Indexes for Document Retrieval with Relevance*

Wing-Kai Hon[1], Manish Patil[2], Rahul Shah[2],
Sharma V. Thankachan[2], and Jeffrey Scott Vitter[3]

[1] National Tsing Hua University, Taiwan. wkhon@cs.nthu.edu.tw
[2] Louisiana State University, USA. {mpatil,rahul,thanks}@csc.lsu.edu
[3] The University of Kansas, USA. jsv@ku.edu

Abstract. Document retrieval is a special type of pattern matching that is closely related to information retrieval and web searching. In this problem, the data consist of a collection of text documents, and given a query pattern P, we are required to report all the documents (not all the occurrences) in which this pattern occurs. In addition, the notion of *relevance* is commonly applied to rank all the documents that satisfy the query, and only those documents with the highest relevance are returned. Such a concept of relevance has been central in the effectiveness and usability of present day search engines like Google, Bing, Yahoo, or Ask. When relevance is considered, the query has an additional input parameter k, and the task is to report only the k documents with the highest relevance to P, instead of finding all the documents that contains P. For example, one such relevance function could be the frequency of the query pattern in the document. In the information retrieval literature, this task is best achieved by using inverted indexes. However, if the query consists of an arbitrary string—which can be a partial word, multiword phrase, or more generally any sequence of characters—we cannot take advantages of the word boundaries and we need a different approach. This leads to one of the active research topics in string matching and text indexing community in recent years, and various aspects of the problem have been studied, such as space-time tradeoffs, practical solutions, multipattern queries, and I/O-efficiency. In this article, we review some of the initial frameworks for designing such indexes and also summarize the developments in this area.

1 Introduction

Query processing forms a central aspect of databases which in turn is supported by data structures that are commonly referred to as *indexes*. In databases, the notion of queries is semantically well-defined; hence a tuple (or a record) either qualifies or does not qualify for the query,

* This work is supported in part by Taiwan NSC Grant 99-2221-E-007-123 (W. Hon) and US NSF Grant CCF–1017623 (R. Shah and J. S. Vitter) and CCF–1218904 (R. Shah).

and a database operation will return exactly all those tuples that satisfy the query conditions. In contrast, information retrieval takes a somewhat fuzzy approach on query processing. The data are often unstructured and the notions of precision, recall, and relevance add their flavors to which tuples are returned. Often the criteria for a tuple to satisfy the query is not just a binary decision. The notion of *relevance-ranking* is central to information retrieval where the output is ranked by relevance score—which is an indicator of how strongly the tuple (or a web document, in case of search engines) matches the query. In recent times, various extensions to the standard relational database model have been proposed to cope with an increasing need to integrate databases and information retrieval. Top-k query processing is one such line of research, which adds the notion of relevance to database query processing.

Formally, a top-k query comes with a parameter k. Amongst all tuples that satisfy the query, they are ranked by their relevance scores, and only the k most relevant tuples are reported. In document retrieval and duplicate elimination (as a part of the projection operation) in databases, we get multiple occurrences of the same tuple (or key) satisfying the query and relevance depends on the contribution of each such tuple. In this case, only one tuple (out of the multiple occurrences) is to be reported with composite score. A simple example of such a score function is the frequency —which is number of times a particular attribute occurs in the query result. In terms of web-search this is known as *term-frequency*, which is the number of times the query term occurs in a given document. There can be even more complex statistical scoring functions, for instance when one considers OLAP queries (with slice-and-dice type ranges).

In terms of document retrieval, we are given a set $\mathcal{D}=\{d_1, d_2, d_3, ..., d_D\}$ of D string documents of total length n. We build an index on this collection. Then pattern P (of length p) comes as an query, and we are required to output the list of all *ndoc* documents in which the pattern P appears (not all *occ* occurrences). This is called the *document listing* problem and was introduced by Matias et al. [28]. Muthukrishnan [29] gave the first optimal $O(p+ndoc)$ query time solution in linear space, i.e., $O(n)$ words. Since then, this has been an active research area [38, 41, 16] with focus on making the index space-efficient. In *top-k document retrieval*, there is a relevance score involved in addition to the uniqueness condition. Let $S(P, d_i)$ be the set of occurrences of pattern P in the document d_i. The relevance score of P with respect to d_i is a function $w(P, d_i)$ that depends only on the set $S(P, d_i)$. Now, as a query result, we are required to report

only the top-k highest scoring documents. The formal definition is given below:

