing values of w. Adding SALSA as the third feature increased performance more than either PageRank or IDID did. Comparing PageRank and IDID, PageRank contributed more under αnDCG@5 and IA-P@5, while IDID was preferable at higher document cut-off values. We also note that the fairly simple combination of MAC, BM25F and SALSA performs substantially better than any of our submitted runs (*cf.* Table 1), even after discounting the “host-collapse bug” (*cf.* Table 2). In other words, when preparing our submitted runs, we did indeed combine features in a suboptimal fashion.

Finally, Table 8 shows the performance of three-way combinations of evidence when ignoring unjudged results. The combination of SALSA, BM25F and MAC performs best as expected. Two things to note are that the best

$score = \text{BM25F} + \text{MAC} + \ldots$	αnDCG@5	αnDCG@10	αnDCG@20	IA-P@5	IA-P@10	IA-P@20
$\ldots 500 \cdot \log \text{SALSA}$	0.291	0.331	0.375	0.138	0.122	0.112
$\ldots 0.1 \cdot \log \text{IDID}$	0.279	0.328	0.372	0.134	0.120	0.109
$\ldots 10^6 \cdot \log \text{PageRank}$	0.285	0.322	0.364	0.139	0.117	0.109

Table 7: Combining BM25F and MAC with one of SALSA, IDID or PageRank

performance is higher than that of MSDiv3(3) (*cf.* Table 5) as well as that of SALSA in isolation (*cf.* Table 4), further supporting the notion that MSDiv3 combines features in a suboptimal way. Second, performance is maximal for $w \approx 5000$ (as opposed to $w \approx 500$ in Table 7. This is because giving higher weight to SALSA draws in more unjudged results, which is being penalized by the metrics used in Table 7 but not those used in Table 8.

Weight	judged	relevant	relevant/judged
$\text{BM25F} + \text{MAC} + w \log \text{SALSA}$			
100	19.76	6.64	0.336
500	19.04	6.72	0.353
1000	18.44	6.56	0.356
5000	13.86	5.16	**0.372**
10000	11.16	3.98	0.357
$\text{BM25F} + \text{MAC} + w \log \text{IDID}$			
0	19.86	6.62	0.333
0.001	19.88	6.64	0.334
0.01	19.88	6.66	0.335
0.05	19.54	6.66	0.341
0.1	19.04	6.56	**0.345**
0.2	18.82	6.38	0.339
0.5	17.22	5.54	0.322
1	15.50	4.54	0.293
$\text{BM25F} + \text{MAC} + w \log \text{PageRank}$			
10^9	18.46	5.90	0.320
10^8	19.82	6.66	**0.336**
10^7	19.86	6.64	0.334
10^6	19.86	6.64	0.334
10^5	19.84	6.62	0.334
10^4	19.84	6.62	0.334
10^3	19.86	6.62	0.333
10^2	19.86	6.62	0.333
10^1	19.86	6.62	0.333
10^0	19.86	6.62	0.333
0	19.86	6.62	0.333

Table 8: Combining BM25F and MAC with SALSA, IDID or PageRank, and ignoring results without judgments

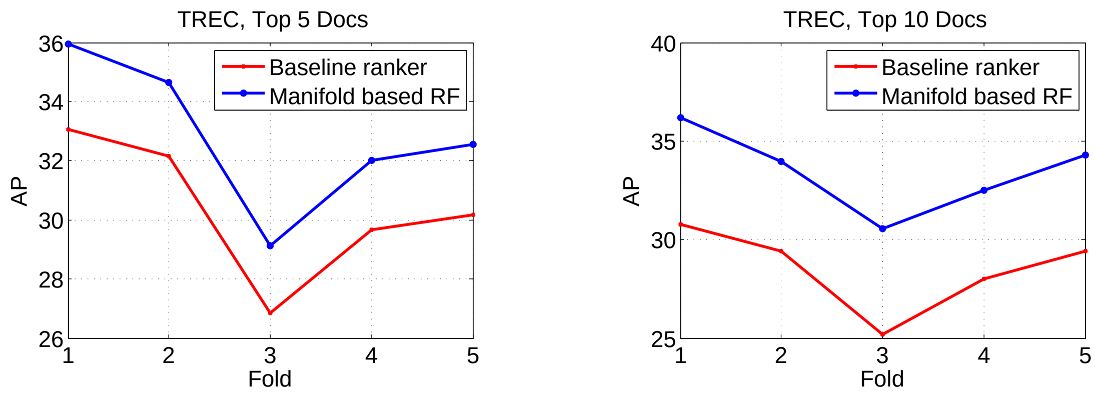

Figure 1: Improvements over the 5 folds when the feedback documents are (left) the top 5 documents, and (right) the top 10 documents in the initial ranking.

