



Information Retrieval System for Circulars

**A dissertation submitted for the Degree of Masters
of Computer Science**



D. P. Wijeratne

University of Colombo School of Computing

2022

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

Student Name: Wijeratne D.P.

Registration Number: 2017/mcs/094

Index Number: 17440941

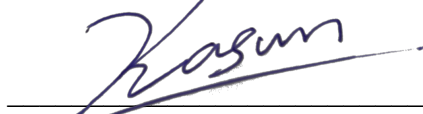
Danushka 2022-11-20

Signature of the Student & Date

This is to certify that this thesis is based on the work of Mr. /Ms. D.P. Wijeratne under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by,

Supervisor Name: Dr. Karunanayaka K.A.K.T.



20/11/2022

Signature of the Supervisor & Date

I dedicate this achievement to my parents, teachers and everyone contributed to the success of this process for me to get fortified with knowledge and to march towards the path of success

ABSTRACT

Revolution of physical infrastructure, increasing population, and advancement in technology contributed to rapid increase in published information. This rapid growth of published information will give potential users problems such as how to find specific information from a large document corpus. The researcher focused on circular corpus and identified the absence of a well functioning circular retrieval system will lead to significant problems such as need to put a lot of effort and time to search for a specific circular which interested in. This dissertation will address these issues by implementing a circular storage and retrieval system that will enable users to quickly and easily retrieve the circulars what they are looking for. Along with circular management system a personalized circular recommendation system will also be built which recommends circulars based on previously viewed circulars.

Circular storage and retrieval system will be implemented on top of circular document collection maintained by a government ministry. To enable users to quickly and easily retrieve circulars, the system

- 1) Allows users to enter tags specific to circulars when uploading circulars.
- 2) Allows users to narrow down the searched results by advanced filter criteria such as words to include, words to exclude, matching phrase, year of circular and tags.
- 3) Query match happens using an inverted index and circulars will be ranked using BM25 algorithm before passing the results to front end.

Also for the ease of the user there will be a personalized circular recommendation system to recommend circulars based on previously viewed circulars and that is implemented using TF-IDF matrix and cosine-similarity matrix.

TABLE OF CONTENT

DECLARATION	i
ABSTRACT	iii
TABLE OF CONTENT	iv
LIST OF FIGURES	vi
LIST OF TABLES	vii
LIST OF ABBREVIATIONS	viii
CHAPTER 1 - INTRODUCTION	1
1.1. Motivation	1
1.2. Statement of the Problem	1
1.3. Research Aims and Objectives	2
1.3.1. Aim	2
1.3.2. Objectives	3
1.4. Research Questions	3
1.5. Scope	4
CHAPTER 2 - LITERATURE REVIEW	6
2.1. Information Retrieval	6
2.2. Background Study	7
2.2.1. Current Information Storage and Retrieval Systems	7
2.2.2. Document Categorization	8
2.3. Basic Components in an Information Retrieval System	13
2.3.1. Inverted Index	13
2.3.2. User Query Types	16
2.3.3. Document Retrieval and Ranking	18
• Boolean Model	18
• Vector-Space Model (VSM)	18
• Bag of Words Assumption	19
• Term Frequency–Inverse Document Frequency (TF-IDF)	21
• Best Match 25 (BM25)	22
2.4. Recommender System	23
2.4.1. Content Based Recommenders	23
2.4.2. Collaborative Filtering	25
2.4.3. Knowledge-based Recommender Systems	26
2.4.4. Hybrid Recommender Systems	26
2.5. Document similarity measures	27
• Cosine Similarity	27

