TO RETRIEVE THE INFORMATION USING DATA STRUCTURES

Subin Paul

Lecturer in Computer Engg

GPTC, Koratty

ABSTRACT

The purpose of this paper is to shed light on the application of data structures in the field of information retrieval. As the need and desire to share and investigate information grows day by day, information retrieval is a growing field of study. Information structures have been the area of examination for a significant stretch in the field of software engineering. As the volume of data increases at an exponential rate, the importance of having effective data structures has increased. Using a computer's memory and processor effectively to efficiently index large collections of documents for information retrieval necessitates judicious use of these resources. Based on the vector-space model (VSM), our method for creating such an index is described in this paper. We go over how an index is created, how terms are weighted, and how documents are represented in the VSM. From parsing the collection of documents to creating index terms and document representations, we explain our choice of data structures.

Keywords: *Data structures, Information retrieval*

INTRODUCTION

The process of "finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)" is referred to as information retrieval (IR) [1]. Full text retrieval systems, like search engines, use mechanisms that go beyond document titles, author names, and other high-level metadata to enable search and access to document content. The fact that the search is performed on unstructured documents, i.e., documents that lack the kind of well-defined structure provided by attributes that is inherent in relational database tables, is what makes information retrieval distinct from and significantly more challenging than traditional relational database search. Unstructured documents like text files and web pages can only be searched by examining all the tokens, typically words, that make up these documents. In contrast, relational database search requires searching for the presence or absence of objects that have specified values in well-defined fields. At search time, when the user provides a search criterion or query (i.e., a set of terms used to describe the search), the information retrieval system should generate a set of documents that hopefully meet the search criterion. This is the traditional method of information retrieval. During indexing, the features of the document, typically words, are extracted. The ways in which IR systems use the query to generate relevant documents can set them apart. Every index term is represented by a Boolean vector of the same size as the number of documents in the collection in Boolean retrieval. In this method, the absence of a term in the corresponding document is indicated by a zero term in the vector, and the presence of a term in the document is indicated by a one term. Consequently, a query is a Boolean expression of term vectors using the operators AND, OR, and NOT to combine terms. The drawbacks of Boolean retrieval are well-documented (for instance, [1], [2], [3], and [4]). The following are examples of these flaws:

1.It is difficult to rank the retrieved documents according to their relevance because Boolean retrieval primarily relies on determining the presence or absence of terms in documents without taking into account important statistics like the frequency with which terms occur in documents or across documents or the number of documents that contain the various query terms.

2.When users use the terms AND, OR, or NOT, which have a different meaning in natural language than in Boolean search engines, it is difficult for them to create efficient Boolean queries.

3.When specifying queries, boolean retrieval requires a controlled vocabulary; The vocabulary should use words that are in the index. Users would prefer to be able to submit queries in text without being restricted by vocabulary.

4.Boolean queries that make use of the AND operator typically result in results with very high precision but, regrettably, very low recall; On the other hand, Boolean queries made with the OR operator produce results with very high recall but unfortunately low precision. It would be helpful to run queries that find a middle ground when it comes to the retrieved documents.

Boolean retrieval has been largely replaced by other retrieval methods that address a number of these flaws since the advent of the Web and the proliferation of electronic documents.

For instance, the Vector-space model (VSM) is utilized for IR much more frequently than Boolean retrieval is. Each document in the VSM is represented as an N-dimensional vector of the N terms that make up the index. The weight of each term in a document is determined by statistics about how frequently the terms occur in the document and across the collection. Any one of a number of similarity measures can be used to figure out how similar two documents are, where a document could be one of the documents in the collection, a user query, a user profile vector, or any N-dimensional representation of the terms in the document collection. Therefore, when performing IR with the VSM, the query vector is compared to each of the collection's documents, and the documents with the highest similarity measures are shown to the user. There are three stages in the vector space model. The indexing of documents is the first step. The dimensionality of the document representation is reduced, first by passing the terms of the document collection as parameters of a stop-list algorithm to remove the less descriptive terms, and then by using a stemming algorithm to ensure that several words with the same stem are treated as the same term. This is done to reduce the difficulties of working in a high-dimensional space, which is known as the "curse of dimensionality." When a query is made, the second stage gives the index terms weights to make it easier to find relevant documents. The term "term frequency inverse document frequency" (TF IDF) is used to describe VSM because the assignment of term weights assumes that the importance of a term in a document is proportional to the frequency of its occurrence within the document (TF) and inversely proportional to its document frequency (DF), i.e., the number of documents that contain the term. Document lengths are normalized in the model to prevent giving preference to longer documents in which terms are likely to appear more frequently.