6 Relevance feedback track submissions

In the first phase of the Relevance Feedback Track, we used BM25, tf-idf and language modeling parameters computed on the Category B collection as features. To pick the feedback documents, a baseline ranker was obtained by forming a weighted linear combination of these parameters. The feedback documents were picked using two different methodologies:

MSRC1 Top 5 documents retrieved by our baseline ranker are submitted for feedback.

MSRC2 Top 20 documents were extracted using our baseline ranker. The top document is always picked for feedback. The second document picked for feedback is the one that is the least similar of the remaining documents to the first document, and so on. The similarities between different documents are computed using cosine similarity as the metric.

For the second phase of the track, our intention was to use a relevance feedback model based on manifold regularization and document contents. Using some previous training set (formed from the Bing-induced labels or manual assessments), we use a regression based model to assign some initial weight to each of our features. Given the feedback documents for each query, we would then like to update these weights separately for each query as different features may be important for different queries. Hence, we learn a different ranker separately for each query. This could be done by updating the initial weights using regression on the feedback documents. However, since the number of judged documents is very limited, using simple regression on the feedback documents will easily run into the problem of overfitting. To overcome this problem, we add a *regularizer* to the regression model to include the information provided by thousands of *unjudged* documents. The regression model guarantees that the feature weights are updated so that the scores of the feedback documents match their labels, and the regularizer guarantees that the documents that are similar to each other in terms of contents are ranked similarly.

Figure 1 shows the result of our initial experiments using data from TREC 6, 7 and 8 Ad-hoc retrieval track. Collectively, this dataset contains 150 queries and is split into five parts in order to conduct five-fold cross validation. For each fold, we use one part for training the baseline ranker, one fold for validation and three folds for testing our relevance feedback algorithm. The left and right plots in the figure shows how relevance feedback using manifold regularization compares with the quality of the baseline ranker using top 5 and 10 documents from the baseline ranker as the feedback documents, respectively. The x axis in the figure show the fold number and the y axis shows the average precision value. It can be seen that using the manifold regularization based relevance feedback algorithm, one can obtain significant improvements over the baseline ranker. The differences are statistically significant according to a Wilcoxon sign-rank test at $p = 0.5$ significance level. The details of both the experiments and the manifold regularization based relevance feedback algorithm will be published elsewhere.

Unfortunately, we were unable to finish the formulation of this algorithm in time for the Relevance Feedback Track. Instead we submitted one run using a single feedback set (from one of the other participants) and a method inspired by the usual Rocchio approach [8]. We extract the top 50 terms from the positive (relevant) documents according to their term frequencies (largest sum of term frequencies over the relevant set). We then alter the query using these terms, retrieve a new list of documents according to BM25 and language modeling features for the altered query. If s_1 is the score of a document in the original ranked list and s_2 is the score of a document in the ranked list for the altered query, we now merge the two results lists using a score of $s_1 + 0.02 * s_2$.

7 Conclusions

Although most TREC participants use systems based on the usual inverted-file structure, there have been occasional forays into different experimental infrastructures (at the first TREC, for example, there was one system based on a hardware text scanner, another on passing documents sequentially for matching against an inverted file of queries). For this submission we have begun experimenting with a new setup which avoids inverted-file building. As a preliminary impression, it appears to provide a flexible environment in which to do a range of experiments. But we need to do many more to discover its benefits and limitations. As regards the value of the specific approaches to the Web and Relevance Feedback tasks, we have yet to perform the necessary analyses and further experiments.

References

[1] C.J.C. Burges, R. Ragno and Q.V. Le. Learning to rank with non-smooth cost functions. In *Proc. of the 20th Annual Conference on Neural Information Processing Systems*, pages 193–200, 2006.

[2] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998.

[3] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks and ISDN Systems*, 33(1–6):387–401, 2000.

[4] M. Najork. The scalable hyperlink store. In *Proc of the 20th ACM Conference on Hypertext and Hypermedia*, pages 89–98, 2009.

[5] M. Najork, S. Gollapudi, and R. Panigrahy. Less is More: Sampling the neighborhood graph makes SALSA better and faster. In *Proc of the 2nd ACM International Conference on Web Search and Data Mining*, pages 242–251, 2009.

[6] L. Page, S. Brin, and T. Winograd. The PageRank citation ranking: bringing order to the web. Technical Report, Stanford InfoLab, 1999.

[7] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. of the 3rd Text REtrieval Conference*, 1994.

[8] J. J. Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall, Englewood, Cliffs, New Jersey, 1971.

[9] Y. Yu, M. Isard, D. Fetterly, M. Budiu, Ú. Erlingsson, P. K. Gunda, J. Currey. DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language. In *Proc. of the 8th USENIX Symposium on Operating Systems Design and Implementation*, pages 1–14, 2008.

[10] H. Zaragoza, N. Craswell, M. Taylor, S. Saria, and S. Robertson. Microsoft Cambridge at TREC–13: Web and HARD tracks. In *Proc. of the 13th Text Retrieval Conference*, 2004.