CHAPTER 3 - METHODOLOGY	29
3.1. Data Retrieving and Pre-processing	29
3.1.1. Data Retrieving	29
3.1.2. Bulk Uploading	30
3.1.3. Adding Tags, Filter with Tags Which PDF belongs to	30
3.1.4. Pre-processing	31
• Removal of punctuations and white spaces	32
• Tokenization	32
• Conversion to lower cases	32
• Removal of stopwords	32
• Stemming	33
3.1.5. Crawling and Indexing	33
3.1.6. Searching and Ranking	35
3.1.7. Recommendation System	38
3.1.8. Presentation of Circulars	40
CHAPTER 4 - EVALUATION AND RESULTS	42
4.1. Introduction	42
4.2. Features to be Tested	43
4.2.1. Admin Side Tests	43
4.2.2. User Side Tests	43
4.2.3. System Side Tests	47
4.3. Test Scenarios	47
4.4. Testing Approach	48
4.4.1. Upload Test	48
4.4.2. Recommendations Test	53
4.4.3. PDFs That We Have Used To Test	56
4.5. The Resources Allocated to Test The App	57
4.6. Test Results	57
CHAPTER 5 - CONCLUSION AND FUTURE WORK	62
5.1. Conclusions	62
5.2. Limitations	62
5.3. Future Work	62
REFERENCES	I

LIST OF FIGURES

■	CHAPTER 1 - INTRODUCTION	
○	Figure 1.1 - Structure of a Circular	02
■	CHAPTER 2 - LITERATURE REVIEW	
○	Figure 2.1 - Inverted Index Example	15
○	Figure 2.2 -Example of Boolean Queries	17
○	Figure 2.3 - Graphical Representation of Term Frequency-Inverse Document frequency	22
○	Figure 2.4 - Personalized Example of Recommandations	25
○	Figure 2.5 - Major Functionalities of a Knowledge-based Recommender System	27
○	Figure 2.6 - Graphical Representation of Cosine Similarity Calculation	28
■	CHAPTER 3 - METHODOLOGY	
○	Figure 3.1 - Architecture of the Proposed circular management and recommendation system	30

LIST OF TABLES

■	CHAPTER 2 - LITERATURE REVIEW	
○	Table 2.1 - Document-Term Matrix (DTM)	21
■	CHAPTER 4 - EVALUATION AND RESULTS	
○	Table 4.1 - Expected Outcomes of Normal Search at Every Possibilities	44
○	Table 4.2 - Expected Outcomes of Advance Search at Every Possibilities	45
○	Table 4.3 - PDFs to Upload Under “Efficiency Bar Examination”	49
○	Table 4.4 - PDFs to Upload Under”Agrahara Insurance Scheme”	50
○	Table 4.5 - Normal Search Test Cases	50
○	Table 4.6 - Advance Search test Cases	51
○	Table 4.7 - Scores that were calculated in the recommendation system when the PDF "EBE01" was first opened	55
○	Table 4.8 - Scores that were calculated in the recommendation system when the PDF "AIS01" was first opened	56
○	Table 4.9 - Scores Calculated in Recommendation System when PDF “EBE01” and PDF “AIS01” was Opened Before	57
○	Table 4.10 - Results of the Test are shown below	58

LIST OF ABBREVIATIONS

- IR - Information Retrieval
- IT - Information Technology
- TF-IDF - Term Frequency–Inverse Document Frequency
- EM algorithm - Expectation - Maximum algorithm
- AHC - Agglomerative Hierarchical Clustering
- LSA - Latent Semantic Analysis
- LDA - Latent Dirichlet Allocation
- VSM - Vector Space Model
- CBOW - Common Bag of Words
- BM25 - Best Match 25
- NLP - Natural Language Processing
- BoW - Bag of Words
- DTM - Document-Term Matrix
- AI - Artificial Intelligence

CHAPTER 1

INTRODUCTION

In this chapter we are going to discuss about the motivation, statement of the problem, research aims and objectives, research questions and the scope of this research.

1.1. Motivation

Revolution of physical infrastructure, increasing population, and advancement in technology contributed to rapid increase in published information. This rapid increase of published information made life difficult for information managers, who are facing problems such as how to store the information for quick and easy retrieval. Also this rapid growth of information will give potential users problems such as how to find specific information from a large document corpus.

1.2. Statement of the Problem

The researcher's goal for this project is to implement a simple method for managing circulars. The system will be based on circulars published by, Ministry of Provincial Councils, Public Administration, Home Affairs and Local Government of Sri Lanka (www.pubad.gov.lk). This Ministry circulars corpus is also accessible to the public on their website (<https://goo.by/jTYmv>).

A circular is essentially a letter that is sent to a large number of people with important information (**Toppr-guides, 2018**). If, for instance, you need to invite every department to a meeting or update the office's dress code, a circular will be the most effective method of communication.