Problem 1 (Top-k document retrieval problem) *Let $w(P, d)$ be the score function capturing the relevance of a pattern P with respect to a document d. Given a document collection $\mathcal{D}= \{d_1, d_2, ..., d_D\}$ of D documents, build an index answering the following query: given input P and k, find k documents d with the highest $w(P, d)$ values in sorted (or unsorted) order.*

This problem was introduced in [18], where they proposed an $O(n \log D)$ words index with query time $O(p + k + \log D \log \log D)$ (works only for document-frequency as the score function). The recent flurry of activities [6, 11, 15, 20, 24, 26, 32, 37, 19, 25, 22, 27, 40, 42] came with Hon et al.'s work [23]. In this survey article, we review the various aspects of top-k document retrieval as listed below:

- We begin by describing the linear space and optimal time (internal memory) framework based on the work of Hon et al. [23] and of Navarro and Nekrich [31].
- In Section 3, we focus on the I/O model [4] solution for top-k document retrieval by Shah et al. [39] that occupies almost-linear $O(n \log^* n)$ space and can answer queries in $O(p/B + \log_B n + k/B)$ I/Os.
- In Section 4 we briefly explain the first succinct index that was proposed by Hon et al. [23] occupying roughly twice the size of text with $O(p + k \log^{O(1)} n)$ query time and also review the later developments in this line of work.
- We also briefly discuss variants of document retrieval problem in Section 5 such as multipattern queries, queries with forbidden pattern, parameterized top-k queries.
- Finally, we conclude in Section 6 by listing some of the interesting open problems in this research area.

2 Linear Space Framework

This section briefly explains the linear space framework for top-k document retrieval based on the work of Hon, Shah and Vitter [23] and Navarro and Nekrich [31]. The generalized suffix tree (GST) of a document collection $\mathcal{D}= \{d_1, d_2, d_3, \ldots, d_D\}$ is the combined compact trie (a.k.a. Patricia trie) of all the non-empty suffixes of all the documents. We use n to denote the total length of all the documents, which is also the number of

the leaves in GST. For each node u in GST, consider the path from the root node to u. Let $\mathit{depth}(u)$ be the number of nodes on the path, and $\mathit{prefix}(u)$ be the string obtained by concatenating all the edge labels of the path. For a pattern P that appears in at least one document, the *locus* of P, denoted as u_P, is the node closest to the root satisfying that P is a prefix of $\mathit{prefix}(u_P)$. By numbering all the nodes in GST in the pre-order traversal manner, the part of GST relevant to P (i.e., the subtree rooted at u_P) can be represented as a range.

Nodes are marked with document-ids. A leaf node ℓ is marked with a document $d \in \mathcal{D}$ if the suffix represented by ℓ belongs to d. An internal node u is marked with d if it is the lowest common ancestor of two leaves marked with d. Notice that a node can be marked with multiple documents. For each node u and each of its marked documents d, define a *link* to be a quadruple $(\mathit{origin}, \mathit{target}, \mathit{doc}, \mathit{score})$, where $\mathit{origin} = u$, *target* is the lowest proper ancestor[4] of u marked with d, $\mathit{doc} = d$, and $\mathit{score} = w\big(\mathit{prefix}(u), d\big)$. Two crucial properties of the links identified in [23] are listed below.

- For each document d that contains a pattern P, there is a unique link whose *origin* is in the subtree of u_P and whose *target* is a proper ancestor of u_P. The *score* of the link is exactly the score of d with respect to P.
- The total number of links is bounded by $O(n)$.

