Application of Data Fusion in the Web Track

Chunlan Huang, Shengli Wu, Jinbo Feng, Yongquan Tao, and Yuping Xing
School of Computer Science and Telecommunication Engineering
Jiangsu University, Zhenjiang, China, 212013

Abstract

In this paper, we report on the experiments we conducted whilst participating in the TREC 2013 Web track. We use data fusion to test how to improve the results from different information retrieval systems. Linear combination is used for fusion and multiple linear regression is used to obtain suitable weights for all the component systems involved. In our experiments, the ClueWeb09 dataset is used as training data to obtain weights for the three component systems Indri, MG4J, and Terrier. After running the official evaluation program, we find that all four runs submitted are better than all component results with one exception.

1. Introduction

This is the first time that we have participated in TREC and we submitted four runs to the Web track. Two of them are to ad hoc and two others to risk-sensitive tasks. We choose the ClueWeb12 Catalog-B collection for both tasks. The main technology used is data fusion, which means that we run three component information retrieval systems Indri[1], MG4J[2], and Terrier[3] separately to search the same document collection, and then merge the results from these different systems to obtain the final fused result.

A collection crawled from the Web is likely to contain a considerable amount of spam pages. Many approaches have been proposed to identify spam pages based on page content, hyperlink structure, URL form, the similarity between pages of a single host, and combinations of the above-mentioned features [1]. We use Waterloo's list that we downloaded from [2]. After checking the spam pages manually, we set the "spamminess" threshold parameter to 15%, which we believe is a reasonable value to use.

[1] http://sourceforge.net/p/lemur/wiki/Indri/
[2] http://mg4j.dsi.unimi.it/
[3] http://terrier.org/

Figure 1 and Figure 2 show the main processes for ad-hoc and risk-sensitive tasks respectively. Three state-of-art systems (Indri, MG4J and Terrier) were chosen to obtain the original results, which is followed by the removal of the spam pages during the post-retrieval phase. Finally, we use linear combination to fuse the component results, with the weights obtained by using the ClueWeb09 Category B collection and 50 queries used in the TREC 2011 Web track.

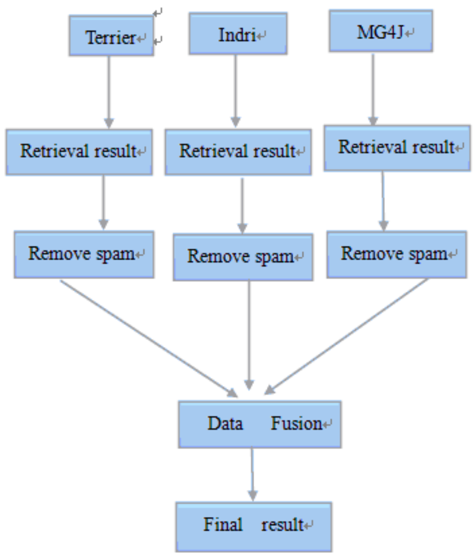

Figure 1. Flow chart for the ad-hoc task

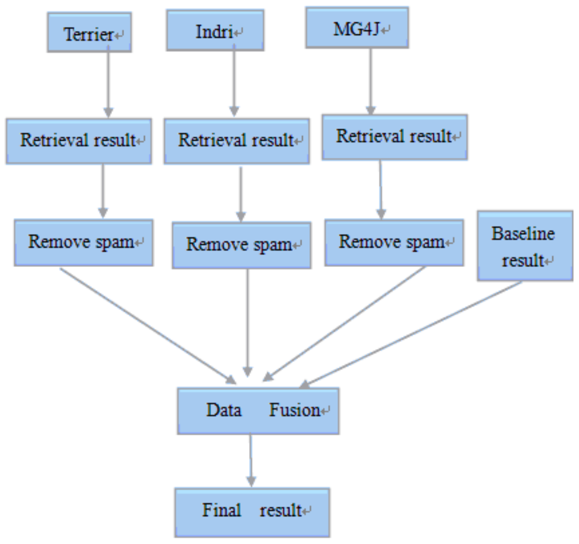

Figure 2. Flow chart for the risk-sensitive task

The remainder of this report is structured as follows. In Section 2, we describe those component systems involved. Section 3 details our data fusion strategy. In Section 4, we describe the settings of the data fusion approach and all related issues. In Section 5, we present the results for the 4 runs submitted to the ad-hoc task and the risk-sensitive task. Conclusive remarks are given in Section 6.

2. Description of Component Retrieval Systems

In this section, we briefly discuss Indri, MG4J, and Terrier. All of them are well-known retrieval systems in the information retrieval research community.

2.1 Indri

Indri is a search engine that provides state-of-the-art text search capabilities and a rich structured query language for text collections of up to 50 million documents (single machine) or 500 million documents (distributed search). It is available for Linux, Solaris, Windows and Mac OSX. Indri is a part of the Lemur project and developed by researchers from UMass and Carnegie Mellon University.

