MWAND: A New Early Termination Algorithm for Fast and Efficient Query Evaluation

Zemani Imene Mansouria*, Zekri Lougmiri*, Senouci Mohamed

Department of Computer Sciences, LAPECI Laboratory, Ahmed Ben Bella Oran1 University (Algeria)

Received 22 October 2018 | Accepted 22 March 2019 | Published 10 April 2019

Abstract

Nowadays, current information systems are so large and maintain huge amount of data. At every time, they process millions of documents and millions of queries. In order to choose the most important responses from this amount of data, it is well to apply what is so called early termination algorithms. These ones attempt to extract the Top-K documents according to a specified increasing monotone function. The principal idea behind is to reach and score the most significant less number of documents. So, they avoid fully processing the whole documents. WAND algorithm is at the state of the art in this area. Despite it is efficient, it is missing effectiveness and precision. In this paper, we propose two contributions, the principal proposal is a new early termination algorithm based on WAND approach, we call it MWAND (Modified WAND). This one is faster and more precise than the first. It has the ability to avoid unnecessary WAND steps. In this work, we integrate a tree structure as an index into WAND and we add new levels in query processing. In the second contribution, we define new fine metrics to ameliorate the evaluation of the retrieved information. The experimental results on real datasets show that MWAND is more efficient than the WAND approach.

Keywords

Evaluation Measures, Information Retrieval, Large Inverted List, MWAND, Query Processing, Top-k, WAND.

DOI: 10.9781/ijimai.2019.04.002

I. Introduction

PRATICAL web search engines are very complex with the goal of returning fast and precise results. The result must be both effective and efficient. These search engines use techniques and algorithms of query processing, such as WAND algorithm, to return a set of ranked documents results named Top-k. These algorithms are executed on a data structure called inverted index [1]. Such structure gives for every term the set of documents in which it appears with additional information, like the term frequencies TF, the list of positions in every document, the format and the size in which it is written. Such construction generates a very large index. In fact, its size is larger than the set of original documents. As a consequence, traversing this index becomes the major bottleneck in query processing. In fact, it is not suitable, really not practical or impossible, to sweep all posting lists. An early termination algorithm is so recommended for such situation. It can return the exact Top-k without scanning the entire posting list. Note here, that a posting list is the part of the inverted index charged in the memory for treatment. We must note also here that the lists are ordered in an ascendant order according to documents numbers [2][3] or on descendant order on TF [4][5]. The choice is done according what the algorithm designer wants. In order to reduce the information representation in the posting lists, a set of compressing techniques have been proposed [6][7][8].

In information retrieval, two major and basic alternatives have been proposed for traversing the posting lists. It is about TAAT (Term-At-A-Time) and DAAT (Document-At-A-Time) strategies [9].

* Corresponding author.

E-mail addresses: imenezemani@yahoo.fr (Z. I. Mansouria), lougmiri@yahoo.fr (Z. Lougmiri)

In fact, SAAT(Score At A Time), GAAT(Graph At A Time) and RAAT(Rank At A Time) and JASS(SAAT) are additional strategies which are proposed for remedying the first strategies weaknesses [10] [11][12].

Since the first works in the field of information retrieval [13], the stopping condition is an interesting part of every early termination algorithm. It consists in ending the execution if k responses are computed even if there are more important results with ranks greater than k. A document is considered relevant if its score is greater or equal to a certain bound. The threshold algorithm TA of Ronald Fagin [14] is one of the most popular algorithms in the context of databases. As information systems are so large and as search engines must deal with large dataset, the WAND algorithm has became unavoidable. It has been used in a number of commercial search engines [15]. It has the ability to skip in an intelligent manner some documents and parts of posting lists according to a precise test, as we will see in next sections.

Really, the strongest weakness of early termination algorithms resides in the lack of precision in their responses. For Top-k and for a query q of length Lq, it is usual that WAND misses in Top-r, with $r \leq k$, some documents which share a high number of terms with the query. It is about this ascertainment that we built our solution. In this paper, we focus on early termination and we propose a new extension to algorithm WAND. Our aims are: a) To return all totally relevant documents that contain query terms. b) To reduce the operations number in query processing. c) To ameliorate the results quality by ameliorating the responses precision. In particular, we propose new fine metrics to measure the relevance degree of the returned documents. Compared to the naive approach that contains at least one of the query terms, our approach returns the relevant documents ranked first, without any loss in precision or recall or in new proposed metrics.

The reminder of the paper proceeds as follows: Section II gives a

representation of index structure, TAAT / DAAT strategies and early termination. The WAND approach is detailed in section III. In section IV, we quote our contributions and section V presents the details of our propositions. In section VI, we describe our experimental results. Finally, we conclude in section VII and discuss our future work.

II. Background

In this section, we provide a background on index structure, compression, early termination and index traversing strategies.

A. Index Structure

In order to evaluate queries in search engines, a well structure is constructed; it gives with precision all information about every term in the set of documents. This structure is called inverted index as it captures the list of documents which contain this term with other information. So, the term is used as a key access to such structure. Every line from this structure is called a posting list when it is charged in memory. This structure was largely presented and explained in literature. Works [7][16][17][18][1] have presented all what can be related to the construction and the use of such structure.

For a term t_i which appears, a record of its inverted list can have the next format:

d_j	TF_{ij}	$pos_1, pos_2, \ldots$

Where d_j is a unique document number, called also document identifier, in the set of documents collection, this number is assigned during the crawling process. TF_{ij} is the number of occurrences of t_i in d_j. The list $pos_1, pos_2,\ldots$ contains all positions of t_i in this document. It is useful essentially for searching phrases. The set of terms composes what is so-called the vocabulary or the dictionary, it can be so large, especially when different languages are considered. As the web and information systems are dynamic, the number of documents (documents, web pages,…) is growing perpetually, which induces that the size of posting lists grows exponentially also. As a result of such evolution systems behavior, the length of inverted lists becomes an important bottleneck [19].

In order to exploit efficiently the inverted lists, many organizations have been proposed. It is possible to present them briefly as follows [8]:

Document-Sorted Index: It is the first representation of the inverted index. In this situation, the inverted lists are sorted by document identifier [8][20][3].

Impact-Sorted Index: In this situation, the importance is given to the terms. Their scores will contribute to computing the similarity between a query q and the set of documents. The inverted lists are sorted by their impact; BM25 [21] is the most popular method which has implemented such idea. [22] and [10] have proposed and compared another method for retrieving information based on impact-sorted index.

Impact-layered index: This is a combination of the two precedent presentations. It computes the impact of every term according to a function similarity and each list is partitioned into a number of layers, such that the scores of one layer are higher than the next [8].

It is important to see that these organizations are built during the indexing process. Every organization is suitable for the solutions designers; it depends on what these ones want.

With the growing amounts of data on the internet, every time new documents (books, papers, tweets, Web pages, comments ...) are inserted. The preprocessing step will take a huge time for preparing indexes. The major problem will be finding a suitable manner for representing information. As the number of documents is too large, it will be essential to reduce the size of the inverted index. Here a compressing method will be helpful. Every document is designed by a unique integer. The most used method is to compute the gap between successive numbers of documents that contain a term *t*. For example, suppose $d_1, d_2, d_3, \ldots, d_k$ are the identifiers of the documents which contain the term *t*, the gap method is to store them in the inverted list as follows [6]:

$d_1; d_2\text{-}d_1; d_3\text{-}d_2; \ldots; d_k\text{-}d_{k-1}$

It has been shown in [24] that if the database can fit in the memory, so the accesses to it will be faster than the use of indexes. This result is so easy to see. Besides the inverted index contains all the vocabulary of the document collection, it contains more and more information, like TFs and positions of terms, so its size is larger than document collection's size. So, it will be impossible to fit it in the memory without compressing it. For this object, the first step is to compute the d_gaps, as shown before. The second step is to implement a well method for compressing it and getting information without decompressing. Sholer [23] has shown that for documents identifiers it is well to use the Elias [24] code when these numbers are too large; but for the positions, as they are little numbers, it is better to code them with the Gamma code.

In the same logic, a high number of methods of compression have been proposed in the literature. Everyone depends on the situation. It is possible to cite for classical methods: Elias encodings [24] and Golomb/Rice's encoding [25]. Newer methods are VByte [26], Simple [27], Interpolative [28], PForDelta [29]. Other techniques are proposed in [30][31]. Trotman [32] gives a comparison between different compressing methods. More material can be found in [7].

B. Early Termination

As we have said before, it is impossible to scan all posting lists for evaluating a query. Since first works, it has been questioned to stop execution if a certain condition is verified [13].The ranking operations usually apply Early Termination algorithms to avoid fully processing of documents. We say that the set of responses is not exhaustive; so many important responses can be avoided. There are three cases in early termination:

Stop early: In this case, the most promising documents are ranked first in posting lists and the execution will stop once k documents are obtained according to a well defined monotonic function. The algorithms like TA, FA, and NRA algorithms of Fagin [33][34][35] are examples of such type. They can be used for computing the most important objects, like Web pages or cars, in a database, but they cannot be used in information retrieval of context as the lists here cannot have the same format.

Skips: By defining a certain criteria, it will be possible to skip a set of documents if the criteria are not respected. WAND [20][11] are examples of such algorithms.

Score only partially: Algorithms evaluate documents by computing only approximate scores if their score is lower than a certain value [36].

Algorithms like WAND, BM25, BMW, BMI use early termination techniques avoiding processing complete lists [36][37][38][39].

C. Traversing Index Strategies

In the query processing domain, Turtle and Flood [9] classified evaluation into two main classes.

Term-At-A-Time (TAAT): The strategy traverses query terms term-by-term, while partial documents score are cumulated [40][10]. In this strategy, more information is stored in inverted lists.

Document-At-A-Time (DAAT): This strategy is very fast and processes all posting lists in a parallel manner, with respect to a single document before moving to the next one. DAAT is destined for big

collections processing. Examples: WAND [20], MaxScore [41].

