Information Processing and Management 000 (2016) 1–18

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

Fast top-k preserving query processing using two-tier indexes

Caio Moura Daoud[a,**], Edleno Silva de Moura[a,*], Andre Carvalho[a], Altigran Soares da Silva[a], David Fernandes[a], Cristian Rossi[b]

[a] Institute of Computing, Federal University of Amazonas –Av. Gen. Rodrigo Otávio, 3000, Manaus 69077-000, AM, Brazil
[b] Neemu S/A, Av. Via Lactea, 1374, Manaus 69060-020, AM, Brazil

ARTICLE INFO

Article history:
Received 3 January 2015
Revised 24 March 2016
Accepted 27 March 2016
Available online xxx

Keywords:
Information retrieval
Query processing
Inverted index
Search system
BMW

ABSTRACT

In this paper we propose and evaluate the *Block Max WAND with Candidate Selection and Preserving Top-K Results* algorithm, or BMW-CSP. It is an extension of BMW-CS, a method previously proposed by us. Although very efficient, BMW-CS does not guarantee preserving the top-*k* results for a given query. Algorithms that do not preserve the top results may reduce the quality of ranking results in search systems. BMW-CSP extends BMW-CS to ensure that the top-k results will have their rankings preserved. In the experiments we performed for computing the top-10 results, the final average time required for processing queries with BMW-CSP was lesser than the ones required by the baselines adopted. For instance, when computing top-10 results, the average time achieved by MBMW, the best multi-tier baseline we found in the literature, was 36.29 ms per query, while the average time achieved by BMW-CSP was 19.64 ms per query. The price paid by BMW-CSP is an extra memory required to store partial scores of documents. As we show in the experiments, this price is not prohibitive and, in cases where it is acceptable, BMW-CSP may constitute an excellent alternative query processing method.

1. Introduction

Systems that allow fast access to information available in large textual collections are nowadays an integral part of day to day activities of billion of users around the world. Such activities include using web search engines, e-commerce search systems and textual contents of any modern database management system, among others. Users of such systems are usually satisfied with only a few, highly relevant results for each submitted query. Thus, many researchers have been working to develop newer methods to speed-up the computation of only the top results of a query, which are the ones usually presented to the users.

Modern search systems usually deploy a number of different sources of relevance evidence for determining the top-k results for answering each query. For instance, an e-commerce search system may take into account not only the textual description of each product, but also its popularity and price. Citing another example, in addition to the page contents itself, web search engines use information such as the titles of the pages and link structure. The sources of relevance evidence are combined using a myriad of approaches, such as learning to rank techniques, in order to produce the final ranked results.

* Corresponding author. Tel.: +55 92 33052808.
** Corresponding author. Tel.: +55 92 981437321.
E-mail addresses: caiodaoud@icomp.ufam.edu.br (C.M. Daoud), edleno@icomp.ufam.edu.br (E. Silva de Moura).

http://dx.doi.org/10.1016/j.ipm.2016.03.005
0306-4573/

The combination of multiple sources of evidence is usually expensive, and thus it is not applied to the full set of documents that suffice the query being processed (Carvalho, Rossi, de Moura, Fernandes, & Silva, 2012; Rossi, de Moura, Carvalho, & Silva, 2013). Usually, only a small subset of documents selected from an initial process of ranking is considered in a second, more sophisticated ranking phase. The initial process of computation of the ranking consists of applying a basic IR model, such as BM25 (Robertson & Walker, 1994) or the Vector Space Model (Salton, Wong, & Yang, 1975), to compute an initial rank of top results. Most of the research aimed at reducing computational costs in query processing is related to the reduction of number of documents considered while creating this initial ranking.

In this paper we propose and evaluate *Block Max WAND with Candidate Selection and Preserving Top-K Results* algorithm, or BMW-CSP, a fast query processing method that ensures that the top-k results will have their rankings preserved. This method is an improved and extended version of BMW-CS, *Block Max WAND with Candidate Selection*, which was also previously proposed by us (Rossi et al., 2013) .

