Implementation of an Information Retrieval System

Penjan Antonio Eng Lim

Department of Computer Science and Engineering, Chungnam National University

daehanlim@o.cnu.ac.kr

ABSTRACT

In an era dominated by digital content, an Information Retrieval System embodies capabilities that enable the storing, retrieving, and managing of data elements. Data elements can extend to cover a range of formats including, but not limited to, textual content (with the possible inclusion of numerical and temporal data), imagery, audio, video, and various other multimedia constructs. This project explores the design and execution of an Information Retrieval System, adept in boolean and ranked retrieval methodologies. The design begins with the building of a uni-gram inverted index using the Single-Pass in Memory Indexing (SPIMI) algorithm. The system then navigates complex Boolean queries and conducts a ranked retrieval process that can handle free-text queries, ranking the results based on their relevance to the query.

1. INTRODUCTION

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) [1]. It is the practice of procuring relevant data resources that cater to a specific information requirement from a vast pool of such resources. This retrieval can hinge on metadata or be driven by full-text or additional content-oriented indices. The process of Information Retrieval starts when a system receives a user query. Its primary concern lies in discovering and prioritizing documents aligning with the user's information demands [2].

One of the first major concepts in IR is the inverted index. It is a structure that uses a dictionary of terms and a list of *postings* for each term in order to offer an efficient path to access relevant data. This index significantly enhances the speed and effectiveness of search mechanisms within large document corpora, representing a fundamental element in modern search engines. One practical implementation of the inverted index is the Single-Pass in Memory Indexing (SPIMI) method, an approach specifically designed to handle the indexing of large-scale datasets. This method, known as SPIMI-Invert, breaks down large amounts of data into manageable chunks, creating partial indices in memory that are subsequently merged into the final inverted index [2].

Another potent instrument in Information Retrieval is Boolean retrieval. It employs Boolean logic—using operators like 'AND', 'OR', and 'NOT'—to refine user queries. This methodology permits intricate and precise search requirements, enhancing the capacity to pinpoint desired information resources within a vast dataset [1].

Lastly, we delve into ranked retrieval, particularly employing the TF-IDF (Term Frequency-Inverse Document Frequency) weighting scheme. Unlike Boolean retrieval, which delivers an unranked set of matching documents, ranked retrieval arranges the documents in order of relevance. TF-IDF assigns a weight to each term in a document

that reflects its importance, which is then used to rank documents for a given query. This technique has been fundamental in Information Retrieval, enabling users to sift through large collections of documents swiftly and effectively.

This paper describes the process of building a simple indexing and Information Retrieval System using the concepts explained above. Through a comprehensive analysis of the system's functionality, the paper aims to elucidate the power and simplicity of these techniques within the context of IR and to illustrate how these methods facilitate the retrieval of relevant documents from a large corpus based on a given user query.

The rest of this paper is arranged as follows: Section 2 elaborates on the methodologies applied, Section 3 examines the system's results, and Section 4 encapsulates the key findings and conclusions.

2. METHODOLOGY

In the realm of IR, multiple strategies can be employed to ensure effective and efficient extraction of data from large repositories. This section highlights the development and usage of two critical strategies, namely Boolean retrieval and ranked retrieval, building upon the foundation of an inverted index. The section presents a detailed description of the methodology used in developing a simple yet effective IR system. In order to provide comprehensive and accurate results, the methodologies chosen rely on several Python packages such as NLTK [3] for stopword removal, and SpaCy [4] for sentence tokenization.

2.1 Inverted Index

In essence, an inverted index is a dictionary-like data structure that allows for efficient keyword searches. Each term in the index has a corresponding list of postings, which are essentially the document IDs where the term appears. This structure is instrumental in enhancing the search mechanisms' speed and effectiveness within extensive document corpora, thereby playing a pivotal role in modern search engines.

The construction of the inverted index for this project utilizes a variant of the SPIMI-Invert (Single-Pass in Memory Indexing) algorithm. SPIMI-Invert is an algorithm tailored to efficiently index large datasets by breaking the data into manageable chunks, creating partial indices in memory that are subsequently merged into the final inverted index. This method has been adjusted to account for the ample free memory available in the current context, eliminating the need to write the index and dictionary to disk, as required in the original algorithm. This modification provides an improvement in terms of speed and efficiency in the indexing process, allowing for rapid access to the dataset and the generation of relevant search results.

During the index creation process, each document is first read and preprocessed by removing punctuations, numbers, URLs, and other unnecessary special characters, normalizing the text (i.e., converting all characters to lowercase), and performing any additional preprocessing steps required to prepare the data for indexing. The document is then segmented into sentences using the SpaCy [4] library's sentencizer. Each sentence is also preprocessed and then split into individual tokens. These tokens are then filtered to remove stopwords (commonly used words that are filtered out due to their lack of meaningful content). Finally, these tokens are added to the postings list of each corresponding term in the inverted index, along with the document ID from which they originated. The result is a dictionary of words in which, for each term, we have a list that records which documents the term occurs in, also referred to as a postings list (or inverted list).