In 2011, Fontoura et al. [42] compared TAAT and DAAT, to show efficiency gains in DAAT strategy [20]. Moreover, TAAT strategy happens to be more complex than that of the DAAT. At the opposite, [43] has shown that TAAT outperforms DAAT. We think the performance depends on how every strategy is implemented.

Recall that query processing use two query models: conjunctive (AND) and disjunctive (OR) queries. In general, conjunctive queries are more significant and disjunctive queries are more expensive than conjunctive queries.

It is important to report that additional methods have been proposed, even if they are deduced from the above strategies.

SAAT (Score At A Time): This strategy is considered when impact-based indexing is used. It searches all lists and processed the posting lists in the decreasing order of the impact values [10][11]. Hao [12] uses a different version of previous works for applying this technique for pseudo-relevance feedback. Zhang [44] made a revision of this technique and compares it to TAAT and DAAT. Joel [45] proposed a set of heuristics for studying efficiency and effectiveness of this technique. This method was implemented over JASS system [46]. This one is dedicated to evaluate queries according to the impact-sorted indexes. This method was compared to JAAS.

RAAT (Rank At A Time): This strategy was proposed by [47]. It is of type impact scores and it combines Boolean intersection queries. The lists are presented in a descending order. The head of every list is the term with the high score. Every list is considered as a query.

GAAT (Graph At A Time): This technique is not in topics here, but we want to cite it as a manner for searching information where information can be presented as a graph, such as RDF (Resource Description Framework), citation between co-authors in DBLP and protein interactions [48].

D. Works Around WAND

In this section, we will present the important works that have studied WAND; we have discarded works which have general aspects of information retrieval as they are not the subject of this paper. We want to show that WAND is really at the state of the art since it was published in 2003. It is cited as the most method which presents correctly the DAAT technique. It has the ability of skipping parts if, at a certain moment, the next documents scores cannot exceed an upper bound. [8] is an enhancement of WAND-style. Ding and Suel defined BMI method, also called BMW. It has augmented the inverted lists by well defined blocks. Every block is delimited by an upper bound. If the score of a document does not exceed, a skip is performed. So BMI will be faster than WAND. [3] defines new bloc indexes which on authors define a new hierarchical algorithm. This work has presented also a comparison between WAND and MaxScore [11]. Rojas [19] proposes a 2-steps method for parallelizing WAND. The objective is to reduce the inter-processor communication and running time cost. This work invokes a multithreading approach to exploit the multi-core parallelism. This same work was resumed in [15] by new formulas. The objective here is to reduce memory usage and computation cost based on WAND and BMI. Andrei Broder, the WAND's designer, proposes SWAND (for Sampling WAND) in [49]. SWAND aims to obtain performances in order to be inserted in any search engine. An object oriented approach is followed and other logic operators are inserted. The comparison between SWAND, IBM's Trevi and JURU search engines is done. SWAND has presented better performances.

In order to couple efficiency and effectiveness, [50] proposes a new method for dynamic pruning. Authors made an adjustment on the threshold and k of WAND in order to recover missed responses. This work has largely benefit from the safe-to-rank of WAND and from its manner of skipping lists. [51] proposes to save pairs of terms in posting lists in order to get more efficient WAND and MaxScore [41]. The new versions are called WANDP and MaxScoreP. [52] studied the effect of document identifier ordering on the dynamic pruning. Authors propose to apply random, document length and url ordering.

[53] examines multi-stage retrieval architecture. This one consists of a candidate generation stage, a feature extraction stage, and a reranking stage using machine-learned models. Authors studied the NDCG(Cumulative gain-based evaluation) metric according to WAND, BM25 and SvS [54]. Note that SvS (Small versus Small) is a method which searches the intersection between a set of ordered lists. This study has shown to be better than the two other methods. [55] has introduced a new bloom filter variation called Bloom filter chains on WAND for generation of a new method called Bloom WAND in order to retrieve tweets in real time.

[56] is an important work as authors work a lot on the aspects of language model. [56] has studied whether WAND is effective in this context. Experimentations have shown that it is not so helpful in this domain. Also, as authors of [57] work on selective search, they studied intensively whether WAND can be useful for selective search. They demonstrate than when indexes are well structured, WAND can be too effective. The effectiveness here is justified by the fact that indexes are grouped by subjects.

As BMW or (BMI) [8] proposes fixed sizes of blocks, [58] uses variable-sized blocks. Authors begun by partitioning blocks after what they define an algorithm for finding an approximate solution. This solution is named VBMW.

Bortnikov [59] defines conditional-skip iterator traversal strategy for pruning dynamically Top-k responses. This method can jump to a target document while skipping all matching documents preceding the target.

Daoud [60] presents a new system called WAVES which is multi-tier indexes for fast evaluation of queries. It is suitable to note here that this one is largely inspired by BMW and MMBW (Multi-tier BMW). WAVES uses BM25 for computing scores and stores upper bounds on blocks like BMW.

Based on WAND, MaxScore and BMW, authors of [61] have proposed a new method for computing pages's scores. The imperfection of BMW is that it does not support static scores. A static score is one given by a function like PageRank of Google.

Petri [62] gives a comparison between WAND, BMW as DAAT strategies with the system JASS which is SAAT strategy. Note here, that every year a set of works are published in order to compare DAAT, TAAT, SAAT and RAAT methods. Sometimes, conclusions between those works are contradictory.

Andrei Broder et al. in [63] have extended WAND by coupling it with the K-means clustering method. The cause is that this later does not scale with millions of documents. Thus this work gave the birth to WAND-k-means algorithm.

Recently in July 2018, by defining the safe-rank, Andrew Kane et al. [64] propose a new method called split-list WAND. Authors define an initial threshold and split lists on two layers according to scores. Authors shown that this proposal is better than WAND and BMW.

Also in 2018, [66] proposes a framework for predicting parameters during query-by-query evaluation. They demonstrate, by prediction, that they do not need experts' judgments. [66] cites WAND as a well understood algorithm for retrieving information in different situations.

III. The Two-Level Evaluation Process

This section describes the WAND mechanism.

As explained by Border in [20] and Oscar in [15]; WAND is a query processing method based on two levels, the first level is simple and the second level use a complex scoring to extract relevant documents. In the first level (named preliminary evaluation), WAND identifies candidate documents using approximate scoring, the advantage of the first level is that it is possible to skip evaluation of a number of documents. At the second level, the identified candidate documents are fully evaluated with precise metrics; and stored in top-k heap. The threshold is an important dynamic value used by WAND approach, varied during execution. The initial value of threshold depends on the type of query (disjunctive or conjunctive).In general, the initial value equals to zero as long as the top-k heap is not completely full, once the heap is full, threshold equals the minimum score in the top-k heaps.

A. First Level "Preliminary Evaluation"

Keeping in mind that WAND algorithm in the preliminary evaluation calculates an approximate score by summing UB (each term is associated with an Upper Bound):

$$UB(d, q) = \sum_{t \epsilon q \cap d} UB_t \quad (1)$$

$$UB_t = \alpha_t \max(w(t, d_1); w(t, d_2) \ldots) \quad (2)$$

a) α_t= tf * idf represents a function of the number of occurrences of term t in a document d, multiplied by the inverse document frequency.

b) W (t, d) represents a function of the term frequency of t in d divided by the document length |d|.

Every document with approximate score smaller than threshold will be skipped.

B. Second Level

In the second level, the candidate documents are evaluated with an exact score:

$$Score(d, q) = \sum_{t \epsilon q \cap d} w(t, d) \quad (3)$$

The top-k heap is a list of k documents results which is initially empty; an admission of a new document in top-k heap is done when the exact score of this document is greater than the threshold (threshold represents the minimum score in the heap). If the heap is full, the new document replaces the document associated with the minimum score. Documents with a score smaller than the minimum score in the heap will be skipped. Every document skipped will not be inserted in the heap results [20].

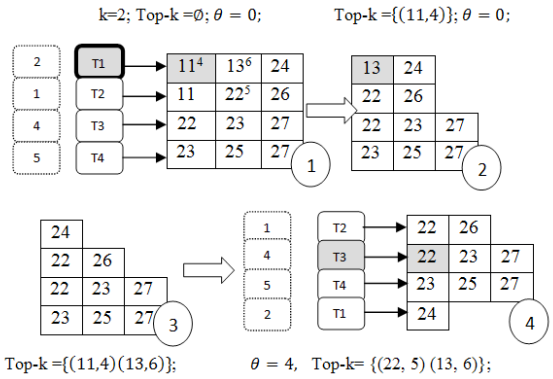

Fig. 1. A scenario of query processing "WAND" of a 4-termquery.

In Fig. 1 step (1), list of UB_t= (2, 1, 4, 5); top-k heap is empty; the initial threshold is 0 and the current pointers on the lists are (11, 11, 22, 23).Note that lists are sorted by their current documents. For (Current upper bound UB_{t1}=2, $UB_{t1} \geq \theta$) the pivot term is T1, and pivot document is 11 and score = 4≥ θ. Insert pivot document in top-k heap: Top-k = {(11, 4)}. As top-k is not completely full threshold equals to zero.

In step (2) list of UB_t= (2, 1, 4, 5), threshold equals 0 and the current pointers on the lists are (13, 22, 22, 23). For (Current upper bound UB_{t1}=2, $UB_{t1} \geq \theta$) the pivot term is T1; pivot document is 13 and score = 6≥ θ. Insert pivot document in top-k heap: Top-k = {(11, 4) ;(13, 6)}. As top-k is completely full, we update threshold θ = 4.

In step (4) lists are sorted by their current document (T2, T3, T4, T1) (22, 22, 23, 24). $UB_{t2}+UB_{t3}$=1+4=5 ≥ θ; the pivot term is T3, pivot document is 22 and score=5. Insert pivot document in top-k heap: Top-k = {(22, 5) ;(13, 6)}. Update threshold θ = 5.

