Efficient Information Retrieval from Unstructured Database using Advanced Algorithm

Prof. Sandip shinde[1] Siddhesh Karande[2] Sohail Kasmani[3] Navnath Kate[4] Bhakti Karangale[5]

Abstract:

The amount of information across the world is increasing day by day. Getting the relevant information related to a specific topic in minimum time is the goal of today's efficient information retrieval systems and this is always being interest of the researchers. The information retrieval systems are getting efficient day by day. Here we proposed an efficient information retrieval system to get the relevant documents from the unstructured database using different page ranking algorithms. Different optimization techniques for relevant document searching based on the given query involve removing of the stopping word i.e. the most common occurring words in English language so that do no have a drastic impact on efficiency of our proposed system. Number of precomputations are performed before finding the relevance score of a particular quey in set of documents. This will ultimately minimize the time for searching the relevant documents from the large databases.

Keywords : Information Retrieval , Inverted Indexing , Term Frequency , Cosine similarity

I. INTRODUCTION

Digital data is growing at an ever-increasing rate in both professional and personal lives. Because the capacity to store and share information has far exceeded the ability to search, retrieve, and present it, this has much more impact. The time and money spent storing, managing, and organizing the ever-increasing volume of digital data are becoming increasingly vital in any company and in our everyday lives. Individuals, rather than companies, are now in charge of creating data to a greater extent. The user must get control of the inundation of emails, images, workplace papers, online, and news material. In the expanding data pandemonium, there is a crucial need for proper tools that may assist users in looking for, searching for, organizing, arranging, managing, controlling, and finding important information. Our research is situated in the field of information retrieval (IR). The science of searching for information in documents, searching for documents themselves, searching for metadata that describes documents, or searching within databases, whether relational stand-alone databases or hyper textuality networked databases like the World Wide Web1, is known as information retrieval.

II. LITERATURE REVIEW

There are number of different data structures used for storing the data in the disk. Designing of efficient data structure becomes important to retrieve the information in minimum time and with a smaller number of disk access. This can be achieved with use of efficient data structure for information retrieving. There are number of techniques such as B-Tree formation and Hashing which minimize the time for searching. Multi-level indexing is also used for this purpose [1]. In information retrieval system it becomes important to design such methods which can efficiently retrieve the required information from the database. In this process ranking the relevant documents in order is required. BM 25 is one of the algorithms used for this purpose [2]. To retrieve the information from the database, number of operations are performed on the query. This include parsing the query, tokenization, finding the most related documents, and reordering them. Three different kinds of token can be extracted from query alphanumeric strings, acronyms, and possessives which can be stemmed further [4]. An important class of optimization techniques called early termination achieves faster query processing by avoiding the scoring of documents that are unlikely to be in the top results. A simple augmented inverted index structure called a block-max index which helps in optimization in such cases [3]. Enhance Inverted Index method is used for retrieving information form online documents. This method is more efficient in terms of storage space and processing time as compared to same standard inverted index algorithm [5]. The rank in SVM is demonstrated to determine relevance document using pointwise learning. The results show an average ability of SVM to identify relevant documents is 88.51%, while the average accuracy of SVM to identify non relevant documents is 88%. Also, identification capability of model was not dependent on number of documents in training process[8]. TF-IDF feature extraction technique is presented to compare between the two techniques. The experiments show that TF-IDF improves the performance evaluation of feature extraction according to the maximum value of F1-measure is 89.77 for TF-IDF and 89.16 for BM25[9].

III. METHODOLOGY

BM25:

BM25 (BM stands for best matching) is a ranking function used by search engines to determine the relevance of documents to a particular search query in information retrieval. It is based on the probabilistic retrieval framework. BM25 extends the scoring function for the binary independence model to document and query term weights.

$$\sum_{i}^{n} IDF(q_i) \frac{f(q_i, D) * (k1+1)}{f(q_i, D) + k1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})} \text{--(1)}$$

The equation (1) is the scoring function for which the extension of the IDF score calculation method

q_i is the ith query term.