We say that a link is *stabbed* by node u if it is originated in the subtree of u and targets a proper ancestor of u. Therefore, top-k document retrieval can be viewed as the problem of indexing the $O(n)$ links described above to efficiently report the k highest scored links stabbed by any given node u_P. By mapping each link $L_i = (o_i, t_i, \mathit{doc}, \mathit{score}_i)$ to a 3d point $(x_i, y_i, z_i) = (o_i, \mathit{depth}(t_i), \mathit{score}_i)$, the above problem can be reduced to the following range searching query: report k points with the highest z coordinate among those points with $x_i \in [u_P, u'_P]$ and $y_i < \mathit{depth}(u_P)$, which is a 4-constraint query. Here u'_P represents the pre-order rank of the right most leaf in the subtree of u_P.

While general 4-sided orthogonal range searching is proved hard [7], the main idea is to make use of the special property that the reduce subproblem can only have p distinct values, hence it can be decomposed into p 3-constrained queries (which can be solved optimally). Thus a linear space index with near-optimal $O(p + k \log k)$ is achieved by Hon et

[4] Define a dummy node as the parent of the root node, marked with all the documents.

al. [23]. This query time is improved to optimal $O(p+k)$ by Navarro and Nekrich [31].

Theorem 1 *There exists a linear space index of $O(n)$-word space for answering top-k document retrieval queries in optimal $O(p+k)$ time.*

Nekrich [31] showed that the index space can be reduced to $O(n(\log\sigma+\log D+\log\log n))$ bits, if the requirement is to retrieve only the top-k documents without their associated scores. With *term-frequency* as the score they achieved the index that is further compressed occupying $O(n(\log\sigma+\log D))$ bits. Hon et al. [19] proposed an alternative approach to directly compress the index to achieve an $n(1+o(1))(\log\sigma+2\log D)$ bits index with $O(p+k\log\log n+poly\log\log n)$ query time.

3 External-Memory Framework

With the advent of enterprise search, deep desktop search, and email search technologies, the indexes that reside on disks (external memory) are more and more important. Unfortunately, the (linear space) approach described in the previous section cannot lead to an optimal external memory solution as it inevitably adds an extra $O(p)$ additive factor in query time. Therefore, we need to explore some other properties that can potentially simplify the problem. In this section, we briefly describe the I/O-efficient framework by Shah et al. [39]. They showed how to decompose the 4-constrained query (as described in the previous section) into at most $\log(n/B)$ (instead of p) 3-constrained queries, by exploring the fact that, out of four constraints in the given query, two of them always correspond to a tree range. Here B denotes the disk block size.

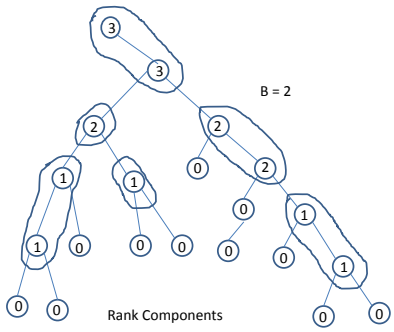

Fig. 1. Rank Components

Here we solve a threshold variant of the problem (i.e., among all those links stabbed by u_P, retrieve those with weight at least a given threshold τ). Note that, both threshold and top-k variants are equivalent due to the existence of a linear-space structure to compute threshold τ given (u_P,k) in $O(1)$ time such that the number of number of outputs reported

by threshold variant of the problem is between k and $k + O(k + \log n)$. It is known that no linear-space external memory structure can answer the (even the simpler) 1d top-k range reporting query in $O(\log^{O(1)} n + k/B)$ I/Os if the output order must be ensured [2]. We thus turn our attention to solving the unordered variant of the top-k document retrieval problem.

We start with some definitions: Let $size(u)$ denote the number of leaves in the subtree of u. We define the *rank* of u,

$$rank(u) = \left\lfloor \log\lceil \frac{size(u)}{B} \rceil \right\rfloor$$

Note that $rank(\cdot) \in [0, \lfloor \log\lceil \frac{n}{B} \rceil \rfloor]$ and nodes with the same rank will form a contiguous subtree, and we call each subtree a *component* (see Figure 1). The *rank* of a component is defined as the rank of nodes within it.