Once the inverted index is constructed, it is utilized in the retrieval process to implement two major methodologies: Boolean retrieval and ranked retrieval.

2.2 Boolean retrieval

Boolean retrieval is an integral part of the Information Retrieval System, allowing for a streamlined, precise, and versatile approach to document search. The method employs Boolean logic to offer flexibility and specificity in user queries. By enabling intricate search requirements, it enhances the capability to pinpoint desired information resources within a vast dataset.

The Boolean retrieval system in this paper is capable of managing a diverse range of queries, including `X OR Y`, `X AND Y`, `X AND NOT Y`, and `X OR NOT Y`, as well as complex combinations of these elementary forms. The implementation for this was achieved through a three-step process: parsing the query, processing the 'NOT' operator, and finally, combining postings.

To begin with, each search query received by the system is parsed to separate and identify its individual components. The query first undergoes the equivalent preprocessing phase that was applied to each document during the construction of the index, as detailed previously. This ensures that the tokens derived from the query match those that are stored in the index. This consistency allows for accurate matching between the query terms and the terms in the documents, thereby improving the effectiveness of the retrieval process. The query is then split into individual tokens, distinguishing between Boolean operators (OR, AND, NOT) and search terms. If the token is a search term, it is mapped to its corresponding postings list from the inverted index, or an empty set if the term does not exist in the index.

Following the parsing, the 'NOT' operator is processed separately, as it represents a negation operation, which essentially removes documents from consideration. To implement this, the function replaces each `NOT Y` in the parsed query with the set of documents that do not contain term Y. This is achieved by subtracting the postings list of term Y from the set of all document IDs.

After the 'NOT' operators have been processed, the remaining operators 'AND' and 'OR' are dealt with by combining the postings of terms using the corresponding Boolean operations. For each 'AND' operation, the intersection of the postings of the two terms is computed, ensuring that only the documents that contain both terms are kept. For each 'OR' operation, the union of the postings is computed, retaining any document that contains either of the terms.

The result of the Boolean retrieval process is a set of document IDs that satisfy the conditions specified by the original user query. The system then retrieves these documents for the user to review.

This Boolean retrieval system, while simple, is highly effective and fundamental in the field of Information Retrieval. It provides a powerful method for sifting through a large corpus based on user-defined criteria, illustrating the importance of logical operators in data search and retrieval.

2.3 Ranked retrieval

Ranked retrieval adds another dimension to Information Retrieval by sorting the resulting documents according to their relevance to the user's query. The ranking of retrieved documents is particularly useful when dealing with large sets of data where a binary Boolean retrieval might return an overwhelming number of documents. By ordering the documents by their relevance score, the ranked retrieval system ensures that the most relevant documents are presented first to the user, thus making the information search more efficient and user-friendly.

The scoring mechanism used for ranking in this study is the TF-IDF (Term Frequency-Inverse Document Frequency) scheme. TF-IDF is a statistical measure that reflects how important a word is to a document within a corpus. The TF-IDF score increases proportionally to the number of times a word appears in the document (tf) and is offset by the frequency of the word in the corpus (df). It combines the definitions of term frequency and inverse document frequency to produce a composite weight for each term in each document. This weight is calculated as follows:

$$w_{t,d} = \log\left(1 + tf_{t,d}\right) * \ log_{10}(N/df_t) \qquad (1)$$

where t denotes the term and d denotes the document. In the project, Laplace smoothing is applied when calculating the second term in Equation (1).

As for the ranking strategy, Term-At-A-Time (TAAT) retrieval was employed. In this approach, the query is processed one term at a time, scoring each document for each term before moving on to the next term in the query.

The system first computes the term weights for the documents using Equation (1). For each document, it computes a vector where each element represents the term weight.

Just like in Boolean retrieval explained previously, the input query is preprocessed and tokenized similarly to the documents. The system then computes the term weights for the query similarly to the documents.

With the weights calculated, the system computes the scores for each document using cosine similarity. Both the query and each document are represented as vectors in a space where each dimension corresponds to a term from the query. The cosine of the angle between each document vector and the query vector is calculated, producing a score that represents their similarity. A higher score indicates that the document is more relevant to the query.

Lastly, the system ranks the documents according to these scores and returns the top 'K' documents, where 'K' is a user-defined parameter that dictates how many top-scoring documents should be returned. This method enables the user to control the number of results according to their needs. This process is illustrated in Figure 1.

```
CosineScore(q)
1  float Scores[N] = 0
2  float Length[N]
3  for each query term t
4  do calculate w_{t,q} and fetch postings list for t
5     for each pair(d, tf_{t,d}) in postings list
6     do Scores[d] += w_{t,d} × w_{t,q}
7  Read the array Length
8  for each d
9  do Scores[d] = Scores[d]/Length[d]
10 return Top K components of Scores[]
```

Figure 1: Cosine score calculation

The implementation of ranked retrieval with the TF-IDF scoring mechanism enhances the precision and utility of the information retrieval system. It underscores the importance of considering not only the presence of query terms in a document but also their significance in both the document and the entire corpus, leading to a more nuanced and effective retrieval of information.