IV. Our Contributions

As we have remarked, WAND is not so effective during queries' evaluation, as it can miss some important responses. We had the idea to propose a new method for recovering missed responses with the ability to propose an early termination algorithm. It is logic to see that when a user submits a query with length Lq, he will wait to responses with documents containing all, or a maximum of terms of this query; WAND can respond by a document containing few terms in first ranks, so WAND is missing effectiveness [65][66][50]. In theory of information retrieval, efficiency and effectiveness are major tradeoffs of search engines [50][67].Also it is important to see that effectiveness may have an impact on efficiency. Really all works that have extended WAND, cited in this paper, agree with us about the weakness of WAND in terms of effectiveness. In the next paragraph, we explain the cause of this imperfection.

During the steps of execution, and precisely when the threshold is equal to 0, an important time is spent as every document score is greater than this threshold; the insertions of these documents are implicit. So, it is unnecessary to test the relation between documents scores and the threshold as it will deal to a high number of fruitless rounds. This step is too hard and is the cause of the absence of effectiveness of WAND. More of this, it is possible to insert responses with high probabilistic scores but which share a part of terms with the query. Search engines always attempt to return responses in a semantic way such the responses contain a maximum terms in sharing with the queries.

In the next sections, we give in detail our proposal presented as follows:

(a) We propose modified algorithm based on WAND algorithm, which is composed of four levels of query processing;

(b) We propose to integrate a tree index;

(c) We maximize responses, which contain a maximal number of shared terms with the query;

(d) We evaluate our technique with classic metrics and with new fine proposed metrics named: fidelity, exact relevance degree, almost total relevance degree, and exact recall.

V. Our Approach

In this section we explain our algorithm and we give complete details of the four level evaluations of MWAND.

For large systems the full evaluation is an expensive task. The intention of our algorithm is to minimize the number of processing iterations and to select in first level the most relevant documents. In this paper we describe a novel extension to WAND and we employ DAAT strategy. MWAND is destined for both conjunctive and disjunctive queries that contain at least 3 terms.

Keeping in mind that query processing on WAND algorithm has two levels:

Level 1: Select a candidate document to be scored in the preliminary

evaluation.

Level 2: Select the Top-k document by exact scoring, if the score is larger than the current threshold.

Although the anticipation logic proposed by WAND has been very attractive and has inspired many researchers, there still are some palpable imperfections in the execution such as:

a) How to avoid the loss of relevant documents in the passage between operations?

b) The unnecessary approximate evaluation of several documents.

c) Are the returned documents, relevant documents?

d) Can we have a faster way, to speed up the search for terms in the index table?

A. Index Structure

Our aim is to resolve the time consuming problem, since the main task of search engines is by far the need to answer user queries within fractions of second, and since naively going through all the list of index terms can take hundreds of milliseconds.

More of this and as we have explained above, the vocabulary is too large, in order to reduce the space occupation, we choose to compute the longest common prefix between index terms [68]. This proposal reduces significantly the occupied space. According to each prefix the index is partitioned.

Our model integrates a tree index for IR systems into WAND and consists of a set of trees (Fig. 2). In which, every tree head represents an alphabet letter (a, b, c.....z) or a number (0, 1, 2.....9) or character (&.....#), and the descending nodes of the tree represent decompositions of terms in letter (Fig. 3). Each last letter term is associated with an inverted list. This index structure was chosen to optimize the search time and the indexation time.

Fig. 2. The set of index trees.

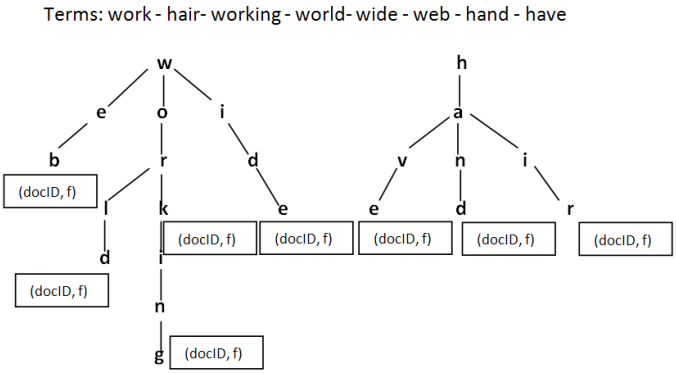

Fig. 3. Example of index tree.

Fig. 3 shows a scenario of tree index that is an example to index or to create the path of terms: t_1=work and t_2=world

*The first letter in t_1 is (w)

- Insert t_1 in tree(w);
- Create a node (o) in level m_1, and connect this node to tree head;
- Create a node (r) in level m_2, and connect this node to (o)m_1;
- Create a node (k) in level m_3, and connect this node to (r)m_2.

*The first letter in t_2 is (w)

- Insert t_2 in tree(w);
- Node (o) exist in level m_1;
- Node (r) exist in level m_2;
- Create a node (l) in level m_3, and connect this node to (r)m_2;
- Create a node (d) in level m_4, and connect this node to (l)m_3.

B. MWAND Algorithm

Our query processing algorithm is based on WAND and our scoring is based on formulas (1); (2) and (3).Our method is divided into 4 levels:

- First level "Intersection function";
- Second level "Filling function";
- Third level "Approximate evaluation of WAND";
- Fourth level "Exact evaluation of WAND".

1. First Level "Intersection Function"

The key idea of the first level is to avoid the loss of total relevant documents in the passage between operations of query processing. This intersection function takes as input parameter: The number of documents to be returned k and a query q of | q | terms (q > 2) using an index I which contains posting lists (It) sorted by DocID. The results parameters returned by intersection function are: a) The relevant top-k heap (top-k'). b) The number of documents returned k'. c) New posting lists destined to the second level. This function points on the first document in the first posting list and verifies its existence in the other lists; its role is to retrieve all the documents that appear in the | q | and | q-1 | postings. The top-k' heap returned represents the union of the set of documents that contain | q | terms ($D_{i|\,q\,|}$) and the set of documents that contain | q-1 | terms ($D_{j|\,q-1\,|}$),as explained below:

$$\text{Top-k'} = It_{|q|} \cap It_{|q|-1} \cap It_{|q|-2} \ldots\ldots \cap I_{t1}$$
$$= \{ D_{i|\,q\,|} \} \cup \{ D_{j|\,q-1\,|} \}$$

The objective of this function is to minimize the number of documents to be evaluated in the next levels.

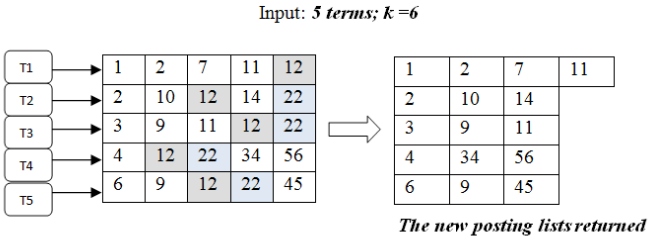

Fig. 4. Example of "Intersection function" and the results returned.

In Fig. 4 a scenario of intersection function is shown, the input parameters are a query of 5 terms (|q|=5) and k=6; parameters returned are:

- Top-k'=$\{D_{i|q|}\} \cup \{D_{j|\,q-1\,|}\}$=$\{12\} \cup \{22\}$=$\{12, 22\}$; Top-k' represents an union of sets of documents that contain 5 terms and 4 terms of query;
- k'= Card(Top-k')=2; k' represents the number of documents in Top-k';
- Intersection function returns new posting lists: I_{t1}=$\{1, 2, 7, 11\}$; I_{t2}=$\{2, 10, 14\}$; I_{t3}=$\{3, 9, 11\}$; I_{t4}=$\{4, 34, 56\}$; I_{t5}=$\{6, 9, 45\}$.

2. Second Level "Filling Function"

The basic WAND algorithm is proposed in [20], our MWAND algorithm improves it in the pivot term selecting. Critical of the computation of WAND approach is the definition of the top-k filling:

the initial value of threshold equals to zero as long as top-k heap is not completely full. As it is shown in Fig. 1: ∀ (k ∈ N); For k first iterations of query processing, WAND algorithm loses time passing through the two phases of approximate and exact evaluation. In this context, our approach adds a new level, that have to optimize the processing time, using a filling function to avoid the time lost in the approximate evaluation phase. This function takes as input parameters: a) the new posting lists returned by the first level b) the value of k' returned by the first level; and returns as result: a) initial top-k heap b) value of threshold c) new lists destined to WAND. In our function we eliminate the approximate evaluation and unnecessary tests; the sum of upper bound score is greater than the value of threshold that equals zero ($\forall$ *approximate score* $= UB\,(d,q) = \sum_{t \in q \cap d} UB_t \geq \theta \geq 0$) for k first iterations. Thus, our filling function of top-k heap selects rapidly a set of initial pivot documents. Fig. 5 shows an example on filling function, to select a set of pivot documents. In this example, all posting lists are sorted by their current DocID (1, 2, 3, 4, 6). We know that It_1 current DocID is (1), It_2 current DocID is (2), It_3 current DocID is (3), It_4 current DocID is (4) and It_5 current DocID is (6). The maximum current DocID is named "MaxCurDocID" and its value is equal to 6. In this scenario, we select all DocID smaller than maximum current DocID (candidate document = DocID \ if DocID ≤ MaxCurDocID). The set of documents selected is {It_1 (1, 2); It_2 (2); It_3 (3, 5); It_4 (4); It_5 (6)} = {1; 2; 3; 4; 5; 6}.This list is sorted in ascending order and the k" first documents are inserted in top-k heap and their exact scores can be calculated. Knowing that $k''=k-k'$. The initial *top-k"* heap selected by MWAND algorithm is similar to the initial top-k heap of WAND.