IDF (q_i) is the inverse document frequency of the ith query term.

$f(q_i, D)$ is how many times does the ith query term occur in document D

k1 is a variable which helps determine term frequency saturation characteristics that is, it limits how much a single query term can affect the score of a given document.

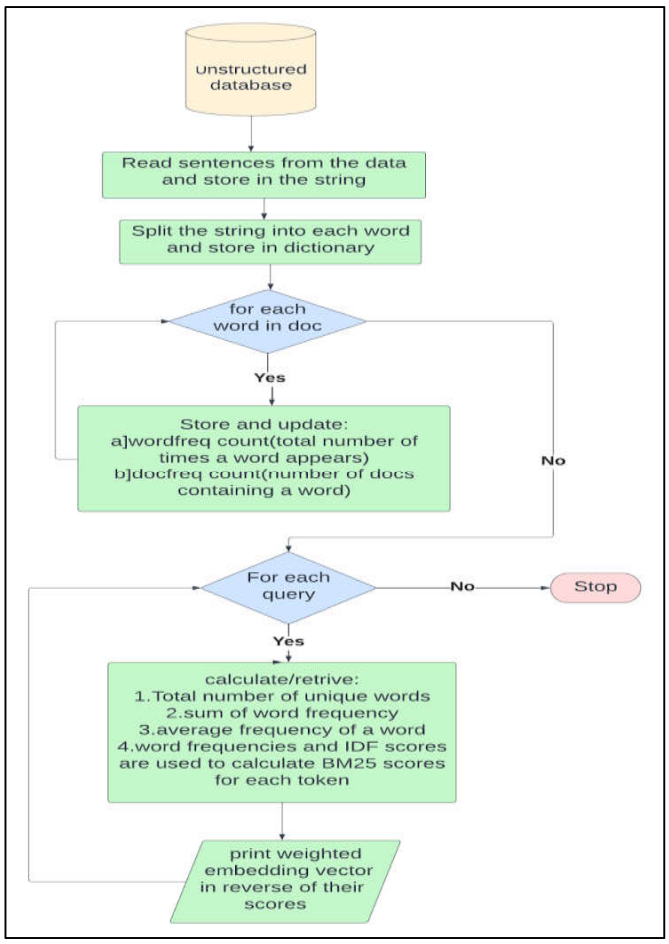

Fig. 1 Proposed system using BM25

The fig 1 explain the way how BM25 algorithm is implemented in our proposed system and all the steps that need to be done before calculating the ranking score (i.e BM25 Score) for the given query. First step include reading the complete document and storing its content in a string. The next step is to tokenize all the string word and store them into a hashing data structure (dictionary) to store the frequency of the word in the whole database of files and do the same to count the number of times a word appears in a doc i.e. calculate the doc frequency. This is the precomputation required before we calculate the ranking score for each query for a given document, Now the next step is to take user query from a query doc and tokenize the query. In this process split each word of the query then calculate the ranking score using the formula in equation 1 for each individual word of the query then merge the result . .

TF-IDF:

The term frequency-inverse document frequency (TF-IDF) is a metric used in the disciplines of information retrieval (IR) and machine learning to quantify the importance or relevance of string representations (words, phrases, lemmas, and so on) in a document among a collection of documents (also known as a corpus).

An Overview of the TF-IDF

The TF-IDF is divided into two sections:

1. TF (Theoretical Framework) (term frequency)

2. International Defense Forces (inverse document frequency).

The term frequency tool examines the frequency of a term in relation to the remainder of the document. There are a few ways to figure out how often something happens:

The frequency with which a term appears in a document (raw count).

The frequency of terms has been adjusted to meet the document's length (raw count of occurrences divided by number of words in the document).

Logarithmically scaled frequency (e.g. log(1 + raw count)).

