# MG4J at TREC 2005*

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

**Abstract**

MG4J participated in two tracks of TREC 2005 — the *ad hoc* task and the efficiency task of the Terabyte Track (find all the relevant documents with high precision from 25.2 million pages from the `.gov` domain). It was the first time the MG4J group participated to TREC, and we concentrated our efforts on the *ad hoc* task, using a combination of techniques based on a new multi-index minimal-interval semantics and PageRank.

## 1 Introduction

MG4J is Java indexing system that we developed in the last three years to support searches over the crawls performed by UbiCrawler [3]. Initially a loosely coupled set of classes supporting standard text-indexing techniques inspired by MG [12], 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 [2] and for building databases of protein names from textual documents [11].

After the first implementation phase, MG4J has been used as a playground for research ideas in text retrieval. We developed a new skipping system [5] based on the embedding of compressed perfect skip lists, and we extended the classical Clarke–Cormack–Burkowski [9] lattice for structured queries to support multiple indices and negation. We are also developing new scorers based on that extension.

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 has been partially supported by a "Finanziamento per grandi e mega attrezzature scientifiche" of the Università degli Studi di Milano; the first author was also supported by the MIUR COFIN Project "Linguaggi formali e automi".

## 2 Indexing

We did not have the time to develop an *ad hoc* document factory (see the MG4J documentation) and WebGraph [4] support for TREC; thus, we resorted to the trick of recasting the TREC data in the UbiCrawler format. This slowed down somehow the compression phase, as that format compresses each document separately. No stemming or normalisation was performed, except for downcasing. All terms were indexed.

As a part of the indexing process we computed the web graph of the GOV2 collection using WebGraph, and computed PageRank [10] with damping factor 0.85. Albeit the score provided by PageRank was used by the search engine, it had a low weight, as we estimated that PageRank on such a small graph would have had limited significance.

## 3 Querying

MG4J makes it possible to combine several indices over the same document collection; the semantic to multi-index queries is novel, and it is in our opinion one of the most remarkable features of MG4J. 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) 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 append new results from additional queries. This has been used in the *ad hoc* task 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.

We briefly recall the basics of minimal-interval semantics [9]. Intervals of integers are used to identify *witnesses*, that is, regions of the document that satisfy the query. Intervals have a natural partial order defined by inclusion. The distributive lattice used for minimal-interval semantics is the sets of *filters*[1] of intervals ordered by inclusion. Filters can be identified with the *antichain*[2] of their minimal elements. The antichain representation makes it possible to compute easily the lattice operations, and indeed MG4J contains new lazy linear algorithms to this purpose [7]. The antichain representation gives also a natural interpretation of the lattice: each element is a set of *minimal witnesses* of the document satisfying the query.

The definition we just gave is essentially identical to that given in [9], with the notable exception of the addition of an element, the set containing all intervals (represented as an antichain by the empty interval only), which is the top of the lattice, denoted as usual by 1. The natural Heyting algebra structure of the lattice has a negation operator the sends all nonzero elements to 0, and 0 to 1. We use the negation of the Heyting structure to interpret negation in queries: the intuitive interpretation is that anything true becomes false, but false becomes an *unprovable truth* — a truth featuring only a *generic witness* (the empty interval). The addition of 1 is essential to this purpose, or we could emit meaningless witnesses, as the top element would be the set of all singleton intervals.

The consecutivity operator is a restricted $\wedge$ operator that works on a single index and takes

[1] A *filter* (a.k.a. *upper set*) in a partially ordered set $P$ is a subset $X \subseteq P$ such that $x \in X$, $x \leq y$ implies $y \in X$.

[2] An *antichain* in a partially ordered set $P$ is a subset $X \subseteq P$ such that $x \leq y$ implies $x = y$ for all $x, y \in X$.

into consideration only tuples of intervals that are consecutive, whereas the low-pass operator eliminates from an antichain intervals longer than a given threshold (note that necessarily 1 is the identity for consecutivity, and it is a fixed point for any low-pass operator). Both have linear implementations in MG4J.