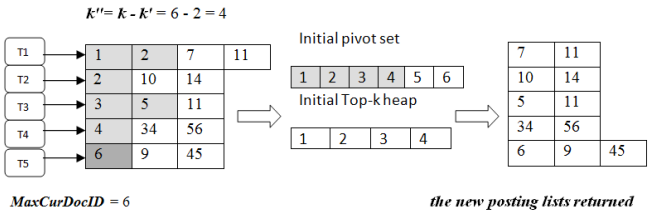

Fig. 5. Example of "Filling function" and the results returned.

In Fig. 5 filling function of MWAND sort lists by their current docIDs, and select all documents with docID ≤ MaxCurDocID. In this scenario initial k=6 and k'=2 then k"= k-k'=4.

- Initial pivot is {1, 2, 3, 4, 5, 6} ; select k first element, top-k heap is {1, 2, 3, 4};
- New posting lists returned are: It_1{7,11}; It_2{10, 14}; It_3{5, 11}; It_4{34, 56}; It_5{6, 9, 45}.

In cases where:

a) k" > n (n=Card(initial pivot set));we execute the filling function again, updating the value of k"=n - k", after we combine the lists.

b) k' equals to k; the execution stop in the first level and we proceed directly to the processing of the next query.

VI. Results

In this section, we will expose our experimental results. But, first we explain why we choose to compare MWAND only with WAND and not with other methods. It is logic to compare MWAND with those works, but we assume that MWAND and WAND work with the same logic. Works like BMW [8][50][51][52][58] or others in our related works, have extended WAND with supplementary structures; for example BMW propose to consult well defined blocks, besides it manipulates the inverted lists. Of course times spent by such proposition will be better than those consumed by WAND. The technique of augmenting solutions by supplementary structure is a common way in research. For example, in order to compute the Top-k in databases, besides it manipulates the same lists as TA (the Threshold Algorithm) of Fagin [34], BPA (the Best Positions Algorithm) of [69] proposes to use a supplementary list which saves the best positions of an object. Before the score of an object is computed, [69] consults the positions of an object. If it has good positions so it is in the Top-k. By this manner BPA will consume at the worst case the same time as TA. We can see the same remark between the algorithms TPUT [35] and HT (the Hybrid Threshold) [70]. This later defines more structures for computing the Top-k, despite it uses the same logic as TPUT. In our case, we want to show that with the same structure we can give a better solution. We do not use any extra-structure for our proposition.

A. Experimental Setup

Datasets. We have tested the WAND and the MWAND algorithms using Reuters collection-21578.

Query sets. We use a collection of 62350 queries which contains queries of lengths that vary between 3 and 10 terms and a list of 677 stop words (we remove stop words in the queries and in the inverted index).We returned top-20 results for each query.

Index structure. We compared the WAND and the MWAND approach on a tree index using DAAT query processing. Note that WAND and our approach by tree index return exactly the same results that WAND and our approach with classical index. In all our runs, we load index completely into the main memory. All codes are available by contacting the authors.

The experimentations were conducted on a machine with Intel (R) Core(TM) i7-4500U CPU @1,8Ghz with 8 GB of RAM. The programs were written in Java in Netbeans 7.0.

B. Results

In this section we compare our algorithm MWAND (using DAAT) and WAND on conjunctives and disjunctives queries. Then we measure the performance by six criteria: precision -- recall -- f-measure --Exact relevance degree of results --almost total relevance degree of results -- fidelity -- exact recall -- number of operations -- running time (in ms).

In the field of information science and systems of IR, the area of results quality is very important, that is why a set of tests must be applied and the question that may be asked is: should new measures be integrated in this domain to define the performance of systems?

1. Performance Evaluation

As explained Buttcher in [71] and Lewis in [72], the definition of precision and recall metrics is obtained by dividing the documents returned by SRI into two main categories: relevant documents and irrelevant documents, as it is shown in Fig. 6.

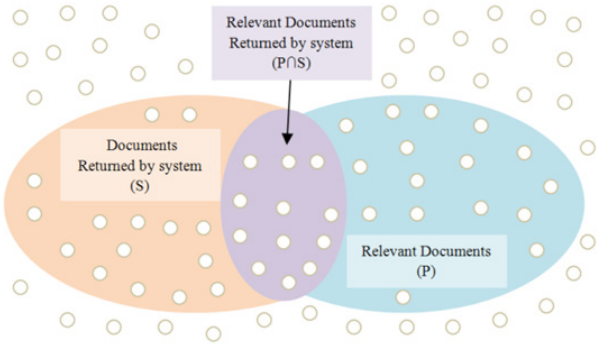

Fig. 6. Performance Evaluation.

At first, we compare WAND and MWAND algorithms on precision, recall and f-measure metrics:

$$\text{Precision} = \frac{|P \cap S|}{|S|} \quad (4)$$

$$\text{Recall} = \frac{|P \cap S|}{|P|} \quad (5)$$

$$\text{F-measure} = \frac{2*(precision*recall)}{precision+recall} \quad (6)$$

a) Comparison on the Precision

We have computed the precision of the results search. Table I gives the values of precision of 20 sets of results. In this experience, we can see that MWAND presents best values of precision. The average precision of: a) WAND is 0.56. b) MWAND is 0.72. The times when the precision is equal to 1 is explained by two facts. The fact one is about a non-open data, it is a closed data where there are new documents insertions, so it is possible to get such precision. The second fact is about the number of responses. When this one is so little it will be possible to get the unique solution. The times where the precision of MWAND is high mean that in the first round where the threshold is null, this algorithm inserts documents where everyone shares at least (Lq-1) terms with query q.

Fig. 7 shows the curve of the average of precision for 10 queries, while Fig. 8 presents the cumulated gain observed on Fig. 7. The accumulation can give more information. These figures illustrate how much MWAND returns better precision than WAND. This observation is in the continuation of what we have seen on Table I. In order to see about the precision if we increase the number of queries, we have executed 100 queries in the same conditions.

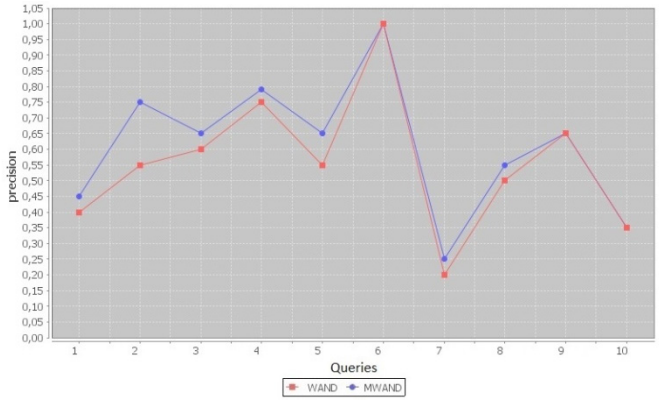

Fig. 7. Precision curve for 10 queries.

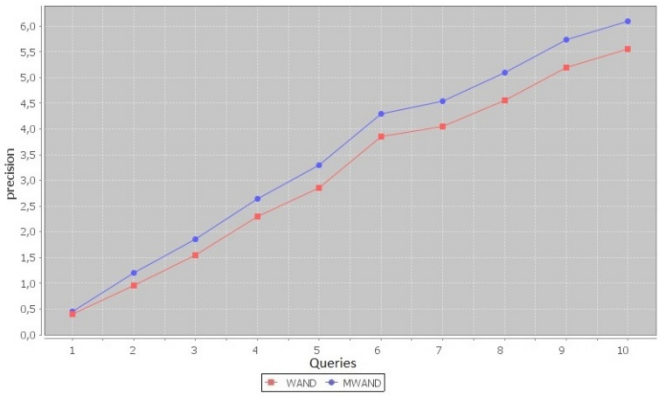

Fig. 8. Cumulated Precision curve for 10 queries.

Fig. 9 presents the curve of precision of these queries and Fig. 10 shows the cumulated precision gain. It is clear that responses in MWAND are more precise than those one returned by WAND. This result is due to the manner with what MWAND computes responses. In Fig. 11, we analyze the area of |P∩S| for results query of 5 terms. The initial value is 0.We test for each document (d) of top-k the existence of (d) in (p); when this condition is verified we increment the initial value. In this example, we see that all top-k documents returned by MWAND are relevant compared to WAND which loses some relevant documents.

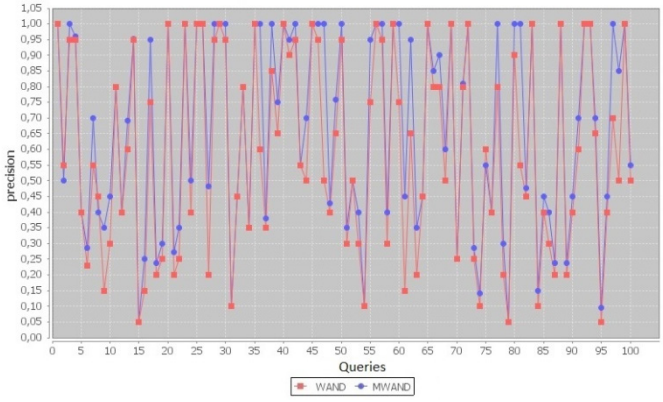

Fig. 9. Precision curve for 100 queries.

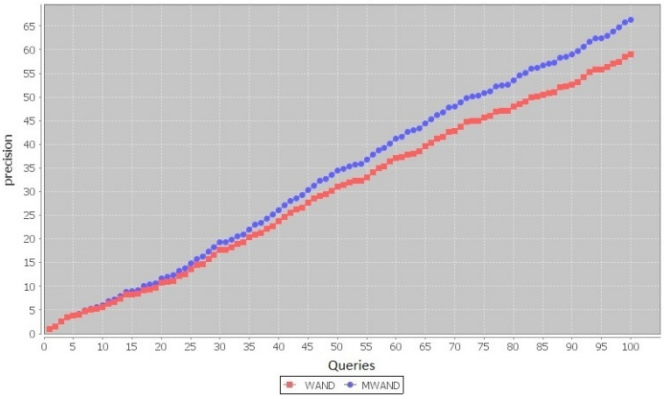

Fig. 10. Cumulated Precision curve for 100 queries.