When dealing with textual data or any natural language processing (NLP) activity, a sub-field of ML/AI that deals with text, the data must first be converted to a vector of numerical values, a process known as vectorization. The process of TF-IDF vectorization entails computing the TF-IDF score for each word in your corpus in relation to that document and then putting that data into a vector.

As a result, each document in your corpus would have its own vector, with a TF-IDF score for every single word in the collection. Once you have these vectors, you can use them to see if two papers are comparable by comparing their TF-IDF files.

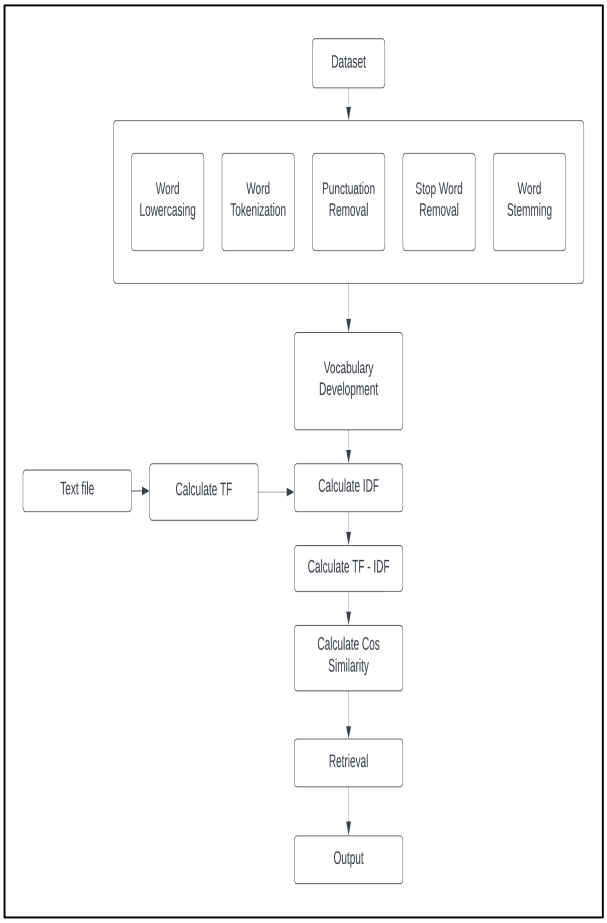

Fig. 2 : Proposed system using TF-IDF

As there is need to execute a series of text pre-processing steps on the original text files. Word lower-casing, tokenization, stop words removal, and stemming are all part of the TF-IDF data processing process. Vocabulary-building is a technique used in the modelling process to tag a set of text documents, indexing each known term, and encode new documents using the index set.

After creating a sparse matrix and a corpus vocabulary, the next step is to count the IDF and TF-IDF values of each word in the corpus and a new text. Only the TF-IDF vector relevant to the entire corpus must be calculated for a new document. We could determine the cosine distance between a new document and all other documents in the corpus by doing a dot product with all other document vectors, allowing us to order comparable documents for retrieval.

IDF and TF-IDF Scores Calculation: TF-IDF Transformer was used to count the IDF value and TF-IDF value of each word in the supplied corpus as well as a new document after Count Vectorizer built corpus vocabulary and a sparse matrix. It should be noted that TF-IDF values for some individual terms may be 0 if they do not present in the training corpus.

Only the TF-IDF vector relevant to the entire corpus has to be calculated for a new document. We could compute the cosine distance between a new document and all other documents in the corpus by doing a dot product with all other document vectors, allowing us to categorise comparable documents for retrieval.