Finally, a query can be prefixed with an index specifier, giving the default index for all terms appearing in the query, or it can be multiplexed, that is, expanded into an $\vee$ of identical queries prefixed with all available index specifiers.

The semantics of a multi-query index is a rather delicate matter that we will describe in a forthcoming paper [6]. The basic idea is that of using the *sum of distributive lattices with* 0 *and* 1 (a.k.a. free product) to provide a natural setting for evaluating queries. If $L_i$ is the lattice of interval filters for index $i \in I$, every query has a natural evaluation in $\sum_i L_i$ by assigning to a term $t$ in index $i$ the element of $L_i$ represented by the set of occurrences of $t$, and applying the lattice operations of $\sum_i L_i$. Note that since the sum identifies 0 and 1 from all $L_i$'s, falseness and unprovable truth are lattice wide, rather than index wide, which turns out to be essential in computations.

# 4 Ranking

Ranking in MG4J is still in its infancy, and the results we obtained for TREC 2005 are essentially a first attempt. The main theme is *witness extraction* from elements of $\sum_i L_i$, and subsequent ranking (possibly with the help of PageRank).

The main problem in a multi-index semantics is that we would like to be able to compute separate witnesses for each index, both for ranking and snippeting purposes, and we would like to compute them *structurally*, that is, by inferring witnesses of a formula from the witnesses of its components. These requirements quickly lead to some technical difficulties: for instance, consider the formula

$$(\text{title} : \mathbf{goo} \vee \text{text} : \mathbf{foo}) \wedge \text{text} : \mathbf{bar}.$$

Assuming that we have given semantics to the query in $\sum_i L_i$, we would like to define structurally a witness function $[\![-]\!]_{\text{text}}$ in $L_{\text{text}}$ and a witness function $[\![-]\!]_{\text{title}}$ in $L_{\text{title}}$ expressing the witnesses of the query *for a specific index*. Usually this would imply the definition of semantic counterparts of the logical operators.

There are a few obvious constraints on such witness functions: they must preserve the Boolean value (only false queries are mapped to 0); they should provide witnesses (true queries are never mapped to 1 for all indices); finally, they should restrict to the identity on each index (when using just one index, the witnesses are those given by the good old Clarke–Cormack–Burkowski semantics).

Consider now a document that satisfies the query above, but does not contain **foo** in its text. What is the value that we are going to assign to $[\![\text{title} : \mathbf{goo} \vee \text{text} : \mathbf{foo}]\!]_{\text{text}}$? If we try to reason about each index separately, we should conclude that $[\![\text{title} : \mathbf{goo} \vee \text{text} : \mathbf{foo}]\!]_{\text{text}} = 0$ and the same for the whole formula, whichever value we assign to $[\![\text{text} : \mathbf{bar}]\!]_{\text{text}}$. But if the title contains **goo** and the text contains **bar** we end up with a true query mapped to 0 for the text index.

There is no easy way out: the deep problem is that defining a separate semantic function for a certain index implies that the Boolean reduction of that value *will not depend on the index only*. In other words, the Boolean value of the subformula might be true because another index (in our example, the title) is providing witnesses.

### 4.1 Presentation Maps

By *witness extraction* we mean the construction of a *presentation map* $p : \sum_i L_i \to \prod_i L_i$ that, given the semantics of a query, defines an antichain of intervals for each index. For a single index one can simply use the identity, but, as we argued in the previous section, the situation for multiple indices is much more intricate.

MG4J uses a presentation map based on the $\vee$-irreducible-elements representation theorem [1]. Due to the theorem, the semantics of every query can be expressed uniquely as a disjunction of *minterms* — conjunctions of intervals from distinct indices. This means that the semantics of a query can be seen as a list of sets of interdependent intervals, where each set contains at most one interval from each index. Given an element $x$ of $\sum_i L_i$ representing the semantics of a query, we are going to assign to each index the $\vee$ of all intervals for that index that appear in some minterm of $x$, or 1 if the index does not appear in any minterm, unless, of course, the query is false, in which case we assign 0 for all indices. This provides all available information to the user, even if some interdependence is lost, and agrees to the principles we discussed above.