3. RESULTS

The dataset used for the index construction and system testing is the ‘stories’ dataset, sourced from the TEXTFILES.COM archive. This dataset consists of 466 stories written in English, each belonging to various genres, stored in a multitude of different file formats.

3.1 Index Construction

Table 1 presents a set of metrics encapsulating various aspects of the constructed inverted index and data. The first column presents the

number of unique terms in the index, also known as the vocabulary size. The second column reports the average number of terms per document. This metric can hint at the complexity and richness of the document content. The third column presents the average size of the postings lists. It represents the average number of documents in which the terms appear. The final column displays the memory usage of the inverted index. This figure is computed by calculating the amount of memory (in bytes) consumed by the index structure in memory. Memory usage provides a crucial gauge of the computational efficiency of the index.

# of unique terms	Avg # of tokens per doc	Avg size of postings lists	Mem usage
49,189	2335	8	2.5 MB

Table 1: Statistics of index construction

As can be seen in the table, the memory usage of the inverted index is relatively small, implying that the index is efficient in terms of memory consumption.

3.2 Boolean Retrieval Results

As part of the experimental evaluation, 20 Boolean queries on the retrieval system were executed. The model returned results that align closely with the intended information need expressed in each query.

The first query tested was "depressed AND sad". The system simply returned all documents that contained both the terms "depressed" and "sad". Out of the 466 documents in the corpus, 10 documents satisfied this condition and thus were retrieved.

Then the query "depressed AND sad AND NOT cool" was tested. Here, the system provided a list of documents containing the term "depressed" and “sad” as before, but this time excluded those also containing the word "cool". This time, the system only retrieved 6 documents, as 4 of the documents retrieved in the previous query contained the word “cool”.

Another query "health OR disease" was conducted to retrieve documents containing either "health" or "disease". This broader query correctly returned a larger set of 62 documents, showcasing the system's ability to handle queries with an 'OR' operator.

These tests demonstrated that the Boolean retrieval system could accurately interpret and respond to Boolean queries, effectively filtering the document corpus based on the Boolean operators applied.

3.3 Ranked Retrieval Results

For the 20 free-text queries executed on the ranked retrieval system, results were ordered based on their relevance score. One of the queries utilized was "Bill Clinton and George Bush". The system notably favored documents that contained a balanced discussion of both figures. For example, the top document is a narrative featuring past U.S. presidents, including Bill Clinton and George Bush. In the narrative, the terms 'Bill Clinton' and 'George Bush' appear in close proximity, indicating a strong relationship to the search query.

Another example is "Depressed and sad", which was one of the queries tested for Boolean retrieval. On executing this query, the ranked retrieval system generated a list of documents that it deemed most relevant to the user's query. The top-scoring document was a poem titled "PEEL ME AWAY" by Bill DeClercq. This poem creatively expresses themes of depression and sadness through the metaphor of peeling layers of an onion. The often recurrence of words such as "depressed", and "sad", and the overarching melancholic tone of the poem contributed to its high relevance score.

3.4 Comparison between Boolean and Ranked Retrieval

When comparing Boolean and ranked retrieval, each has its unique strengths and applications. Boolean retrieval excels in scenarios where the user has a clear and specific information need that can be expressed using Boolean operators. It ensures that returned documents adhere strictly to the user's stated criteria.

Ranked retrieval, on the other hand, is particularly effective for more ambiguous or exploratory queries. By ranking documents based on a measure of relevance, it helps users navigate large result sets and prioritizes documents that most closely align with the user's information needs. It is also able to handle free text queries, which are more suitable for less advanced users.

In the experiments, the Boolean queries including `AND` and the ranked retrieval queries involving several terms resulted in different outputs. In Boolean retrieval, we received a list of all documents containing all the terms between the `AND` keyword. However, ranked retrieval provided a more refined list where documents discussing terms in the query in-depth were ranked higher.

4. CONCLUSIONS

Throughout this paper, we explored the process of building an Information Retrieval System, underlining the effectiveness of different methodologies in retrieving pertinent documents from a vast collection based on user queries. The analysis covered several essential concepts in the Information Retrieval domain, including the inverted index, Boolean retrieval, and ranked retrieval.

The system showed that an inverted index is a powerful tool for efficient data retrieval, significantly enhancing search mechanisms within large corpora. The efficacy of both Boolean and Ranked retrieval methods was put to test through several query examples. While Boolean retrieval excelled at retrieving precise results based on strict user criteria, the Ranked retrieval, utilizing the TF-IDF weighting scheme, demonstrated a capacity to prioritize documents based on relevance.

REFERENCES

[1] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[2] G. J. Kowalski, *Information Retrieval Systems: Theory and Implementation*. Springer, 2007.

[3] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009. [Online]. Available: https://www.nltk.org/book/

[4] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017.