We classify the links into the following three types based on the *rank* of its target with respect to the *rank* of query node u_P: *low-ranked links*: links with $rank(target) < rank(u_P)$, *high-ranked links*: links with $rank(target) > rank(u_P)$, *equi-ranked links*: links with $rank(target) = rank(u_P)$. The links within each of these categories can be processed separately as follows:

1. None of the low-ranked links can be an output as their target will not be an ancestor of u_P, hence can be ignored while querying.
2. For a high-ranked link L_i, if $o_i \in [u_P, u'_P]$, then the condition that t_i is an ancestor of u_P will be implicitly satisfied. Thus, we are left with only 3-constraints, which can be modeled as a 3-sided query [3, 5].
3. We group together all the links whose target node t_i belongs to component C to form a set S_C. Further we replace the origin o_i in each of the links by its lowest ancestor s_i within C (Figure 2). Then, an equi-ranked link $L_i \in C$ is an output iff $t_i < u_P \leq s_i$ and $score_i \geq \tau$, which can be modeled as a 3d dominance query [1].

Putting everything together, the top-k document retrieval problem can be reduced to $O(\log(n/B))$ 3-constraint queries. Thus, by maintaining appropriate structures for handling such queries, we can obtain a linear-space index with $O(\log^2(n/B) + k/B)$ I/Os, which is optimal for $k \geq B\log^2(n/B)$. For optimally handling the case when k is small, bootstrapping techniques are introduced (for details we refer to [39]). We summarize the main result in the following Theorem.

Theorem 2 *There exists external memory index of almost-linear* $O(n \log^* n)$ *words space for answering top-k document retrieval queries in optimal* $O(p/B + \log_B n + k/B)$ *I/Os.*

If the score function is monotonic, the top-k document retrieval problem can be reduced to the *top-k categorical range maxima query* (Top-CRMQ) problem. Given an integer array $A[1...n]$ and associated category (color) array $C[1...n]$, where each $A[i]$ has an associated color $C[i]$, we apply range top-k query (a, b, k) to find the top-k (distinct) colors in the range $[a, b]$. The notion of top-k associates a score with each color c occurring in the query range, where the score of a color c in the range $[a, b]$ is $max\{A[i]|i \in [a, b]$ *and* $C[i] = c\}$. We can now model the top-k document retrieval problem into Top-CRMQ: arrange all links in the ascending order of origin, then construct arrays A and C such that $A[i]$ represents the score of the ith link and $C[i]$ represents the document to which it belongs. Now, top-k document retrieval is equivalent to Top-CRMQ on A with $[a, b]$ as the input range, where $[a, b]$ represents the maximal range of all links with origin within the subtree of u_P. Thus by integrating with the recent solution for the Top-CRMQ problem by Nekrich et al. [36], an external memory top-k document retrieval index with space $O(n\alpha(B))$-words and query I/O bound $O(p/B + k/B + \log_B n + \alpha(B))$ can be obtained, where $\alpha(\cdot)$ is the inverse Ackermann function.

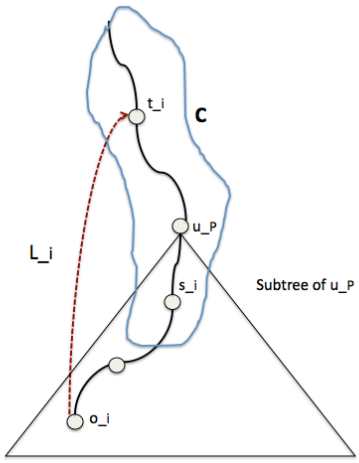

Fig. 2. Pseudo Origin

4 Succinct Frameworks

In the succinct framework, the goal is achieve the index space proportional to the size of text (i.e., $n \log \sigma$ bits). We use the *score* function to be term-frequency. We begin this section by briefly explaining the marking scheme introduced by Hon et al. [23] and then review the later developments in this line of work.

