MG4J at TREC 2006

Paolo Boldi* Sebastiano Vigna
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano, Italy

Abstract

MG4J participated in the *ad hoc* task of the Terabyte Track (find all the relevant documents with high precision from 25.2 million pages from the `.gov` domain) at TREC 2006. It was the second time the MG4J group participated to TREC. For this year, we integrated standard techniques (such as stemming and BM25 scoring) into MG4J, and submitted also automatic runs based on trivial query expansion techniques.

1 Introduction

MG4J is a Java indexing system that we have been developing in the last four years with the initial purpose of supporting searches over the crawls performed by UbiCrawler [2].

Initially a loosely coupled set of classes supporting standard text-indexing techniques inspired by MG [10], it has evolved into a quite complex system implementing a large class of scalable algorithms that are of interest to the text-retrieval community. Because of its flexibility, it has been used, for instance, in IR research to study problems of document reordering [1] and for building databases of protein names from textual documents [9].

After the first implementation phase, MG4J has been used as a playground for research ideas in text retrieval, with a special focus on efficiency even with large corpora. In particular, we developed a new skipping system [4] based on the embedding of compressed perfect skip lists, and we extended the classical Clarke–Cormack–Burkowski [7] lattice for structured queries to support multiple indices and negation. We are also developing new scorers based on that extension, that incorporate novel ideas about multi-index semantics.

In MG4J the emphasis is always on linear algorithms. We are interested in indexing systems that can scale easily to the web size and that can be used under heavy concurrent access with a very low response time. These requirements limit the range of techniques that can be used, but it is at the same time a great stimulus for finding more efficient algorithms and implementations.

MG4J is free software distributed under the GNU Lesser General Public License, and can be downloaded from `http://mg4j.dsi.unimi.it/`.

*This work is partially supported by MIUR PRIN Project "Automi e linguaggi formali: aspetti matematici e applicativi" and by the EC Project DELIS.

2 Indexing

This year we developed an *ad hoc* document factory (see the MG4J documentation) for TREC. This made it possible to index easily all or part of the GOV2 collection. The factory is in turn based on a document collection that exposes a set of (possibly gzip'd) files as a set of segments (the contents of the header and of the body part of a TREC GOV2 document). We did not develop support for WebGraph [3], as for this year we decided to drop PageRank from the scorers. The reasons are several; first of all, the experience gathered the last year shows (as expected) that GOV2 is too small (and heavily biased) to show a significant advantage of PageRank w.r.t. more trivial link-based schemes (e.g., indegree count or weighted indegree count); second, GOV2 contains a large number of duplicate (even triplicate) URIs, and even a larger number of URIs that are duplicate after trivial normalisation procedures. Since our index construction process is based on the idea of unique, normalised URIs, ranking GOV2 cleanly would have required a massive effort (the last year, we simply dropped all duplicate URIs, which however affected recall). For anchor text, we made duplicate URIs into unique URIs by adding greedily random noise to each URI that appeared to be a duplicate.

An advantage of the GOV2 collection is that, as biased as it is, it is mainly monolingual, which made it possible to index it safely after downcasing and Porter stemming. Some experiments showed that, indeed, this choice improved recall.

3 Querying

MG4J makes it possible to combine several indices over the same document collection. In our case, the indices were made of the title of a page, its text and the text of the anchors pointing to the page. Queries can use classical Boolean-like operators (and, or, not), operators that are specific to minimal-interval semantics (consecutivity, low-pass, ordered-and) and operators for multi-index querying (index specifiers, multiplexers). Additionally, MG4J provides an "and then" operator that computes the results for a query and then appends new results from additional queries.

The "and then" operator has been used in the *ad hoc* manual runs to mimic the behaviour of a search engine user who starts with a very stringent query and then relaxes it when too few documents are returned. It has also been used in the automatic runs, as queries were generated by taking the words in the title, dropping stopwords, and generating a chain of queries containing the conjunction of all title terms, "and then" the disjunction of all possible conjunctions of all title terms but one, and so on, up to the disjunction of all terms; more precisely, if you let W denote the set of non-stopword terms appearing in the title, the automatic query submitted to the system was

$$\bigoplus_{k=0}^{|W|-1} \bigvee_{S \in \binom{W}{|W|-k}} \bigwedge_{t \in S} t,$$

where $\oplus$ denotes the "and then" operator, and $\binom{W}{k}$ denotes the set of all subsets of W of cardinality k. Even though the size of automatic queries grows exponentially in $|W|$, the system keeps

responding within reasonable time since every operator (except, of course, $\oplus$) is implemented in an inherently lazy manner [5].

The scoring of our runs is obtained by combining linearly a BM25 standard scorer and the proximity-based scorer used at TREC 2005 (implemented by the class `VignaScorer` of MG4J). Depending on the last letters of the run, the weight of each scorer was 1 or 2 (for instance, in BVV runs the proximity-based scorer has twice the importance of the BM25 scorer). The weighting is purely tentative—clearly, a more sophisticated weight-decision procedure based on previous years' results would provide a more solid choice. For details on the algorithms used by proximity-based scorer, see [5].