IV. RESULTS

TF-IDF :

TF-IDF is a product of term frequency and inverse document frequency. Fig 3 shows the search results corresponding to the query "*president Obama*". The most relevant document (55) w.r.t the given query is displayed at the top followed by other in descending order of their score.

```
[(55, 0.17149858514250882),
 (54, 0.1307027988094984),
 (46, 0.09454470773792907),
 (48, 0.07211185450587057),
 (50, 0.06748251115357565),
 (47, 0.0650443815788276),
 (56, 0.06370009984030921),
 (49, 0.06357678505803717),
 (6, 0.058758789947793745),
 (40, 0.05873813763229152),
 (43, 0.054313083339851576),
 (44, 0.05125329107048035),
 (41, 0.04133076142720178),
 (53, 0.04129233597173456),
 (36, 0.039176545214350475),
 (45, 0.037107844343551584),
 (51, 0.03573915479294039),
 (17, 0.03537844802603371),
 (14, 0.029618885627681935),
 (57, 0.028863979513802072),
 (7, 0.024671309354004366),
 (52, 0.023993171076371316),
 (30, 0.019608834011474945),
 (58, 0.01647496871071924 7),
 (9, 0.013682777499492663),
 (59, 0.011078914543377756),
```

Fig.3: Search Results for a Query

BM25 :

Fig-4 are the queries used for searching:

```
portabl oper system
code optim for space effici
parallel algorithm
distribut comput structur and algorithm
appli stochast process
perform evalu and model of comput system
parallel processor in inform retriev
```

Fig. 4: Query for searching using BM25

Result for the seven queries in fig. 4 using BM25 is (Top Search Result) as shown in table 1.

Table. 1: Results of the query using BM25

Query(i)	Q_i	Document No.	Score
0	Q0	3127	15.017122036293229
1	Q1	2748	12.269281816283971
2	Q2	2714	6.918361144576037
3	Q3	2914	7.656951257769411
4	Q4	1696	9.55304875171257
5	Q5	2318	17.627607477552374
6	Q6	1811	13.274077819375819

V. CONCLUSION

There are number of information retrieval techniques. In this system two efficient information retrieval techniques are used BM-25 and TF_IDF. Both the algorithms are efficient in ranking the unstructured documents for a given query. Moving forward, the suggested work can be further expanded into a larger framework for text categorization. This allows the inclusion of dimensionality reduction and machine learning approaches.

VI. FUTURE SCOPE

This information retrieval techniques can be further integrated with an online platform which will rank the queries according to the relevance of the documents with the query. Optimize the storage that is currently being used in this system in precomputation steps. Implementing inverted index for information retrieval in large scale system.

VII. ACKNOWLEDGE

Without the outstanding assistance of Prof. Sandip Shinde, this work and the research supporting it would not have been possible. His passion, expertise, and meticulous attention to detail have inspired and kept this work on track.

VII. REFERENCES

[1] “Robust and Efficient Algorithms for Storage and Retrieval of Disk-Based Data Structures” by “Kathiravan Srinivasan; Ravinder Kumar; Sahil Singla”

[2] “Research of Page ranking algorithm on Search engine using Damping factor” by “Punit R Patel”

[3] “Faster Top-k Document Retrieval Using Block-Max Indexes” by “Shuai Ding, Torsten suel”

[4] “PISA: Performance Indexes and search for academia” by “Antonio Mallia, Michał Siedlaczek, Joel Mackenzie, Torsten Suel”

[5] “Enhance Inverted Index Using in Information Retrieval” by “Alia Karim, Duaa Enteesha”

[6] “Algorithms for Information Retrieval” by “RaduDaniel”

[7] “Survey Paper on Information Retrieval Algorithms and Personalized Information Retrieval Concept” by “Parul Kalra ,Bhatia Tanya ,Mathur Tanaya Gupta”

[8] “Learning rank for determining relevant document in Indonesian-English Cross language information Retrieval using BM25”by “Syandra Sari; Mima Adriani”

[9]” Term Weighting for Feature Extraction on Twitter: A Comparison Between BM25 and TF-IDF” by “Ammar Ismael Kadhim”

[10] “A Review on Information Retrieval in Indian Multilingual Languages”by “M.S.Madankar” .International Journal of Advanced Research in Computer Science and Software Engineering. Volume 5 Issue 3; March 2015.

[11] “Visual analysis on the research of cross-language information retrieval” by “Zhao Rongying” IEEE, pp-107-113; 2008.”