From an academic perspective, Indri combines inference networks with language modeling. The query language, which is reminiscent of the Inquery query language, allows researchers to experiment with proximity, document structure, text passages, and other document features without writing code. Like other academic engines, Indri can parse TREC newswire and web collections, and it is able to return results in the TREC standard format.

From an industrial perspective, Indri is efficient and easy to integrate with other software components. Indri is freely available from UMass with a flexible BSD-inspired license. Indri includes an API that is accessible from C++, Java, C# and PHP. Indri can also be distributed across a cluster of nodes for high speed query performance. In version 2.0, Indri adds true multithreaded operation, so documents can be added, queried and deleted concurrently.

2.2 MG4J

MG4J (Managing Gigabytes for Java) is a free full-text search engine for large document collections written in Java. It is a customizable search engine providing state-of-the-art features (such as BM25/BM25F scoring) and new research algorithms. As described by its developers, it main characteristics are:

- Support for document collections and factories makes it possible to analyze, index and query consistently large document collections, providing easy-to-understand snippets that highlight relevant passages in the retrieved documents.
- It supports multi-index interval semantics. When a query is submitted, MG4J returns, for each index, a list of intervals satisfying the query. This provides the basis for several high-precision scorers and for very efficient implementation of sophisticated operators. The intervals are built in linear time using new research algorithms.

- MG4J provides implementations of phrase queries, proximity restrictions, ordered conjunction, and combined multiple-index queries. Each operator is represented internally by an abstract object, so a user can easily plug in one's favorite syntax.
- It supports virtual fields—fields containing text for a different, virtual document; the typical example is anchor text, which must be attributed to the target document.
- MG4J is flexible so that smaller indices are possible by dropping term positions, or even term counts. Several different types of codes can be chosen to balance efficiency and index size. Documents coming from a collection can be renumbered (e.g., to match a static rank or experiment with indexing techniques).

2.3 Terrier

Terrier is an open source search engine developed at Glasgow University. It is written in Java, readily deployable on large-scale collections of documents. Terrier provides another good platform for research and experimentation in text retrieval, especially for research carried out on standard TREC test collections.

Terrier can index large corpora of documents, and provides multiple indexing strategies, such as multi-pass, single-pass and large-scale MapReduce indexing. A user can specify fields that need to be indexed. Terrier also offers many models for retrieval, such as BM25, TF_IDF and the BB2 weighting model. When running Terrier in the experiment, we chose the multi-pass mode. During indexing, all the words in the collection were stemmed using the well-known Krovetz stemmer and stop-words were removed using the stop-word list embedded within Terrier.

3. Data Fusion Strategy

Previous studies demonstrate that data fusion, which uses a group of information retrieval systems to search the same document collection, and then merges the results from these different systems, is an attractive option to improve retrieval effectiveness.

Among different data fusion methods, linear combination is an effective method because of its flexibility in assigning different weights to different component systems. For a query *q*, each of the systems ir_i provides a result r_i and each r_i comprises a ranked list of documents with associated scores. The linear combination method uses the following formula to calculate score for every document *d*:

$$\mathrm{M(d,q)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \beta_{\mathrm{i}} * \mathrm{S_i(d,q)}$$

Here $S_i(d,q)$ is the normalized score of document *d* in result r_i, β_i is the weight assigned to system ir_i, and *M* (*d*, *q*) is the calculated score of *d* for *q* . All the documents can be ranked according to their calculated scores.

How to assign weights to the component systems is a key issue in linear combination. Multiple linear regression (LCR) is an effective method [3] that minimizes the difference between the estimated scores of all documents by linear combination and the judged scores of those documents in the sense of least squares.

Let us assume that there are *n* information retrieval systems $ir_1, ir_2, \ldots, ir_n$, *m* queries and *l* documents in a document collection *D*. Every document in at least one of the results can be represented as a score vector $< s_{1k}^i, s_{2k}^i, \ldots, s_{nk}^i, y_k^i >$ for *i*= (1, 2… m), *k*= (1, 2… l). Here s_{jk}^i stands for the score assigned by retrieval system ir_j to document d_k for query q^i and y_k^i is the judged relevance score of d_k for query q^i. We want to estimate

$$\mathrm{Y} = \{ y_k^i ; i = (1,2,\ldots,m), k = (1,2,\ldots,l) \}$$

using the linear combination of scores from all component systems. The least squares estimates are used for the calculation of $(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_n)$ for which the quantity

$$\mathrm{q} = \sum_{\mathrm{i=1}}^{\mathrm{m}} \sum_{\mathrm{k=1}}^{\mathrm{l}} [\mathrm{y_k^i} - (\hat{\beta}_0 + \hat{\beta}_1 s_{1k}^i + \hat{\beta}_2 s_{2k}^i + \cdots + \hat{\beta}_n s_{nk}^i)]^2$$

is a minimum.

Coefficients $\beta_1, \beta_2, \ldots, \beta_n$, are numerical parameters that can be determined from some training data and can be used as weights by $ir_{1,} ir_2, \ldots, ir_n$ for fusion.