Unfortunately, this assignment is not structurally computable: thus, we define a structural approximate computation for it, which is actually implemented in MG4J. We claim that for all practical purposes the approximation is very good; in fact, for a large class of queries (and for most real-world queries) it does coincide with the actual value. For more information, see [6].

Once witnesses are available for each index, a ranking process is applied. Clarke and Cormack [8] have proposed a scorer based on interval lengths, and MG4J provides an implementation.

For normalisation reasons however, we experimented a new scorer that never exceeds 1. Let us define the *extent* of a query as 1 for single-term queries, and then summing up over $\wedge$ and minimising over $\vee$. To give our bounded score for a query, we start with score $s = 0$ and a *residual* $r = 1$. For each witness $[a\,..\,b]$, we move a fraction $\min(1, e/(b-a+1))/2$ of $r$ into $s$, where $e$ is the extent of the query. Thus, shorter intervals move more residual into the score, and intervals arriving later (that is, further down in the document) move less score w.r.t. intervals appearing before. A large number of intervals moves to score towards one, but with a saturation effect that prevents overflow.

## 5 Results

We submitted just manual runs. Thus, the results are strongly biased by our own knowledge of the English language and of the specific topic. Overall, the results obtained by MG4J are average. Nonetheless, they have a large variance — on a few topic we obtained the best bpref/map values, and on a few the worst ones. This reinforces our idea that the primitives provided by minimal-interval semantics are very powerful search tools but that, as any manual search tool,

carelessness in the costruction of the query may lead to very bad results. On a side note, MG4J was by far the biggest contributor of unique releveant documents to the document pool — most probably because by using techniques that were completely different from the other participants' we hit on documents that BM25 or other similar weight functions would not have considered relevant.

## 6 Presenting Results

Albeit the form in which results are presented is not part of the TREC evaluation, we believe that in web search engines presentation is essential. Whereas most traditional IR is involved in retrieving as many relevant documents as possible, retrieving many relevant documents is not a big problem on the web — the user won't be able to look at all results anyway: the main problems are *ranking* and *presentation*. The user should see some relevant documents in the first ten results, and should be able to judge quickly that they are relevant. Note that in many cases it is practically impossible to rank all documents in the index, so prediction techniques must be used to guarantee that with high probability some relevant document is already in the top ten, even if not all documents have been seen.

Minimal-interval semantics has the useful property of providing directly the user with a clear feedback of the relevance of a document, as text snippets can be quickly created and displayed from witnesses. Reading a small snippet satisfying the query can give a hint that a listed document is relevant, and minimality guarantees that we will be presenting the most concise snippets available.

## References


[1] Garrett Birkhoff. *Lattice Theory*, volume XXV of *AMS Colloquium Publications*. American Mathematical Society, third (new) edition, 1970.

[2] 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.

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

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

[5] Paolo Boldi and Sebastiano Vigna. Compressed perfect embedded skip lists for quick inverted-index lookups. In *Proc. SPIRE 2005*, Lecture Notes in Computer Science. Springer–Verlag, 2005. Short paper.

[6] Paolo Boldi and Sebastiano Vigna. Multi-index interval semantics with application to snippet extraction, 2005. Preprint.

[7] Paolo Boldi and Sebastiano Vigna. Efficient lazy algorithms for minimal-interval semantics, 2006. Submitted for publication.

[8] Charles L. A. Clarke and Gordon V. Cormack. Shortest-substring retrieval and ranking. *ACM Trans. Inf. Syst*, 18(1):44–78, 2000.

[9] 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.

[10] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, Stanford University, Stanford, CA, USA, 1998.

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

[12] 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.