We implemented BM25 scoring using the formula reported by Robertson, Zaragoza and Taylor in [8], using the weights proposed therein ($k_1 = .75$, $b = .95$).

4 Results

For the first year we submitted both automatic and manual runs. As explained in the previous section, automatic runs were generated using a very simple term-subset expansion.

After submitting our results, we detected some bugs and configuration mistakes: here we report results obtained after our fixes (see Table 1).

- First of all, we discovered too late that Büttcher and Clarke [6] reported different values for BM25 parameters ($k_1 = 1.2$ and $b = .5$, for essentially the same formula), suggested by feedback on the data from previous TREC runs. The value they suggest offer a significant improvement over the ones we used.

- Due to a mistake in term processing, numbers were not present in the index used to run our queries. As a result, the 1890 census manual queries could not be properly written, and the automatic ones ended up in just searching for "census". We patched the query in the manual runs using the same expansion used for automatic runs on the term sequence "1890 US census".

- In automatic runs, we performed stopword elimination from the titles manually. We missed some prepositions in large queries, resulting (by expansion) in a preposterous answer time of about 10 minutes, which affected significantly the average time.

5 Acknowledgements

We acknowledge the contribution of the students of "Algoritmica per il web", Luca Natali, Mauro Mereu, Alessio Orlandi, and Roberto Valletta, who implemented some new features in MG4J, and in particular developed the new TREC document factory.

Run	bpref Submitted	bpref New	Run	map Submitted	map New	Run	P@10 Submitted	P@10 New
Manual runs			Manual runs			Manual runs		
B	-	0.3995	B	-	0.2924	B	-	0.5900
V	0.3835	0.3891	V	0.2386	0.2418	V	0.4800	0.4820
BV	0.3944	0.4057	BV	0.2822	0.2965	BV	0.5500	0.5720
BBV	**0.3940**	**0.4073**	**BBV**	**0.2833**	**0.3030**	**BBV**	**0.5880**	**0.6060**
BVV	0.3944	0.4032	BVV	0.2772	0.2840	BVV	0.5420	0.5340
Automatic runs			Automatic runs			Automatic runs		
B	-	0.3653	B	-	0.2868	B	-	0.5268
V	0.3772	0.3814	V	0.2643	0.2659	V	0.4899	0.4906
BV	**0.3765**	**0.3911**	BV	0.2914	0.3067	BV	0.5054	0.5188
BBV	0.3692	0.3902	**BBV**	**0.2866**	**0.3118**	**BBV**	**0.5081**	**0.5315**
BVV	0.3794	0.3890	BVV	0.2882	0.2979	BVV	0.4926	0.5054

Table 1: A comparison of bpref, map and P@10 values for the original submitted runs and for the runs obtained after the fixes described in Section 4. The value for B (pure BM25) was not submitted and it is shown here as a baseline. We highlighted the best runs (after the abovementioned fixes), which always involve a combination of BM25 and `VignaScorer`.

References

[1] Roi Blanco and Alvaro Barreiro. Document identifier reassignment through dimensionality reduction. In *Proc. of the 27th European Conference on Information Retrieval Research ECIR2005*, number 3408 in Lecture Notes in Computer Science, pages 375–387, 2005.

[2] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):711–726, 2004.

[3] Paolo Boldi and Sebastiano Vigna. The WebGraph framework I: Compression techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 595–601, Manhattan, USA, 2004. ACM Press.

[4] Paolo Boldi and Sebastiano Vigna. Compressed perfect embedded skip lists for quick inverted-index lookups. In *Proc. SPIRE 2005*, number 3772 in Lecture Notes in Computer Science, pages 25–28. Springer–Verlag, 2005.

[5] Paolo Boldi and Sebastiano Vigna. Efficient lazy algorithms for minimal-interval semantics. In Fabio Crestani, Paolo Ferragina, and Mark Sanderson, editors, *Proc. SPIRE 2006*, number 4209 in Lecture Notes in Computer Science, pages 134–149. Springer–Verlag, 2006.

[6] Stefan Büttcher and Charles L. A. Clarke. Efficiency vs. effectiveness in terabyte-scale information retrieval. In *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings*, 2005.

[7] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. An algebra for structured text search and a framework for its implementation. *Comput. J.*, 38(1):43–56, 1995.

[8] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple BM25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international Conference on Information and Knowledge Management*, pages 42–49, New York, NY, USA, 2004. ACM Press.

[9] Lei Shi and Fabien Campagne. Building a protein name dictionary from full text: a machine learning term extraction approach. *BMC Bioinformatics*, 6(88), 2005.

[10] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, second edition, 1999.