Some of the unique features of Circulars from other documents are as below (Example Figure 1.1).

- A circular no for each document
- These will contain new policies

Public Administration Circular : 10/2021

Circular No

My No: F/FR/03/BRCir
Ministry of Public Services|
Provincial Councils and Local
Government Independence Square
Colombo 07.

Secretaries to Ministries
Heads of Departments
District Secretaries

linking another circular

Revenue Estimates for Year 2022 for Revenue Code 20.02.01.01,
Rent on Government Buildings & Housing

Your attention is drawn to the Department of Fiscal Policy Circular No:01/2015 dated 20.07.2015 and revisions made to the same which consists of guidelines in respect of estimating, collecting, supervising and reporting of government revenue.

02. Therefore, you are kindly informed to prepare estimates on revenue expected to be collected by your office for the year 2022 under revenue code 20.02.01.01, revenue of rent on government buildings & housing as per the annexed format and to submit the same to this Ministry on or before 25.06.2021.

Sgd/ J.J. Rathnasiri
Secretary

Figure 1.1 - Structure of a Circular

The absence of a well-functioning circular storage and retrieval system has led the researcher to build an Information Retrieval (IR) system for circulars.

As there is no proper circular retrieval system, users will find difficulty when searching for a circular that they are interested in.

1.3. Research Aims and Objectives

1.3.1. Aim

This dissertation's research objective is to create an IR system for circulars that will enable users to efficiently search through circulars. This will help users to discover relevant information from a storehouse containing a collection of circulars.

1.3.2. Objectives

- Identify how an authorized person can upload circulars which will allow users of the system for quick and easy retrieval of circulars.
- Identify what are the optimal filtering criteria that will enable users to quickly and easily retrieve the circulars that they are looking for.
- Identify what is the best way to search the matching results for the user entered query.
- Identify how to build a recommendation system to recommend circulars to users by analyzing their previous searches

1.4. Research Questions

1. How can an authorized person upload circulars which will allow users of the system for quick and easy retrieval of circulars?

Along with the uploaded circular document to the system, by capturing more information relevant to a circular will help to quickly and easily retrieve information which they are looking for.

2. Identify what are the optimal filtering criteria that will enable users to quickly and easily retrieve the circulars that they are looking for.

Here we have to identify the query types which the user will be allowed to enter for easy retrieval of circulars. Also along with the option to enter a query we will identify what other filterings that will be available to users which will increase the performance of circular retrieval.

3. What is the best way to search the matching results for the user entered query?

Searching for the matching results for the query which the user entered can be done in many ways. The easiest way is to loop through every document in the corpus and check the user entered query exists. This process will take a long time to process. Therefore, we have to identify the optimal way of searching user entered query through a large document corpus.

4. How to build a recommendation system for circulars?

Many recommendation systems are built by analyzing the previous search history of users. Therefore, we also have to build a recommendation system based on the circulars which users are most interested in.

5. How to evaluate the system which was built?

The system which will be built mainly consists of three sub components.

They are,

- Document upload component
- User query component
- Recommendation component

To test this we will be using a selected set of circulars. We will upload those circulars and write test cases which will cover all the components which mentioned above.

1.5. Scope

The study mainly focused on implementing an IR system and recommendation system for circulars. Therefore, this research will primarily be focused on implementing the following characteristics.

1. Upload documents to the system

This research will focus on circulars. Therefore, there will be a feature to add circulars to the system. Also, circulars uploading can be done only by authorized users and for those users there will be a separate login. After login, when uploading circulars users can specify the tags which circular mainly refers to.

2. Filter circulars by submitting user queries, year and tags

Posed in query language, this specifies what the user wants to know. This can be a simple search or advanced search. In simple search users can enter either a list of keywords or a phrase. In advanced search users can enter a combination of keywords, a phrase and exclude words to filter out

documents more specifically. Also for simple and advanced search users are allowed to filter by using circular year and tags too.

3. Process the user query and return the matching circulars

This step will return circulars which the IRS considers relevant to the user query. Relevance implies that the results in the outcome are likely to be useful to the person who submitted the request.