Although very efficient, BMW-CS has the disadvantage of not guaranteeing to preserve the top-*k* results for a given query. In our preliminary study, we show through experiments that the use of BMW-CS does not cause significant loss in quality of results when applied in a search system, still the property of preserving top results is considered important in literature. Further, the use of BMW-CS is particularly risky, since it may completely remove important results from the final ranking. The new algorithm proposed here, BMW-CSP, maintains the time performance advantages presented by BMW-CS, while presenting the advantage of guaranteeing that the top-*k* ranking results will remain unchanged. In addition, our experiments indicate that BMW-CSP is faster than the best baselines we found in the literature.

The remainder of this article is structured as follows: Section 2 presents the background and related research necessary to better understand the proposed methods. Section 3 presents our method. Section 4 presents the experimental results. Finally, Section 5 presents the conclusion and prospective future research.

2. Background and related work

Search systems that deal with large textual collections use index structures for allowing fast query processing. One of the most adopted index structures is known as *inverted file* (Baeza-Yates & Ribeiro-Neto, 2011). An inverted file contains, for each term *t*, where a term is usually a word, the list of documents where it exists. For each document in the list, it also stores information used to compute the importance of the document in the list, or its *weight*. Inverted lists usually store the frequency of the term in the document to compute such importance. This list of pairs of document and term frequencies is called an *inverted list* of *t* and is used to measure the relative importance of terms in the stored documents. Each document is represented in these lists by a value named *document ID*, referred to as *docId* in this article. Depending on the size and number of documents in the system, the inverted files may become huge, being stored in a compressed format to save memory space.

Queries are usually processed by traversing the inverted files either in a Term-At-A-Time (TAAT) or a Document-At-A-Time (DAAT) query processing. In the TAAT strategy, the inverted lists are sorted by term impact in non-increasing order and the query results are obtained by sequentially traversing one inverted list at a time. In the DAAT strategy, the inverted lists are sorted by *docIds*, which allows the algorithms to traverse all the inverted lists related to a query in parallel.

The TAAT approach takes advantage of the fact that sequentially accessing each inverted lists may lead to a speed up in query processing. Its main disadvantage is to require the usage of large amounts of memory to store partial scores achieved by each document when traversing each inverted list. These partial scores should be stored to accumulate the score results obtained by each document when traversing each inverted list. The final ranking can be computed only when all the inverted lists are processed.

Several authors proposed methods to discard partial scores, thus reducing the amount of memory required to process queries in the TAAT mode (Anh & Moffat, 2006; Anh, de Kretser, & Moffat, 2001; Strohman & Croft, 2007). Some of the benefit of the TAAT approach is lost when the queries are processed in the main memory, since the gains in performing sequential access to the lists, instead of random access, are smaller in main memory than while performing disk accesses.

Our algorithm here adopts the DAAT approach. In the DAAT, the full scores of documents are computed while traversing the lists in parallel. Since the lists are ordered by docIds, it is possible to skip documents that are unlikely to be relevant. The memory requirements are fairly smaller than in TAAT, since DAAT approach requires only the current top-k answers to be stored at each moment while processing a query. The price is that the algorithm changes from list to list all the time, thus performing less sequential accesses when compared to TAAT.

As in TAAT approach, several authors have proposed algorithms and data structures to accelerate the query processing in the DAAT approach. For instance, data structures to allow fast skipping in the inverted lists, named as skiplists (Moffat & Zobel, 1996), are adopted to accelerate the query processing. Skiplists divide the inverted lists into blocks of entries and provide pointers for fast access to such blocks, so that a scan in the skiplist determines the block, in which a document entry may occur in the inverted list.

Several DAAT approaches can be considered related to our proposal. Broder, Carmel, Herscovici, Soffer, and Zien (2003) proposed a successful strategy for query processing, known as WAND. In WAND, a heap of scores is created to maintain the top-k documents with larger scores during each step of the query processing. The smallest score present in the heap at each moment is taken as a *discarding threshold* to accelerate the query processing. A new document is evaluated and inserted in the heap only if it has a higher score than the *discarding threshold*.

Please cite this article as: C.M. Daoud et al., Fast top-k preserving query processing using two-tier indexes, Information Processing and Management (2016), http://dx.doi.org/10.1016/j.ipm.2016.03.005