Marked nodes in GST: Certain nodes in the GST can be identified as marked nodes with respect to a parameter g called the *grouping factor* as follows. The procedure starts by combining every g consecutive leaves (from left to right) together as a group, and marking the lowest common ancestor (LCA) of the first and last leaves in each group. Further, we mark the LCA of all pairs of marked nodes. Additionally, we ensure that the root is always marked. At the end of this procedure, the number of marked nodes in GST will be $O(n/g)$. Hon et al. [23] showed that, given any node u with u^* being its highest marked descendent (if exists), number of leaves in $GST(u \backslash u^*)$ (i.e., the number of leaves in the subtree of u, but not in the subtree of u^*) is at most $2g$.

We begin by the describing the data structure for a top-k document retrieval problem for a fixed k. First, We implement the marking scheme in GST as described above with $g = k \log^{2+\epsilon} n$, where $\epsilon > 0$ is any constant. The top-k documents corresponding to each of the $O(n/g)$ marked nodes (as the locus) are maintained explicitly in $O(k \log n)$ bits, for a total of $O((n/g)k \log n) = o(n/\log n)$ bits. In order to answer a top-k query, we first find the locus node u_P, and then its highest marked descendent node u_P^*. If a document d is in the top-k list with respect to node u_P, then either it is in the top-k list with respect to u_P^* as well or there is at least one leaf in the $GST(u_P \backslash u_P^*)$ with the corresponding suffix in document d. By using this observation, we can obtain a set of $O(g + k)$ possible candidate documents. By computing the term frequencies of each document in the candidate set, we can identify the documents in the final output. Note that instead of a GST, we maintain its compressed variant. An additional $|CSA|$ bits structure is used for computing term-frequency in $O(\log^{2+\epsilon} n)$ time, where CSA represents the compressed suffix array [12, 17] of the concatenated text of all documents, and $|CSA|$ represents its size in bits. Thus the query time can be bounded by $O(p + k \log^{4+2\epsilon} n)$. In order to handle top-k queries for any general k, we maintain the above described data structure for $k = 1, 2, 4, 8, ...$, with overall space requirement roughly equal to twice that of the input text.

Theorem 3 *There exists a succinct data structure of space roughly twice the size of text (in compressed form) with query time $O(\log^{4+\epsilon} n)$ per reported document.*

A series of work has been done to improve the above succinct index. The per-document retrieval time is improved to $O(\log k \log^{2+\epsilon} n)$ by Belazzougui and Navarro [6], whereas the fastest succinct index is by Hon et al. [22], where the query time is $O(\log k \log^{1+\epsilon} n)$. Note that

the space occupancy of all these succinct indexes is roughly twice the size of text. An interesting open question to design a space optimal index (i.e., $|CSA| + o(n)$ bits) has been positively answered by Tsur [40], where the per-document report time is $O(\log k \log^{2+\epsilon} n)$. Very recently, Navarro and Thankachan [34] improved the query time of Tsur's index to $O(\log^2 k \log^{1+\epsilon} n)$, and is currently the fastest space optimal index.

Instead of using an additional CSA for document frequency computation of the candidate document, an alternative approach is to use a data structure called the document array $E[1...n]$, where $E[i]$ denotes the document to which the suffix corresponding to ith leftmost leaf in GST belongs to. The resulting index space is $|CSA| + n \log D(1 + o(1))$ bits. The first result of the kind is due to Gagie et al. [15] with per-document report time is $O(\log^{2+\epsilon} n)$, which was improved to $O(\log k \log^{1+\epsilon} n)$ by Belazzougui and Navarro [6], and to $O((\log \sigma \log \log n)^{1+\epsilon} n)$ by Hon et al. [19]. Here σ represents the alphabet size. Culpepper et al. [11] have proposed another document array-based index. Even though their query algorithm is only a heuristic (no worst-case bound), it is one of the simplest and most efficient indexes in practice. Another trade-off is by Gagie et al. [15], where the index space is $|CSA| + O(\frac{n \log D}{\log \log D})$ bits and query time is $O(\log^{3+\epsilon} n)$. This result is also improved by Belazzougui and Navarro [6], where they achieved by a per-document report time of $O(\log k \log^{2+\epsilon} n)$ with an index space of $|CSA| + O(n \log \log \log D)$ bits.