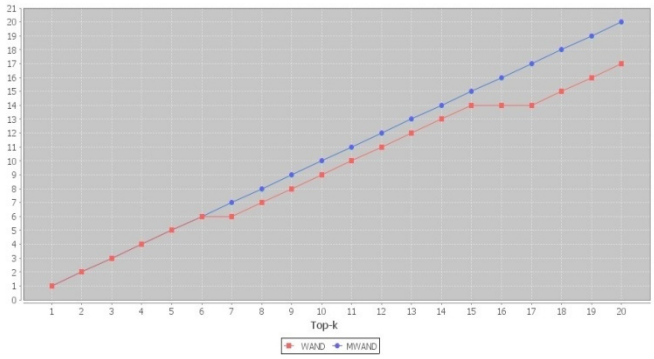

Fig. 11. |P∩S| curve for one query.

TABLE I. Comparison of WAND and MWAND Results in Terms of Precision

Queries	WAND	MWAND
1	0,7	1
2	0,3	0,3
3	0,35	1
4	0,45	1
5	0,4	0,45
6	1	1
7	0,35	0,38
8	0,45	0,8
9	0,8	1
10	0,45	0,476
11	0,01	0,15
12	1	1
13	0,5	0,5
14	0,1	0,15
15	1	1
16	0,55	0,65
17	0,85	1
18	0,05	0,5
19	1	1
20	1	1

b) Comparison of WAND and MWAND in Terms of Recall

In order to compare them on this measure, we have compared them according to the responses of the system. Thus, we have computed the recall for sets of results of 10 queries and 100 queries. Fig. 12 and Fig. 13 show clearly that MWAND has returned the best top-k documents, but WAND is not at the same level of quality of responses as in the precedent level. The missing of documents is the cause of the drop in the case of WAND.

c) Comparison According to the F-measure

In this experimentation, we compare these two methods according to the f-measure metric. As in the precedent experimentation, we have executed sets of 10 and 100 queries and we have calculated the f-measure of each top-k results. Results of this experimentation are depicted in Fig. 14 and Fig. 15; we found that MWAND has returned more relevant documents.

It is known that if the recall is high so the precision will be low. Here, if we plot the curves (precision-recall) of MWAND and WAND in the same plane, we will get MWANDS'curve higher than WAND's one.

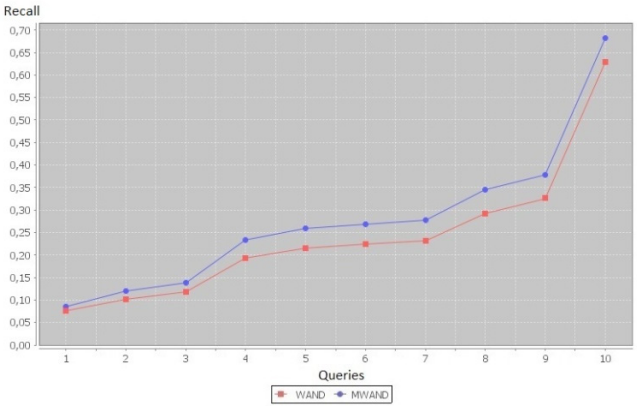

Fig. 12. Cumulated Recall curve for 10 queries.

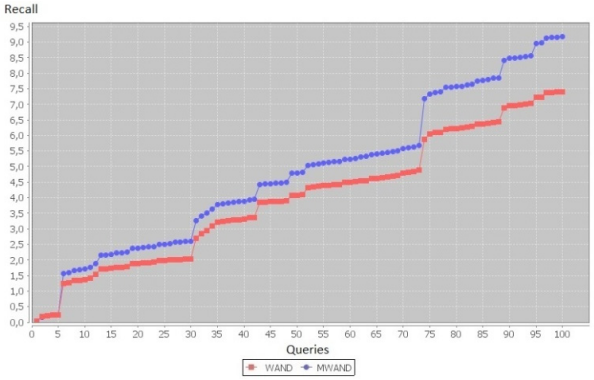

Fig. 13. Cumulated Recall curve for 100 queries.

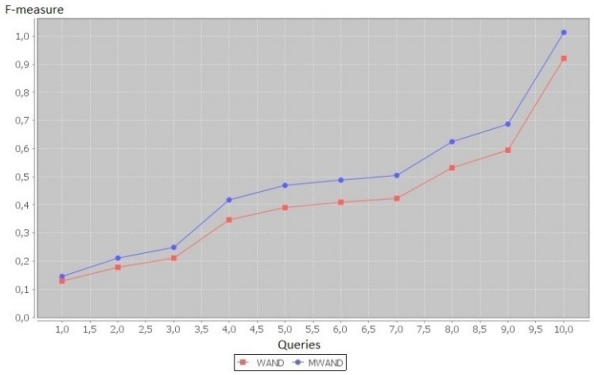

Fig. 14. Cumulated F-measure curve for 10 queries.

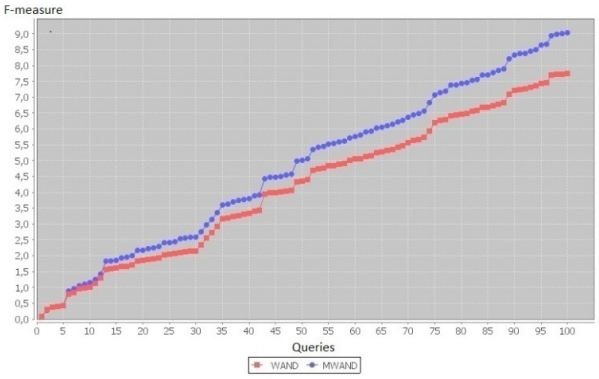

Fig. 15. Cumulated F-measure curve for 100 queries.

2. New Metrics

Before presenting our new metrics (for our knowledge, it is the first time that such definitions are given). We start with the definitions of relevance degree as Mechah explained in [73].

TABLE II. Different Definitions of Relevance For Document *d* and Query *Q*

Definition 1	Relevance of a Document	d ∩ q= ∅
Document d is relevant if it shares some terms with query q		
Definition 2	Total Relevance of a Document	d ∩ q=q
Document d is totally relevant if it contains all query terms		
Definition 3	Document Relevance Degree	d°(d) =|d ∩q |
Relevance degree is the number of terms shared between document d and query q		
Definition 4	Total Document Relevance Degree	td °(d) =|Q |
Relevance document degree is total if document contains all query terms		

The definitions cited in Table II give how much a document d is relevant for a query q. These definitions can give more information about the precision of the responses. Equations (4) and (5) given above are general, they cannot give a fine discrimination between algorithms. For this reason, we propose new metrics to find answers to the following questions:

- Are the returned documents so relevant to the queries?
- Are the total relevant documents positioned first?

In this context, we propose new definitions of results relevance degree and fidelity:

a) Exact Relevance Degree of Results

Definition 1. Exact Relevance Degree of Results: the exact relevance degree of result "r = {d_1, d_2…..d_l}" that contains a set of documents d, is noted Edr°(r). For a query q, Edr° (r) is the number of documents that contains |q| terms, divided by the total number of relevant documents P.

$$\text{For i in}\{1, 2\ldots\ldots l\}:$$

$$Xi = \begin{cases} 1 & \text{if } (td^\circ(d) = |q|) \\ 0 & \text{Otherwise} \end{cases}$$

$$|Edr^\circ(r)| = \sum_{i=1}^{l} X_i \quad (7)$$

$$Edr^\circ(r) = \frac{|Edr^\circ(r)|}{|p|} \quad (8)$$

A result is totally relevant if it contains K documents totally relevant.

b) Almost Total Relevance Degree of Results

Definition 2. Almost Total Relevance Degree of Results: the almost total relevance degree of result "r = {d1, d2…..dl}" that contains a set of documents d, is noted dr°(r). For a query q, dr°(r) is the number of documents that contains |q| terms or |q-1| terms, divided by the total number of relevant documents P.

$$\text{For i in } \{1, 2\ldots\ldots l\}:$$

$$Xi = \begin{cases} 1 & \text{if } (d^\circ(d) = |q| \text{ or } d^\circ(d) = |q-1|) \\ 0 & \text{Otherwise} \end{cases}$$

$$|dr^\circ(r)| = \sum_{i=1}^{l} X_i \quad (9)$$

$$dr^\circ(r) = \frac{|dr^\circ(r)|}{|p|} \quad (10)$$

A result is almost totally relevant if it contains k documents that share |q| or |q-1| terms.

Table III represents the study of comparison of exact relevance degree and almost total relevance degree of the first 20 (top-20) results.

Table III shows values of |Edr°(r)| and |dr°(r)| of 20 sets of results, every set contains 20 documents returned (top-20).We test results and we note that the results of MWAND in queries (1, 6, 9, 15, 17, 19, 20) are almost totally relevant, and results in queries (15, 19, 20) are totally relevant. Results of WAND in queries (15, 19) are almost totally relevant. We can see in Table IV the percentage of totally relevant results and almost totally relevant results of WAND compared to the MWAND.