4. Presentation of result

Circulars which match the user query are shown as hyperlinks. User can click on these hyperlinks and navigate to detailed content which he is interested in.

5. Recommend circulars by analyzing user search history

Circulars will be recommended for future reference by analyzing user search history. As the input for the recommendation system, circulars are passed in descending order by time the user spent on reading circulars.

CHAPTER 2

LITERATURE REVIEW

This chapter talks about the IR system existing right now, similar document categorization mechanisms, indexing techniques, user query types, document retrieval ranking techniques, recommender systems and document similarity measurement methods. Here, the advantages and disadvantages of those components are identified, and insights are obtained to address the research problem.

2.1. Information Retrieval

Searching on the Web is the same as searching for a telephone number in an unsorted directory. It will be really hard to find something for a user who doesn't know where to start? This is the work of search engines to facilitate things.

Search engine provides three main facilities to users:

- It brings together a collection of documents into a corpus from which a user can retrieve information. This process is referred to as crawling. **(Castillo, C., 2005)**
- It organizes information in a manner that can be readily and quickly retrieved. That is what is known as indexing. We're going to discuss it here later.
- It allows users to make requests and find corresponding results from the corpus without any difficulty. It is called search and is discussed later in more detail.

Information is gathered by search engines in two primary ways. A person or business first creates a new page and then submits it to the search engine. According to **Michael Gordon and Praveen Pathak (1999)**, the second method involves search engine companies developing crawlers, or spiders, that collect link-by-link web pages for new corpus material. However, in order to make the system more useful, some businesses employ both methods of information collection (Like Google Inc.). The strategies that website owners employ for Search Engine Optimization, or SEO, make search engine tasks significantly simpler.

Internet users frequently search for information and have no idea where to find it due to the complexity of the Internet architecture. Researchers have facilitated this problem through the development of search engines. How do users look for

information? Users type a query and perform a search. Then it gets sent to the backed (Search engine). Next, the search engine will search for the query for this information in the indexes that are already indexed. Finally returns the output and displays it for the user. Search engines are a good example of mass-scale information search systems. (Alexandros Ntoulas, Junghoo Cho, and Christopher Olston. 2004)

2.2. Background Study

2.2.1. Current Information Storage and Retrieval Systems

The current Web search engines are specialized information search systems, designed to be able to search the very large collection of documents called the World Wide Web. One of the main characteristics of Web search engines is their usability, which is one of the main reasons for their popularity among users. Most sites like Google (<https://www.google.com>) provide users with a simple interface that is also very powerful to retrieve data.

Google Scholar (<http://scholar.google.com>): its design and manipulation are similar to those used by the popular search engine provided by Google Inc. It is a simple means of doing a general search in scholarly literature. In a particular place, you can search through a number of disciplines and sources such as articles, theses, books, abstracts etc. Google Scholar search results can be limited to a couple of filters like title, author, publication source and publishing date

In 1997, PubMed (<http://www.ncbi.nlm.nih.gov/PubMed>) was made available through the National Medical Library on the Internet. It is one of the most popular and responsible World Wide Web resources for medical practitioners and academics (Falagas ME, Pitsouni EI, Malietzis GA, Pappas G, 2008). PubMed is a free search engine that carries out research on the medicine and literature of biomedical journals. It searches a number of Medline databases and interfaces directly. This search engine combines the search terms of the user with the header of the medical topic (mesh) and the text words in the Med-line records, then with the search (Anders ME, Evans DP, 2010). PubMed provides users with multiple powerful search filters to limit their searches and provide them with desirable search information. (<https://pubmed.ncbi.nlm.nih.gov/help/>)

Science Direct (<http://www.science direct.com>) is a full-text scientific database that forms part of the scientific verse and is provided by the 1997 Elsevier publication. The Science Direct web portal opens with features that invite users to simply scroll through the word scientific publications (**Tober, Markus, 2011**). The search engine is one of the world's largest electronic bibliographic and full-text collections on science, technology and medicine.

Library users have not been sufficiently trained in the art of taking full advantage of the library. As a result, users of the library were known to leave the library frustrated. A research project was conducted to develop a reliable, effective and efficient library document retrieval system to satisfy library users (**Chimah and Unagha, 2010**).