A related issue is how to obtain reliable scores for those retrieved documents. Sometimes scores are provided by component retrieval systems, but usually those raw scores from different retrieval systems are not comparable and some kind of score normalization is needed before fusing those documents. An alternative is to convert ranking information into scores if raw scores are not available or not reliable. In our experiment, we use a reciprocal function of rank [4] for this.

4. Experimental setup

In this section we provide information for the setup of the experiment including how to obtain weights, running component retrieval systems, score normalization, etc.

3.1 Obtaining weights from training data

The ClueWeb09 Category B dataset is indexed by three information retrieval systems Indri, MG4j and Terrier, separately. 50 queries in the TREC 2011 web track are used to obtain results and each retrieves 10000 documents for every query. In the training process, documents are divided into two types, relevant and irrelevant documents, for linear regression analysis. The weights we obtain are 5.2534, 6.0191, and 5.1540 for Indri, MG4J, and Terrier, respectively.

3.2 Running component retrieval systems

When running the component retrieval systems on the ClueWeb12 Category B collection, we use all the words given by TREC for each topic. For example, the words for topic 203 are “reviews of les miserables”. Next, we remove probable spam pages (top 15% of the pages in the list downloaded from [2]). We keep two slightly different results for each component retrieval systems: the original result and the result with the spam pages removed.

3.3 Normalizing scores

In order to generate reliable and comparable scores for all the documents involved, we use a reciprocal function of rank [4]. For a group of ranked documents <d_1, d_2…, d_n>, whose ranks are 1, 2, …, *n*, we normalize them by $\mathrm{s_i} = \frac{1}{\mathrm{i}+60}$ for $1 \le \mathrm{i} \le \mathrm{n}$.

3.4 Data fusion

For the ad hoc task, we fuse results from Indri, MG4J and Terrier with the aforementioned weights in Section 3.1. Two runs are generated: UJS13LCRAd1 from the original results of those component systems, and UJS13LCRAd2 from modified results with spam pages removed. For the risk-sensitive task, we fuse results from Indri, MG4J, Terrier and the baseline (as provided by TREC) as well. The weights for Indri, MG4J, and Terrier are the same as in the ad hoc task; we give baseline a weight equal to the average of all 3 component systems. As in the ad hoc task, we also submitted two runs UJS13LCRAd1 and UJS13LCRAd2. The former is the fused result from original results, while the latter is the one from modified results after spam page removal.

5. Results analysis and conclusions

Results from component systems and fusion are presented in Tables 1 and 2 for ad hoc and risk-sensitive tasks, respectively.

Table 1. Performance of component and fused results in ad-hoc task

Run	ERR@20	nDCG@20	P@20	MAP
UJS13LCRAd1	0.097	0.144	0.248	0.045
UJS13LCRAd2	0.107	0.148	0.251	0.052
Terrier	0.082	0.112	0.186	0.026
Indri	0.1	0.122	0.203	0.044
MG4J	0.082	0.106	0.175	0.033

Table 2. Performance of component and fused results in risk-sensitive task

Run	ERR-IA@20	@-nDCG@20	NRBP
UJS13Risk1	0.451	0.511	0.415
UJS13Risk2	0.468	0.522	0.434
Terrier	0.367	0.431	0.328
Indri	0.387	0.450	0.345
MG4J	0.431	0.493	0.393
Baseline	0.326	0.385	0.291

From Tables 1 and 2, we can see that the fused results are better than all component results with only one exception. When measured by ERR@20, UJS13LCRAd1 does not perform quite as well as Indri. In the ad-hoc task, UJS13LCRAd2 is the best performer outperforming the best component result (Indri) by 7.00% (measured by ERR@20), 21.31% (measured by nDCG@20), 23.65% (measured by P@20), and 18.18% (measured by MAP) - the improvement is notable. In the risk-sensitive task, UJS13Risk2 is the best, outperforming the best component result by 8.58% (measured by ERR-IA@20), 5.88% (measured by @-nDCG@20), and 10.43% (measured by NRBP). If we consider the improvement rate of the fused result to the best component result, then data fusion did better in the ad-hoc task than in the risk-sensitive task. This is not very surprising since the method of data fusion we have used was initially designed for ad hoc tasks, in particular with the measures for binary relevance judgment such as MAP and P@20.

6. Conclusive remarks

This is the first time that we have participated in TREC. In the future, we will investigate data fusion algorithms that are suitable for different kinds of situations such as the risk-sensitive task and measures that are based on graded relevance judgment. We will participate in more such evaluation activities to evaluate our algorithms.

7. References

[1] Ntoulas, A., Najork, M., Manasse, M. and Fetterly, D.: Detecting spam web pages through content analysis. In Proceedings of the 15th international conference on World Wide Web, pages 83–92, 2006.

[2] http://www.mansci.uwaterloo.ca/~msmucker/cw12spam/

[3] Wu, S., Bi, Y., and Zeng, X.: The linear combination data fusion method in information retrieval. In Proceedings of 22nd International Conference on Database and Expert Systems Applications, Part II， pages 219-233, 2011.

[4] Cormack, G. V., Clarke, C. L., & Buettcher, S.: Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval, pages 758-759, 2009.