In IR, it is common to scale the IDF score by using log(N/DF), where N is the number of terms in the collection. The model's final step is to determine how similar the query and document vectors are to one another. Document-to-document similarity can be measured using a variety of methods, the most common of which are cosine similarity, the Pearson correlation index, Jaccard coefficients, and Dice coefficients.

Data has always been and will continue to be a resource that must be utilized and shared wisely for the benefit of institutions and organizations. With the proliferation of social networking sites and technological advancements, data sharing has never been greater. The handled information is called as data and the errand of finding them from the current store are called as data recovery.

Multiple areas of computer science are actively involved in the field of research known as information retrieval. Information retrieval is a tool that is utilized in a variety of advanced computer science research areas and makes use of many of the fundamental ideas of computer science. Text mining, for instance, starts with information retrieval before moving on to other mining operations.

INFORMATION RETRIEVAL

Information retrieval and information extraction

The terms "information extraction" (IE) and "information retrieval" (IR) are frequently used interchangeably. They are very different areas with different goals and unique tasks. There is no objective or target for information extraction. It does make use of templates to give otherwise unstructured information structure. Because it must meet the user's need to find the exact information from an existing repository, information retrieval necessitates advanced methods. Data recovery likewise includes assistant elements of choosing an adept record for better questioning and an insightful data recovery framework utilizes the input from the client to expand upon the current framework and to calibrate the methods. Summarization and clustering are features of information retrieval that are analogous to those of data mining. The general architecture of an information retrieval system is shown in Figure 1.

PERFORMANCE OF AN IR SYSTEM

The response time of the system and the quality of the output are used to calculate an information retrieval system's success rate or performance. Again, a qualitative term, the user's feedback can be used to evaluate the quality of the information retrieval response. Precision and recall are typically the quality measurement metrics. The proportion of relevant documents retrieved to the total number of relevant documents is the definition of the recall measure. The proportion of relevant documents retrieved to the total number of retrieved documents is the precision measure. The relevance of the documents is again a qualitative term, despite the fact that these metrics have precise definitions. The information retrieval system's response time is a measurable quantitative metric. The information retrieval system's response time is influenced by the type of query posed to it, the type of index used, and the size and arrangement of the corpus to be searched. Therefore, the type and size of the corpus, the type of index, the type of the query, and the method of searching must all be taken into consideration when attempting to shorten the IR system's response time. Presently we consider the arrangement of information structures in the field of data recovery and what the decision of the information structures means for the exhibition of the data recovery framework.

DATA STRUCTURES

The various methods that are utilized to store data in persistent memory are referred to as data structures. Each application uses different data structures for different things. The authors of [3] classify the data structures according to their intended use. The information structures like exhibits, connected structures, hash tables are principally utilized for putting away the information and thus are named capacity structures. Process-oriented data structures are the other type of data structures that are used to process data and include stacks, queues, and priority queues. There are still a few data structures that don't just store the data but also help describe the data by how it is arranged in them. linear lists, binary trees, collections, etc. are what the authors refer to as descriptive data structures because they describe the nature of the data they store.

INFORMATION RETRIEVAL TASKS

Answering users' inquiries is the primary goal of information retrieval. The goal of the research is not only to answer the questions that users ask, but also to anticipate the questions that users will ask of a document corpus. In [2], Fei Song and Bruce Croft calculate the probabilities of each document in the corpus generating the user's query terms. Any information retrieval system's efficacy is determined by user feedback. In [4], the authors look at how well the information retrieval works when user preferences are met.

STORAGE

Word-oriented indexing techniques are used in information retrieval to retrieve documents based on user queries using data structures. Signature files [7], inverted files, and other kinds of indexing structures are used, However, the hash function and hash tables are the primary tools they use. Key values and data items are linked in a hash data structure. The search key is mapped to a key value using a hash function. The bucket number that the data item falls under typically appears as the key value. A memory area is all a bucket is. A hash table can be used as an in-memory data structure and is more efficient than most array structures. The hash functions are chosen so as to prevent collision. It has been demonstrated that hashes work better for equality searches than tree structures, which work better for range searches. The hash functions make an index that makes it easier to find the documents that answer the user's query. For filtering, a hash file known as a signature file is used. Typically, the documents that match the query are pinpointed by the filtering. Using the hash function, the filtering process generates a signature for each document. A hash file in an inverted file contains a list of sorted words, each with a set of pointers to the page where it appears. The various data structures utilized for indexing in the process of information retrieval are outlined by the authors of [1]. They talk about how well hashing, B trees, and B+ trees work to put the index structures into action.