TABLE III. COMPARISON OF SIMPLE AND EXACT RELEVANCE DEGREE

q	|Edr°(r)|		|dr°(r)|	
	WAND	MWAND	WAND	MWAND
1	1	7	4	20
2	0	0	0	0
3	0	1	0	17
4	1	1	1	15
5	1	1	2	3
6	1	1	5	20
7	1	2	1	2
8	1	1	2	2
9	0	0	4	20
10	1	2	1	2
11	1	1	1	2
12	0	0	0	3
13	4	4	4	4
14	2	2	2	3
15	9	20	20	20
16	0	1	0	3
17	4	6	5	20
18	1	1	1	1
19	9	20	20	20
20	8	20	16	20

TABLE IV. COMPARISON OF WAND AND MWAND IN TERMS OF RELEVANCE

	Percentage of total relevance		Percentage of almost total relevance	
WAND	0	0%	2/20	10%
MWAND	3/20	15%	7/20	35%

Note that, in all cases $|Edr^\circ(r)|_{MWAND} \geq |Edr^\circ(r)|_{WAND}$ and if |Edr°(r)|=K automatically |dr°(r)|=K; we can see this case in results queries evaluation of (15, 19, 20).There are two main observations: Firstly, in some cases relevant documents that contain all the query words are ignored by WAND. Secondly all the MWAND results are better than the WAND results.

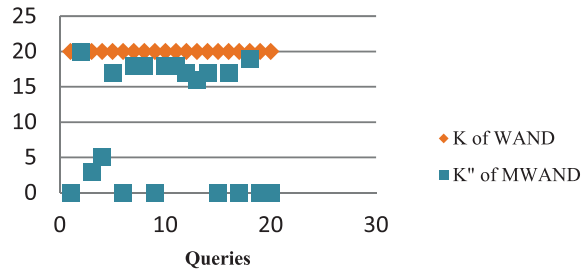

Fig. 16. K and K" variations.

As we explained in section V, K represents the number of documents to be returned, k' is the parameter returned by the first level and k"=k-k'. Fig. 16 shows the variation of K" during query processing. For WAND the value of k is static but for MWAND it varies between 0 and K (K=20). In MWAND we can find two cases: a) The best case is when k"=0, here MWAND stops processing in the first level and results are almost totally relevant or totally relevant. b) The worst case is when k"=20; in this case totally relevant documents do not appear in results set.

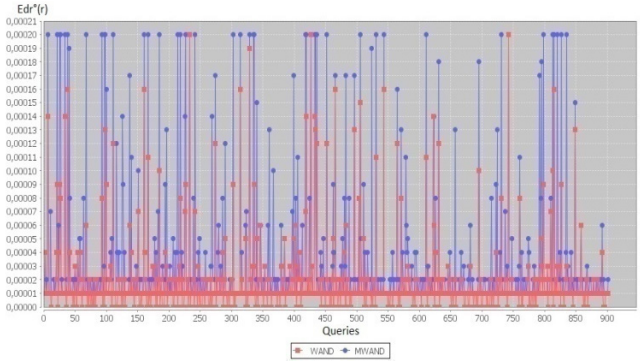

Fig. 17. Exact relevance degree of WAND and MWAND.

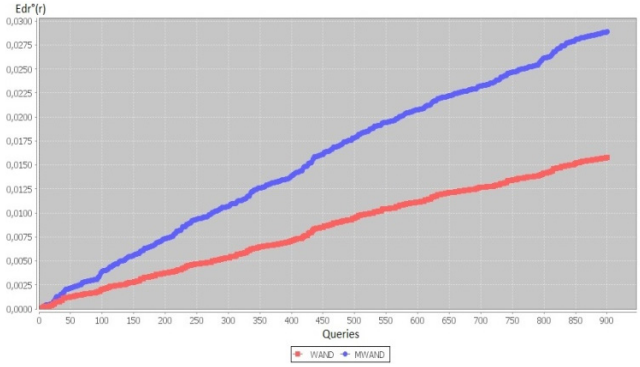

Fig. 18. Cumulated Exact relevance degree of WAND and MWAND.

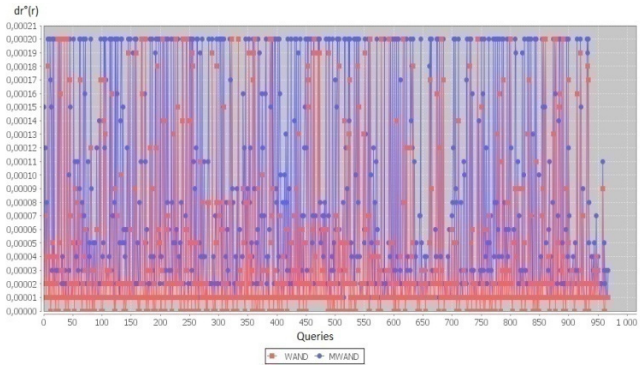

Fig. 19. Almost total relevance degree of WAND and MWAND.

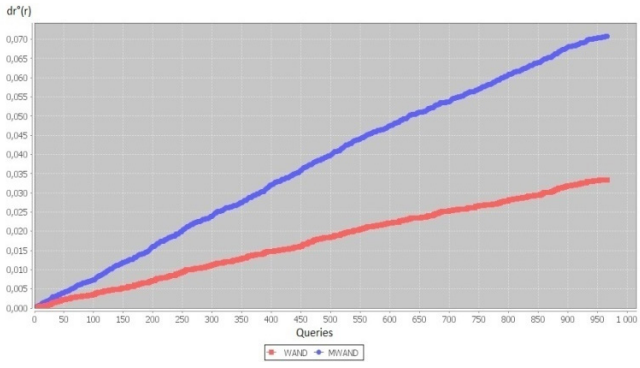

Fig. 20. Cumulated Almost total relevance degree of WAND and MWAND.

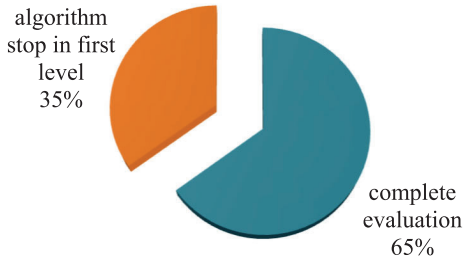

Fig. 21. Percentage of queries that stop processing in first level, of MWAND algorithm.

In Fig. 17 and Fig. 19 we represent exact relevance degree and almost total relevance degree of about 950 set of results and we see that MWAND results are more relevant than those of WAND. To clarify the results we show the Cumulated exact relevance degree and Cumulated almost total relevance degree in Fig. 18 and Fig. 20. We conclude that we can get more relevant documents using MWAND algorithm. We found that MWAND shares more documents with the real relevant documents. Fig. 21 shows that, in some cases MWAND can terminate earlier, where the value of k' returned in first level equals to K (see in Table III: queries (1, 6, 9, 15, 17, 19, 20) where d°(r)|=K=20)), the algorithm stops processing in first level and proceeds directly to the processing of the next query.

c) Exact Recall

The classical recall measures the ability of the system to retrieve all relevant documents responding to a query. We propose a new metric called Exact recall to highlight all set of results that contain total relevant documents.

Definition 3. Exact recall: This new measure noted ***Nrecall*** is defined as follows:

For query q of $|q|$ terms

$|q_i|$:The number of documents sharing i terms with the query q.

P: The number of real relevant documents.

For i{ 1, 2……|q| }:

$$\alpha_i = \frac{i}{|q|}$$

$$\text{Nrec} = \alpha_i \frac{|qi|}{|p|} + \alpha_{i-1} \frac{|qi-1|}{|p|} + \alpha_{i-2} \frac{|qi-2|}{|p|} \ldots\ldots + \alpha_1 \frac{|q1|}{|p|} \quad (11)$$

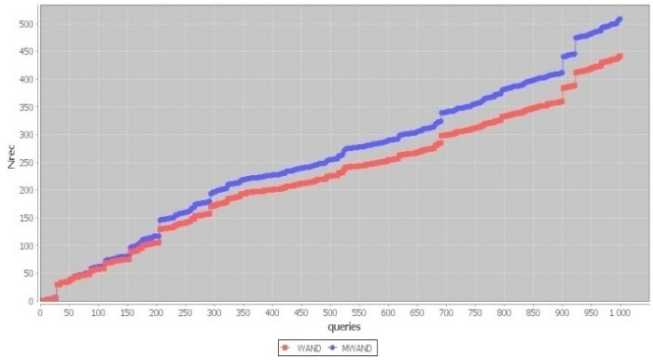

Fig. 22. Cumulated exact recall of 1000 queries.

In this stage of comparison, we executed a set of 1000 queries. So we tested their exact recall to evaluate the performance of algorithms, as shown on the Fig. 22. MWAND has presented better cumulated gain than WAND. This is due essentially to the manner of fitting responses in Top-k lists, where documents which share more terms with the query are inserted. For this we say that the strategy of MWAND is more efficient than that of WAND.

d) Fidelity

Definition 4. Fidelity: In this experience, we send a query q and we collect a set of documents, we also collect the positions of returned documents and we compared them with the positions of real relevant documents.

$$\text{Fid} = \frac{Doc_{fid}}{|S|} \quad (12)$$

Doc_{fid}: The number of documents returned by the system, having the same ranking position, as the relevant documents.

S: The number of documents returned by the system.

In this part we focus on ranking position. Our aim is to maximize fidelity value. We have executed sets of 500 and 6400 queries as shown in Fig. 23 and Fig. 24. The results of these experiences show the big performance of MWAND.

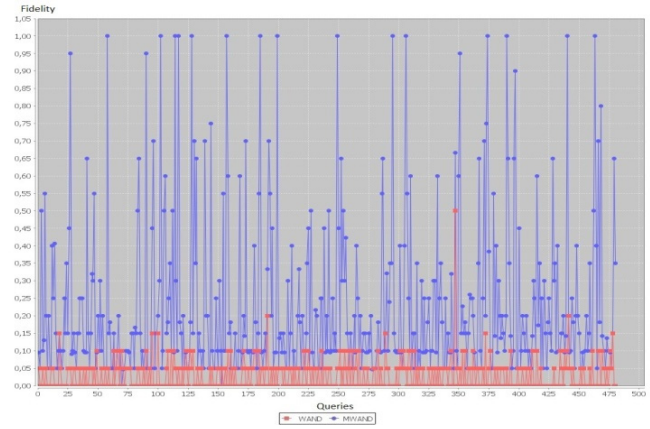

Fig. 23. Fidelity of 500 queries.