Many users are interested in gathering useful information about Twitter for future use. In this respect, the user needs a system that makes it easier for users to restore tweets and regain them with a higher degree of relevancy with the user's request. A research project has been conducted that acquires information from Twitter using the Twitter Research API. It develops a corpus of user content by removing noisy and ambiguous elements from a carefully selected set of tweets. As well, it allows users to ask questions in order to obtain system results (**W. Ahmad and R. Ali 2016**).

2.2.2. Document Categorization

Document categorization can be done using classification or clustering approaches. Some of the research that has been carried out with various clustering algorithms are mentioned below.

Research was conducted on the categorization of rules-based issues in the management of Information Technology (IT) services for support tickets . The problem of classification of tickets is one category of problems of classification of documents in information science. Since problems can occur at any level of the software and hardware stack, an important activity to help proactively manage problems is the creation of leading indicators (a.k.a., signatures or failure codes). Then it can be used to classify incidents into groups for analyzing the root cause and monitoring of failure trends. (**Diao, Y., Jamjoom, H. and Loewenstern, D., 2009**)

Here are a few examples of incident categories:

- Unavailability of the application,
- Exceeded disk usage limit,
- System unavailability,
- Printing error
- Password change.

Above categories are typically identified through the analysis of incident files (i.e. problem tickets). After analysis, it records incident information, including customer name, platform, problem details, severity code, resolution methods, and various time stamps. In this paper, they proposed a rules-based approach to provide a scalable yet effective methodology to support proactive identification of issues in a global service delivery environment. The primary means by which this method of categorization is applied to the classification of tickets is through the utilization of straightforward IF-THEN rules (The IF-THEN rule is when a hypothesis is followed by a conclusion. For instance: On the off chance that the page that client mentions isn't found, show the client the blunder message "Page you're searching for is right now inaccessible."). These rules are checked to see how good the statistics are, and if they're bad, the rules are changed to get better results. The disadvantage of such an approach is that it is a supervised learning approach. Therefore, knowing about the domain is important.

Research conducted on document grouping based on K-Means text mining algorithm (When there is unlabeled data, a type of unsupervised learning technique called K-Means clustering is used.) using Euclidean distance Similarity (A method for measuring similarity that uses the distance between two things to determine how similar they are.) on 20 Newsgroup data sets (**Lydia, L. et al. (04 2018)**). It is carried out on the basis of the terms Term Frequency–Inverse Document Frequency (TF-IDF) (A technique for collecting the occurrences of a word in a collection of documents is TF-IDF.) weights, where similar documents are grouped into a single group using the K-Means and the Euclidean distance. The disadvantage of such an approach is that it only does figure out if a document is in a cluster or not. However a document is supposed to cover several topics. Therefore, the output should not be a boolean but a percentage.

Research carried out on the grouping of texts with the help of mixture models on the corpus kannada of CIIL, Mysore. It is composed of texts from various books in various fields such as business, social sciences, aesthetics, natural sciences, etc. A mixture model is a special type of statistical model that views the data as a set of observations from a mixture of different probability distributions (**Saritha, R. C. and Kulkarni, A. (2013)**). Expectation - Maximum algorithm (EM algorithm - In statistical models, the EM model uses looping to estimate the parameters of an assumed probability distribution. These models are based on latent variables that aren't observed.) used to parameter estimation process of the blended model using the maximum probability. This article shows that the mixing patterns are broader and find groups of different sizes and shapes than K-Means. K-Means tells you whether a document is in a cluster or not, whereas mixture models work on the odds that a document is in a cluster. The EM algorithm has the disadvantage that it is slow and may not be a good tracer for large datasets.