PROCESS-ORIENTED DATA STRUCTURES IN INFORMATION RETRIEVAL

A stack is a linear data structure in which data items are stored and retrieved from one end. For string matching in suffix arrays, information retrieval algorithms use a stack. A data structure with nodes and edges connecting is called a graph. It is one of the data structures that is used in a lot of different areas. It has been utilized to determine the connectivity between two distinct computer network nodes or the relationship between two components or data items. In information retrieval, graphs are used to determine the connection between the user's queries and the corpus' documents. The structure similarities between query and document graphs are examined in the implementation of semantic nets and frames. In the field of information retrieval, the search space is also defined by graphs. Fuzzy information retrieval concept networks are built on the foundation of graph structures. A concept or document is represented by each node. An edge connects two distinct concepts, Ci, to a document, Di, in a concept network. The edge is labeled with a real value between zero and one to indicate the relationship's fuzzy weight. Web-based information retrieval also makes use of the graphs to provide a relevance score based on the relevance propagation in the document graph [9]. Collaborative filtering, document classification, and unified link analysis are additional applications of the graphs in the field of information retrieval.

DESCRIPTIVE DATA STRUCTURES IN INFORMATION RETRIEVAL

A data item serves as the root of a tree, and a node serves as the parent node for the subtrees. In a pursuit tree commonly the arrangement is found as leaf. Depending on how the tree is arranged and how it is traversed, different kinds of trees exist. A self-balancing B tree is a binary search tree with this additional quality. The fact that searching only takes a logarithmic amount of time is a benefit of B-tree. A B+ tree is a self-balancing tree whose nodes are linked with pointers and can be adjusted in height. In a B+ tree structure, these pointers make it possible to efficiently carry out range searches. A digital tree is one in which a bit value of 1 is searched in the right tree and a bit value of 0 is searched in the left subtree. For the most part any pursuit activity can be treated as an age of hub in a hunt tree. In the process of retrieving information, binary tree structures such as B trees and B+ trees are utilized for the implementation of the index. The inverted files are implemented using a B-tree. A Prefix B-tree does not store all of the prefixes, but each time the tree is searched, it is rebuilt [5]. B-trees, digital search trees, and key compression techniques are all enhanced by this particular approach. Additionally, it reduces compression techniques' processing overhead. A trie is an information structure that is utilized for putting away a string beginning from the root hub and continuing till the leaf hub. Figure 2 is a portion of [6] in which the authors demonstrate the trie's storage of strings, ape, apple, organ, and organism. In the field of information retrieval, a PAT tree is a binary tree with the abbreviation "Practical Algorithm to Retrieve Information Coded in Alphanumeric" as the short form for PATRICIA. Any path whose interior vertices all have only one child is compressed into a single edge in a simple PAT variant. With a radix of 2, it is a trie data structure, which means that each key bit is compared separately and that each node is a two-way (left versus right) branch. Patricia trees differ from the tries in that they only have one child and no nodes. Either a leaf or at least two children are present at each node. This quickly suggests that the quantity of inner (non-leaf) hubs doesn't surpass the quantity of leaves. Data structures like linear linked lists make it easier to move data items around more quickly. It is a collection of data objects combined with pointers to the next data item. We would have pointers to both the previous and subsequent data items if it were a doubly linked list. Posting lists are implemented with linear linked lists. The list of documents that contain a particular term is maintained by a data structure called a posting list. Typically, a term dictionary is created, and then a posting list with the list of documents that contain each term is created for each term in the dictionary. A skip pointer is a new data structure that is used to traverse a posting list once more. A pointer is typically a variable that stores a data item's address.

CONCLUSIONS

In this paper, we have shown that dictionary and document files can be efficiently generated for reasonably large document collections like big data set by choosing appropriate data structures (primarily BSTs and linked lists) and making reasonable compromises in the face of limited main memory by delegating some processing to secondary storage. Computers with even more memory will be of some assistance to the process as the size of the data sets grows. However, considering other data structures like B-trees, which should be able to read and write much larger blocks of data than binary search trees, might be a more effective strategy.