5 Variants of Document Retrieval

In this section, we briefly describe some of the variants of document retrieval problem along with the known results.

5.1 Two-Pattern Document Listing

In this case, the query consists of two patterns P_1 and P_2 (of length p_1 and p_2 respectively), and the task is to report all those *ndoc* documents containing both P_1 and P_2. The first solution was given by [29], which requires $\tilde{O}(n^{3/2})$ space and answers a query in $O(p_1 + p_2 + \sqrt{n} + ndoc)$ time[†]. Clearly, this solution is not practical due to its huge space requirement. Cohen and Porat [9] showed that this problem can be reduced to set-intersection. Based on their elegant framework for the the set-intersection problem, they proposed an $O(n \log n)$-word space index

[†] The notation $\tilde{O}$ ignores poly-logarithmic factors. Precisely, $\tilde{O}(f(n)) \equiv O(f(n) \log^{O(1)} n)$.

with $O(p_1 + p_2 + \sqrt{n \times ndoc} \log^{2.5} n)$ query time. Later Hon et al. [20] improved the space as well as the query time of Cohen and Porat's index to $O(n)$-word and $O(p_1 + p_2 + \sqrt{n \times ndoc} \log^{1.5} n)$ respectively. In addition, Hon et al. [20] extended their solution to handle multipattern queries (i.e., query input consists of two or more patterns) and also to top-k queries. Using Geometric-BWT techniques [8], Fischer et al. [13] showed that in pointer machine model, any index for two-pattern document listing with query time $O(p_1 + p_2 + \log^{O(1)} n + ndoc)$ must require $\Omega(n(\log n / \log \log n)^3)$ bits space.

5.2 Forbidden/Excluded Pattern Queries

A variant of a two-pattern document listing is pattern matching with forbidden (excluded) pattern. Given two patterns P_1 and P_2, the goal is to list all $ndoc$ documents containing P_1 *but not* P_2. Fischer et al. [13] introduced the problem and proposed an index of size $O(n^{1.5})$ bits with query time $O(p_1 + p_2 + \sqrt{n} + ndoc)$. Recently, Hon et al. [21] gave a space-efficient solution for this problem, occupying linear space of $O(n)$ words. However, the query time is increased to $O(p_1 + p_2 + \sqrt{n \times ndoc} \log^{2.5} n)$.

5.3 Parameterized Top-k Queries

In this case, the query consists of two parameters x and y ($x \leq y$) in addition to P and k and the task is to retrieve the top-k documents with highest $w(P, \cdot)$ among only those documents d with $Par(P, d) \in [x, y]$, where $Par(\cdot, \cdot)$ is a predefined function. Navarro and Nekrich [31] showed that such queries can be answered in $O(p + (k + \log n) \log^{\epsilon} n)$ time by maintaining a linear-space index. For the case when $w(\cdot, \cdot)$ is page rank, $Par(\cdot, \cdot)$ is term-frequency and y is unbounded, Karpinski and Nekrich [26] gave an optimal query time data structure with $O(n \log D)$-word space.

6 Conclusions and Open Problems

In this article, we briefly reviewed some of the theoretical frameworks for designing top-k document retrieval indexes in different settings. However, we have not covered the details of practical solutions [25, 37, 27, 33, 35] as well as some of the other related topics (we recommend the recent article by Navarro [30] for an exhaustive survey). Even though many efficient solutions are already available for the central problem, there are still many

interesting variations and open questions one could ask. We conclude with some of them as listed below:

1. The current I/O-optimal index requires $O(n \log^* n)$-word space. It is interesting to see if we can bring down this space to linear (i.e., $O(n)$ words) without sacrificing the optimality in the I/O bound. Designing these indexes in the Cache-Oblivious model is another future research direction.
2. The optimal space-compressed index (by Navarro and Thankachan [34]) takes $O(\log^2 k \log^{1+\epsilon} n)$ query time. The fastest compressed space index (by Hon et al. [22]) takes twice the size of text. An interesting problem is to design a space-optimal index, while keeping the query time the same (or better) as that of the fastest compressed index known.
3. Top-kth document retrieval: instead of reporting all top-k documents, report the kth highest-scored document corresponding to the query.
4. Top-k version of forbidden pattern query: the query consists of P_1, P_2, and k, and the task is to report the top-k documents based on $w(P_1, \cdot)$ among all those documents d which does not contain the forbidden pattern P_2.
5. Another space-time trade-off for parametrized top-k query. For example, design an optimal query time index using $O(n \log^{\epsilon} n)$ words of space.
6. Currently the gap between the upper and lower bound for two-pattern query problem is huge. It is interesting to see if this gap can be reduced. Can we obtain similar (or better) lower bounds for the forbidden pattern query problem. We strongly believe that the lower bounds for this problems are different from the currently known upper bounds [13, 21] by at most $poly \log n$ factors only.
7. Even though many succinct indexes have been proposed for top-k queries for frequency or page-rank based score functions, it is still unknown if such a succinct index can be designed if the score function is term-proximity (i.e., $w(P, d)$ is the difference between the positions of the closest occurrences of P in document d). Designing such an index even for special cases (say, long patterns or allow approximate score, etc), or deriving lower bounds are interesting research directions.
8. Approximate pattern matching (i.e., allowing bounded errors and don't cares) is another active research area [10]. Adding this aspect to document retrieval leads to many new problems. The following is one such problem: report all those documents in which the edit (or

hamming) distance between one of its substrings and P is at most π, where $\pi \geq 1$ is an input parameter.

9. Indexing a highly repetitive document collection (which is highly compressible using LZ-based compression techniques) is an active line of research. In the recent work by Gagie et al. [14], an efficient document retrieval index suitable for a repetitive collection is proposed. An open problem is to extend these results for handling top-k queries.

References

1. P. Afshani. On dominance reporting in 3d. In *ESA*, pages 41–51, 2008.
2. P. Afshani, G. S. Brodal, and N. Zeh. Ordered and unordered top-k range reporting in large data sets. In *SODA*, pages 390–400, 2011.
3. P. Afshani, G. S. Brodal, and N. Zeh. Ordered and unordered top-k range reporting in large data sets. In *SODA*, pages 390–400, 2011.
4. A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Commun. ACM*, 31(9):1116–1127, 1988.
5. L. Arge, V. Samoladas, and J. S. Vitter. On two-dimensional indexability and optimal range search indexing. In *Proc. 18th Symposium on Principles of Database Systems (PODS)*, pages 346–357, 1999.
6. D. Belazzougui and G. Navarro. Improved compressed indexes for full-text document retrieval. In *SPIRE*, pages 386–397, 2011.
7. B. Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *J. ACM*, 37(2):200–212, 1990.
8. Y.-F. Chien, W.-K. Hon, R. Shah, S. V. Thankachan, and J. S. Vitter. Geometric burrows-wheeler transform: Compressed text indexing via sparse suffixes and range searching. *Algorithmica*, 2013.
9. H. Cohen and E. Porat. Fast set intersection and two-patterns matching. *Theor. Comput. Sci.*, 411(40-42):3795–3800, 2010.
10. R. Cole, L.-A. Gottlieb, and M. Lewenstein. Dictionary matching and indexing with errors and don't cares. In *STOC*, pages 91–100, 2004.
11. J. S. Culpepper, G. Navarro, S. J. Puglisi, and A. Turpin. Top-k ranked document search in general text databases. In *ESA (2)*, pages 194–205, 2010.
12. P. Ferragina and G. Manzini. Indexing compressed text. *J. ACM*, 52(4):552–581, 2005.
13. J. Fischer, T. Gagie, T. Kopelowitz, M. Lewenstein, V. Mäkinen, L. Salmela, and N. Välimäki. Forbidden patterns. In *LATIN*, pages 327–337, 2012.
14. T. Gagie, K. Karhu, G. Navarro, S. J. Puglisi, and J. Sirén. Document listing on repetitive collections. In *CPM*, pages 107–119, 2013.
15. T. Gagie, G. Navarro, and S. J. Puglisi. Colored range queries and document retrieval. In *SPIRE*, pages 67–81, 2010.
16. T. Gagie, G. Navarro, and S. J. Puglisi. New algorithms on wavelet trees and applications to information retrieval. *Theor. Comput. Sci.*, 426:25–41, 2012.
17. R. Grossi and J. S. Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. *SIAM J. Comput.*, 35(2):378–407, 2005.
18. W.-K. Hon, M. Patil, R. Shah, and S.-B. Wu. Efficient index for retrieving top-k most frequent documents. *J. Discrete Algorithms*, 8(4):402–417, 2010.