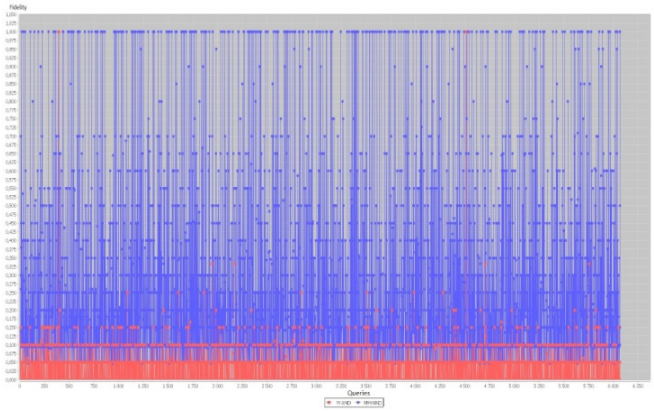

Fig. 24. Fidelity of 6400 queries.

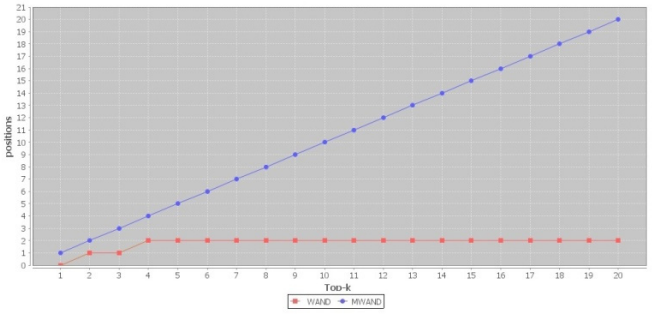

Fig. 25. Comparison of ranking Positions of WAND and MWAND.

We test if WAND and MWAND algorithms keep the same ranking position of the returned documents by comparing them with the ranking position of real relevant documents. In Fig. 25 we take an example and we find that MWAND keeps the same ranking for all documents and WAND keeps the same ranking for the second and fourth documents in top-k.

3. Comparison on Number of Operations

The goal of the first and second level in MWAND is to minimize the number of operations (NO) during query processing. Results of Fig. 26 show the variation of NO for a set of 20 queries. We observe that the MWAND processing is less expensive than WAND. We have also executed 1680 queries; we calculated the percentage of *NO* of WAND and MWAND. The results obtained show that MWAND percentage is 42.03% less than WAND percentage. That is because in MWAND, after executing the first and second level, the size of posting lists are smaller.

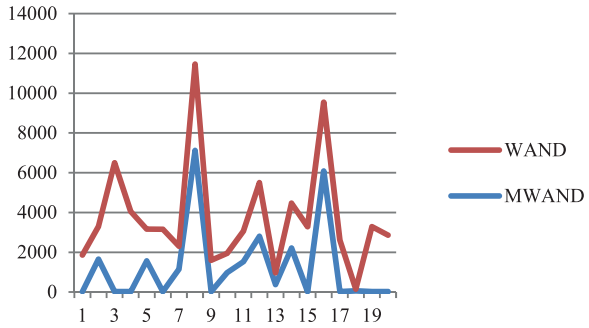

Fig. 26. Number of operations of WAND and MWAND.

4. Comparison on Time Processing

TABLE V. Comparison of Time Processing in (ms)

QUERIES	WAND	MWAND
1	234	209
10	561	499
20	1170	911
50	1966	1730
100	4070	4026
500	40131	19925
1000	54242	36438
6400	272551	271636
20052	759723	749879
39977	1775253	1768014

Table V shows the query processing time with different sets of queries. MWAND algorithm is faster than WAND algorithm. Especially, the performance of MWAND drops rapidly when: a) It stops in the first level. b) It selects a set of initial pivot terms in the second level, avoiding the approximate scoring.

VII. Conclusion and Future Works

Information retrieval is the science of searching information in a system; it calls a high number of techniques for satisfying users. This information can be a paper, a book, a piece of news, a photo or a state like in sentiment analysis or other thing, it depends on what we intend about the term “information”. The relevance is the greatest challenge, by which search engines are called to ensure for convincing users. In the context of retrieving documents, many techniques can be applied. The most important ones are term-at-a-time TAAT and document-at-a-time DAAT.

In this paper, we have presented techniques and structures that are used in retrieving documents in information systems. We have presented the strategies TAAT and DAAT, and we have described all what is known in this field. We have presented the structure of inverted lists and the major bottleneck of them. We have also presented their principal organizations in order to rapidly compute documents scores. The process of early termination is also presented. This process is applied for computing the exact solution without traversing the large inverted lists, as we have presented. In order to give new solution which is more effective and more efficient, we have presented our approach; we call it MWAND. It has the ability to avoid the unnecessary steps in WAND. We insert the best solution which shares the maximal number of terms with queries. By this way, we consume less time and we insert better documents. This one can give a better quality of responses.

We have compared WAND and MWAND according to general metrics, and in order to show how our work is efficient and effective; we have defined new metrics. For the best of our knowledge, it is the first time that such definitions are given. These metrics have the ability to analyze and show the hidden quality of responses. According to these metrics and by intensive experimentations, we have shown that MWAND can give better solutions in terms of quality and in execution time.

It is important to see that we have compared our work only to WAND as we do not use any other extra structure like BMW or others.

As future works, we want to do more comparisons with others works likes BMW. We will also insert the problem of intersection of ordered lists as we work intensively on it.

References

[1] J. Zobel, A. Moffat, "Inverted Files for Text Search Engines. ACM Computing Surveys," vol. 38, no. 2, article 6, July, 2006.

[2] J. Zobel, A. Moffat, R.S. Davis, "An Efficient Indexing Technique for Full-Text Database Systems," Proceedings of the 18th VLDB Conference Vancouver, British Columbia, Canada, 1992.

[3] C. Dimopoulos, S. Nepomnyachiy, T.Suel, "Optimizing Top-k Document Retrieval Strategies for Block-Max Indexes," WSDM'13, February 4–8, Rome, Italy, 2012.

[4] M. Persin, " Document filtering for fast ranking," In W. Croft and C. Van Rijsbergen, editors, Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval, Dublin, Ireland, 1994, pp 339-348.

[5] M. Persin, J. Zobel, R. Sacks-Davis, "Filtered document retrieval with frequency-sorted indexes," Journal of the American Society for Information Science, vol. 47, no. 10, pp 749-764, 1996.

[6] V. Anh, A. Moffat, "Index Compression using Fixed Binary Codewords," Proceedings of the Fifteenth Australasian Database Conference (ADC2004), Dunedin, New Zealand, 2004.

[7] A. Witten, A. Moffat, T. Bell, "Managing Gigabytes: Compressing and Indexing Documents and Images," Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, second edition, 1999.

[8] S. Ding, T. Suel, "Faster Top-k Document Retrieval Using Block-Max Indexes,"SIGIR'11, 2011.

[9] H. Turtle, J. Flood, "Query evaluation: Strategies and optimizations," Information Processing and Management, vol. 31, no. 6, pp. 831–850, 1995.

[10] V. Anh, A. Moffat, "Pruned query evaluation using pre-computed impacts," In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006, pp. 372–379.

[11] T. Strohman, W. Bruce Croft, "Efficient Document Retrieval in Main Memory," SIGIR'07, July 23–27, Amsterdam, The Netherlands, 2007.

[12] H. Wu, H. Fang, "An Incremental Approach to Efficient Pseudo-Relevance Feedback," SIGIR'13, Dublin, Ireland, July 28–August 1, 2013.

[13] C. Buckley, A.F. Lewit, "Optimization of Inverted Searches," SIGIR '85 Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval, 1985, pp. 97-110.

[14] R. Fagin, A. Lotem, M. Naor, "Optimal aggregation algorithms for middleware," In Symposium on Principles of Database Systems, 2001.

[15] O. Rojas, V. Gil-Costa, M. Marin, "Distributing efficiently the Block-Max WAND algorithm," In proceedings of the conference on computational Science, 2005.

[16] A. Moffat, J. Zobel, "Self-indexing inverted files for fast text retrieval," ACM Transactions on Information Systems, vol. 14, no. 4, October 1996, pp. 349 –379.

[17] R. Baeza-Yates, B. Ribeiro-neto, D. Mills, O. AmsterdamBonn, "Modern Information Retrieval," Addison Wesley, First edition, 1999.

[18] C.D. Manning, P. Raghavan, H. Schütze, "An Introduction to Information Retrieval," Cambridge University Press Cambridge, England, 2009.

[19] O. Rojas, V. Gil-Costa, M. Marin. "Efficient Parallel Block-Max WAND Algorithm," Euro-Par 2013 Parallel Processing - 19th International Conference, Aachen, Germany, 2013, pp. 26-30. Proceedings. Lecture Notes in Computer Science 8097, Springer 2013.

[20] A.Z. Border, M. Herscovici, A. Soffer, J. Zien, "Efficient query evaluation using a two-level retrieval process,"CKIM'3, 2003.

[21] E. Robertson, S. Walker, M. Beaulieu "Okapi at TREC–7: automatic ad hoc, filtering, VLC and filtering tracks," In Proceedings of the Seventh Text REtrieval Conference (TREC-7), NIST Special Publication 500-242, July 1999, pp. 253–264.

[22] V. Anh, A. Moffat, "Simplified Similarity Scoring Using Term Ranks," SIGIR'05, August 15–19, Salvador, Brazil, 2005.