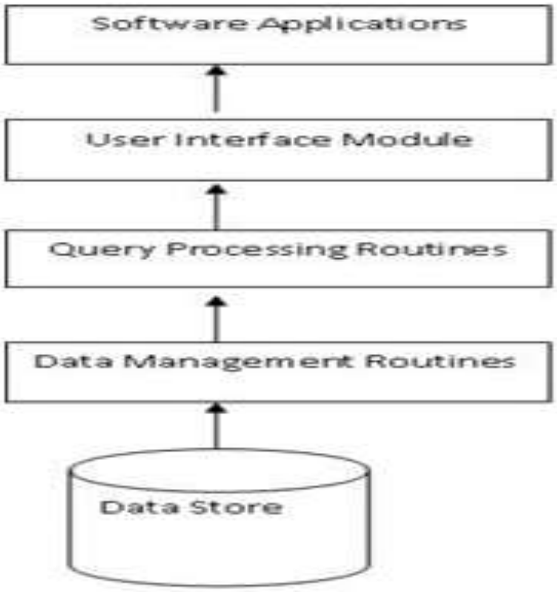

Figure 1. Architecture of a typical IR system

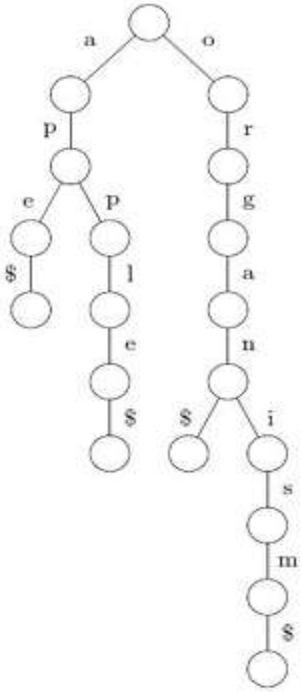

Figure 2: A Trie storing a string

REFERENCES

[1]. Kaufmann, M., Manjili, A.A., Vagenas, P., Fischer, P.M., Kossmann, D., Färber, F. and May, N., (2013, June). Timeline index: a unified data structure for processing queries on temporal data in SAP HANA. *In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 1173-1184.

[2]. Gog, S., Beller, T., Moffat, A. and Petri, M., (2014). From theory to practice: Plug and play with succinct data structures. In *Experimental Algorithms: 13th International Symposium*, *SEA* 2014, Copenhagen, Denmark, June 29–July 1, 2014. Proceedings 13 (pp. 326-337). Springer International Publishing.

[3]. Mehlhorn, K., (2013). *Data structures and algorithms 1: Sorting and searching* (Vol. 1). Springer Science & Business Media.

[4]. Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., Morgan, M.T. and Carey, V.J., (2013). Software for computing and annotating genomic ranges. *PLoS computational biology,* 9(8), p.e1003118.

[5]. Khurana, U. and Deshpande, A., (2013, April). Efficient snapshot retrieval over historical graph data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 997-1008. IEEE.

[6]. Del Blanco, Á., Serrano, Á., Freire, M., Martínez-Ortiz, I. and Fernández-Manjón, B., (2013, March). E-Learning standards and learning analytics. Can data collection be improved by using standard data models?. In *2013 IEEE Global Engineering Education Conference (EDUCON)* pp. 1255-1261. IEEE.

[7]. Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.C. and Steiner, M., (2014). Dynamic searchable encryption in very-large databases: Data structures and implementation. *Cryptology ePrint Archive*.

[8]. Gog, S. and Petri, M., (2014). Optimized succinct data structures for massive data. *Software: Practice and Experience*, 44(11), pp.1287-1314.

[9]. Ribeiro, P. and Silva, F., (2014). G-tries: a data structure for storing and finding subgraphs. *Data Mining and Knowledge Discovery,* 28, pp.337-377.

[10]. Zitnick, C.L. and Dollár, P., (2014). Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014: 13th European Conference*, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13 (pp. 391-405). Springer International Publishing.

[11]. Andoni, A., Indyk, P., Nguyễn, H.L. and Razenshteyn, I., (2014, January). Beyond locality-sensitive hashing. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms,* pp. 1018-1028. Society for Industrial and Applied Mathematics.

[12]. Fan, B., Andersen, D.G., Kaminsky, M. and Mitzenmacher, M.D., (2014, December). Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies* pp. 75-88.