19. W.-K. Hon, R. Shah, and S. V. Thankachan. Towards an optimal space-and-query-time index for top-k document retrieval. In *CPM*, pages 173–184, 2012.
20. W.-K. Hon, R. Shah, S. V. Thankachan, and J. S. Vitter. String retrieval for multi-pattern queries. In *SPIRE*, pages 55–66, 2010.
21. W.-K. Hon, R. Shah, S. V. Thankachan, and J. S. Vitter. Document listing for queries with excluded pattern. In *CPM*, pages 185–195, 2012.
22. W.-K. Hon, R. Shah, S. V. Thankachan, and J. S. Vitter. Faster compressed top-k document retrieval. In *DCC*, 2013.
23. W.-K. Hon, R. Shah, and J. S. Vitter. Space-efficient framework for top-k string retrieval problems. FOCS '09, pages 713–722, 2009.
24. W.-K. Hon, R. Shah, and J. S. Vitter. Compression, indexing, and retrieval for massive string data. In *CPM*, pages 260–274, 2010.
25. M. P. J. S. Culpepper and F. Scholer. Efficient in-memory top-k document retrieval. In *SIGIR*, 2012.
26. M. Karpinski and Y. Nekrich. Top-k color queries for document retrieval. In *SODA*, pages 401–411, 2011.
27. R. Konow and G. Navarro. Faster compact top-k document retrieval. In *DCC*, 2013.
28. Y. Matias, S. Muthukrishnan, S. C. Sahinalp, and J. Ziv. Augmenting suffix trees, with applications. ESA '98, pages 67–78, London, UK, UK, 1998. Springer-Verlag.
29. S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *SODA*, pages 657–666, 2002.
30. G. Navarro. Spaces, trees and colors: The algorithmic landscape of document retrieval on sequences. *CoRR*, abs/1304.6023, 2013.
31. G. Navarro and Y. Nekrich. Top-k document retrieval in optimal time and linear space. In *SODA*, pages 1066–1077, 2012.
32. G. Navarro and S. J. Puglisi. Dual-sorted inverted lists. In *SPIRE*, pages 309–321, 2010.
33. G. Navarro, S. J. Puglisi, and D. Valenzuela. Practical compressed document retrieval. In *SEA*, pages 193–205, 2011.
34. G. Navarro and S. V. Thankachan. Faster top-k document retrieval in optimal space. In *submitted*.
35. G. Navarro and D. Valenzuela. Space-efficient top-k document retrieval. In *SEA*, pages 307–319, 2012.
36. Y. Nekrich, M. Patil, R. Shah, S. V. Thankachan, and J. S. Vitter. Top-k categorical range maxima queries. In *Submitted*.
37. M. Patil, S. V. Thankachan, R. Shah, W.-K. Hon, J. S. Vitter, and S. Chandrasekaran. Inverted indexes for phrases and strings. SIGIR, pages 555–564, 2011.
38. K. Sadakane. Succinct data structures for flexible text retrieval systems. *J. Discrete Algorithms*, 5(1):12–22, 2007.
39. R. Shah, C. Sheng, S. V. Thankachan, and J. S. Vitter. On optimal top-k string retrieval. *CoRR*, abs/1207.2632, 2012.
40. D. Tsur. Top-k document retrieval in optimal space. *Inf. Process. Lett.*, 113(12):440–443, 2013.
41. N. Välimäki and V. Mäkinen. Space-efficient algorithms for document retrieval. In *CPM*, pages 205–215, 2007.
42. J. S. Vitter. Compressed data structures with relevance. In *CIKM*, pages 4–5, 2012.