[23] F. Scholer, H. Williams, J. Yiannis, J. Zobel," Compression of inverted indexes for fast query evaluation," In Procedure of the 25th Annual SIGIR, Tampere, Finland, pp. 222-229, 2002.

[24] P. Elias, "Universal codeword sets and representations of the integers," IEEE transactions on Information Theory, vol. 21, no. 2, pp. 194-203, 1975.

[25] S.W. Golomb, "Run-length encoding," IEEE Transactions on Information Theory, vol. 12, no. 3 ,pp. 399–401, 1966.

[26] H.E. Williams, J. Zobel, "Compressing integers for fast file access," Comput. J, vol. 42, no. 3, pp. 193–201, 1999.

[27] V.N. Anh, A. Moffat, "Inverted index compression using word-aligned binary codes," Inf. Retr, vol. 8, no. 1, pp. 151–166, 2005.

[28] A. Moffat, L. Stuiver, "Binary interpolative coding for effective index compression," Inf. Retr, vol. 3, no. 1, pp. 25–47, 2000.

[29] M. Zukowski, S. H´eman, N. Nes, P.A. Boncz, "Super-scalar ram-cpu cache compression," in: ICDE, pp. 59, 2006.

[30] H. Yan, S. Ding, T. Suel, "Inverted index compression and query processing with optimized document ordering," in: WWW, pp.401–410, 2009.

[31] H. Yan, S. Ding, T. Suel, "Compressing term positions in web indexes," in: SIGIR, pp. 147–154, 2009.

[32] A. Trotman,"Compressing Inverted Files. Information Retrieval," 6, 5–19. Kluwer Academic Publishers. Manufactured in The Netherlands, 2003.

[33] R. Fagin, "Combining fuzzy information: an overview," SIGMOD Record, 31, 2002.

[34] H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum. "IO-Top-K: Index-accessoptimized top-k query processing," In Proceedings of the 32th International Conference on Very Large Data Bases, 2006.

[35] P. Cao, Z. Wang, "Efficient Top-k Query Calculation in Distributed Networks.In Proceedings of the 23rd PODC. 2004.

[36] HW. Wong, D. Lee, "Implementations of partial document ranking using inverted files," Information Processing and Management, vol. 29, no. 5, pp. 647–669, 1993.

[37] R. Blanco, A. Barreiro, "Probabilistic static pruning of inverted files," ACM Trans. Inf. Syst, vol. 28, no. 1, 2010.

[38] A. Soffer, D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y.S. Maarek. "Static index pruning for information retrieval systems," in: SIGIR, 2001, pp. 43–50.

[39] A. Ntoulas, J. Cho, "Pruning policies for two-tiered inverted index with correctness guarantee," in: SIGIR, pp. 191–198, 2007.

[40] C. Buckley, A.F. Lewit, " Optimization of inverted vector searche,"In Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1985.

[41] T. Strohman, H. Turtle, W.B. Croft, " Optimization strategies for complex queries," In Proc. SIGIR, pp. 219–225, 2005.

[42] M. Fontoura, V. Josifovski, J. Liu, S. Venkatesan, X. Zhu, J. Zien, "Evaluation strategies for top-k queries over memory-resident inverted

indexes," Proc. VLDB Endowment, vol. 4, no. pp. 12:1213–1224,2011.
[43] M. Kaszkiel and J. Zobel. "Term-ordered query evaluation versus document-ordered query evaluation for large document databases," In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Austrailia, August 1998, pp 343–344.
[44] F. Zhang, S. Shi, H. Yan, J. Wen, "Revisiting Globally Sorted Indexes for Efficient DocumentRetrieval," WSDM'10, February 4-6, New York City, New York, USA, 2010.
[45] J. Mackenzie, F. Scholer, J. Shane, "Early Termination Heuristics for Score-at-a-Time Index Traversal," ADCS 2017, December 7–8, 2017, Brisbane, QLD, Australia.
[46] J. Lin, A. Trotman, "Anytime Ranking for Impact-Ordered Indexes," ICTIR'15, September 27–30, Northampton, MA, USA.
[47] A. Elbagoury, M. Crane, J. Lin, "Rank-at-a-Time Query Processing," ICTIR '16, September 12 - 16, Newark, DE, USA.
[48] H. He, A.K. Singh, "Graphs-at-a-time: Query Language and Access Methods for Graph Databases," SIGMOD'08, June 9–12, Vancouver, BC, Canada.
[49] A. Anagnostopoulos, A.Z. Broder, D. Carmel, "Sampling Search-Engine Results," WWW 2005, May 10-14, 2005, Chiba, Japan.
[50] N. Tonellotto, C. Macdonald, I. Ounis, "Efficient and Effective Retrieval using Selective Pruning," WSDM'13, February 4–8, 2013, Rome, Italy.
[51] N. Tonellotto, C. Macdonald, I. Ounis, "Efficient Dynamic Pruning with Proximity Support," LSDS-IR 2010, July 23, 2010, Geneva, Switzerland.
[52] N. Tonellotto, C. Macdonald, I. Ounis, "Effect of Different Docid Orderings on Dynamic Pruning Retrieval Strategies," SIGIR'11, July 24–28, 2011, Beijing, China.
[53] N. Asadi, J. Lin, "Effectiveness/Efficiency Tradeoffs for Candidate Generation in Multi-Stage Retrieval Architectures,"IGIR'13, July 28–August 1, 2013, Dublin, Ireland.
[54] J.S. Culpepper, A. Moffat, "Efficient set intersection for inverted indexing," TOIS, vol. 29, no. 1, 2010.
[55] N. Asadi, J. Lin, "Fast Candidate Generation for Real-Time Tweet Search with Bloom Filter Chains," ACM Transactions on Information Systems, vol. 31, no. 3, article 13, July 2013.
[56] M. Petri, J.S.Culpepper, A.Moffat, "Exploring the Magic of WAND," ADCS '13, December 05 - 06 2013, Brisbane, QLD, Australia.
[57] Yubin Kim1, Jamie Callan, J.Shane Culpepper, Alistair Moffat, "Does Selective Search Benefit from WAND Optimization," ECIR 2016, LNCS 9626, pp. 145–158.
[58] A. Mallia, G.O. aviano, E. Porciani, N. Tonello, R. Venturini, "Faster BlockMax WAND with Variable-sized Blocks," SIGIR'17, Shinjuku, Tokyo, Japan, 2017.
[59] E. Bortnikov, D. Carmel, G. Golan-Gueta, "Top-k Query Processing with Conditional Skips," WWW 2017 Companion, April 3–7, 2017, Perth, Australia.
[60] C.M. Daoud, E. Silva de Moura, D. Fernandes, A. Soares da Silva, C. Rossi, A. Carvalho, "Waves: a fast multi-tier top-k query processing algorithm," Information Retrieval Journal, vol.3, n. 20, march 2017.
[61] D. Shan, S. Ding, J. He, H. Yan, X. Li. "Optimized Top-K Processing with Global Page Scores on Block-Max Indexes," WSDM'12, February 8-12, + Seattle, Washington, USA, 2012.
[62] M. Petri, J.S. Culpepper, A. Moffat, "Exploring the Magic of WAND," ADCS '13, December 05 - 06, 2013, Brisbane, QLD, Australia.
[63] A. Broder, L. Garcia-Pueyo, V. Josifovski, "Scalable K-Means by Ranked Retrieva,"WSDM'14, February 24–28, 2014, New York, New York, USA, 2012.
[64] A. Kane, F.W. Tompa, "Split-Lists and Initial Thresholds for WAND-based Search," SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA, 2018.
[65] M. Crane, J.S. Culpepper, J. Lin, J. Mackenzie, A. Trotman, "A Comparison of Document-at-a-Time and Score-at-a-Time Query Evaluation," WSDM 2017, February 06 - 10, 2017, Cambridge, United Kingdom.
[66] J. Mackenzie, J.S. Culpepper, R. Blanco, M. Crane, C.L.A. Clarke, J. Lin, "Query Driven Algorithm Selection in Early Stage Retrieval,"WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA.
[67] L. Wang, J. Lin, D. Metzler, " A cascade ranking model for efficient ranked retrieval," In Proc. of SIGIR, 2011.
[68] D. Gusfield,"An increment-by-one approach to su x arrays and trees," Technical Report CSE-90-39 , UC Davis, Dept, Computer Science, 1990.
[69] R. Akbarinia, E. Pacitti, P. Valduriez. "Best position algorithms for top-k queries," In Proceedings of the 33rd international conference on Very large data bases, VLDB '07, 2007, pp. 495–506.
[70] H. Yu, H. Li, P. Wu, D. Agrawal, A. Abbadi, "Efficient Processing of Distributed Top-k Queries," In Proceedings of the 16th DEXA, 2005.
[71] S. Buttcher, Clarke, G.V. Cormack, "Information Retrieval: Implenting and Evaluating Search Engines," MIT Press, 2010, pp. 68.
[72] D. Lewis, Y. Yang , G. Rose, Li ,"RCVI/ A new benchmark collection for text categorization," Journal of Machine Research, 2004, pp.361-397.
[73] K. Mechach, L. Zekri, M.K. Abdi, "Collection and Selection Based Relevant Degrees Of Documents," In Journal of Digital Information Management (JDIM) , vol. 13, no. 2, pp. 110-119, 2015.

Zemani Imene Mansouria

Zemani Imene Mansouria is a PhD student at the department of computer sciences at Oran1 Ahmed Ben Bella University. She obtained her master in the field of operational research. Actually, she is interested in information retrieval, indexation and big data.

Zekri Lougmiri

Zekri Lougmiri is assistant professor at the department of computer sciences at Oran1 Ahmed Ben Bella University. He received his Phd at the same university. He works on multi-criteria optimization, information retrieval and Big data.

Senouci Mohamed

Senouci Mohamed is professor at the department of computer sciences at Oran1 Ahmed Ben Bella University. His current research area includes pattern recognition, deep learning and ad hoc networks.