Research carried out on “A Novel Indexing Technique for Web Documents using Hierarchical Clustering” (**Gupta, D. Bhatia, K. and Sharma, A. (01 2009)**). The approach used here is that all words from the entire set of documents are identified, each word is assigned a word id and for each document assign the corresponding word ids according to the word it contains. Then Euclidean distance (determining the distance between two things to determine their similarity) and levenshtein distance (fuzzy string-matching algorithm) is calculated and use this value to create the base clusters. By use of the average linkage method the base clusters are merged into a higher-level cluster. So, this approach uses Agglomerative Hierarchical Clustering (AHC) to cluster documents and create indexes. At the outset, the AHC views each object as a singleton cluster. After that, cluster pairs are merged one at a time until all clusters are combined into a single, massive cluster that contains all objects. (**Murtagh, F. and Contreras, P., 2012**)

When it comes to documents, one of the most useful ways to understand the text is to analyze its subjects. The process of learning, recognizing and retrieving these topics from a collection of papers is called topic modeling. The following are the two topic modeling techniques that are used the most. (**Cvitanic, T., Lee, B., Song, H.I., Fu, K. and Rosen, D., 2016**)

- **Latent Semantic Analysis (LSA):** By analyzing representative corpora of natural text, this mathematical approach allows the modeling and simulation method using computer, where it extracts the meaning or the core of the given set of words or even sentences and passages.
- **Latent Dirichlet Allocation (LDA):** It is a Dirichlet distribution-based statistical generative model.

All topic models are based on the same basic hypothesis where,

- Each paper consists of a mixture of topics and
- Each topic consists of a set of words.

Modeling methods for topics are based on the distribution hypothesis, suggesting that similar words happen in similar contexts. However, clustering methods are designed to partition data into coherent groups. Therefore, per-document topic distributions learned by topic models can be used to cluster documents.

Research carried out on document Clustering Using Latent Semantic Indexing (LSI) to Determine Subject Area (**Antai, Roseline. (2011)**). In a documentary collection, noise may be caused by synonyms and polysemy. LSA uses Singular Value Decomposition (SVD) to reduce the document's spatial size (**Baker, K., 2005**). Primarily by mapping a smaller conceptual space that does not have this noise and facilitating the grouping of similar documents. LSA also serves as a Vector Space Model (VSM), however it improves on conventional "Vector Space Modeling". (VSM involves an algebraic model which involves vectors of identifiers as a strategy of text document representation. We'll get into more detail about this later.) by the way it reduces dimensions. Afterwards, the grouping can be made. The absence of a probabilistic approach is one drawback of this strategy.

Research conducted on exploring documents using the LDA subject model for wikipedia articles (**Tong, Zhou & Zhang, Haiyi. (2016)**). Run the data preprocessing, LDA algorithm on the data set and for the output, topic set and proportion of subjects per document will be returned. The method of grouping documents with the LDA Subject Model is to find the highest proportion of subjects and assign the document to that subject. Also to point out documents similar to another document that is interesting, Jensen-Shannon divergence is used (**Menéndez, M.L., Pardo, J.A., Pardo, L. and Pardo, M.C., 1997**).

To capture the context of a word within a document, a semantic and syntactic similarity, embedding of words is useful. Embedding of words is the most common document vocabulary representations methods. Vaguely speaking, they are vectorial representations of a specific word. Word2Vec is one of the most common techniques for learning how to integrate words using shallow neural networks. The model was developed in 2013 by Tomas Mikolov. It can be obtained through two methods (the two involving neural networks): Skip Gram and Common Bag Of Words (CBOW) **(Dhruvil Karani (2018))**.

A language model is fundamentally a probability distribution about words or sequences of words. Practically speaking, a language model gives the probability that a certain sequence of words is "valid". Validity in this context makes no mention of grammatical validity. This indicates that the language model learns to look like how people talk (or, more specifically, write). That is a significant point. There is no magic in a language model (as do other machine learning models, especially deep neural networks). It is "just" a tool for incorporating abundant information in a concise way that can be reused in a non-sampled context. It is possible to use an abstract understanding of natural language for a variety of purposes, including the ability to derive word probabilities from context. Lemmatization (a.k.a. Stemming - The process of stemming or removing the last few characters from a word is known as stemming.) aims to reduce a word to its most basic shape, thus drastically reducing the number of tokens. These algorithms work best if the word is known to play a role. The postfixes of a verb may differ completely from the postfixes of a noun. Hence the justification for marking a part of the word (or POS-tagging), a common task for a language model **(Kapronczay, M. (2021))**.

Researchers decided to go with an improved version of the TF-IDF approach which is Okapi Best Match 25 (Search engines use a ranking function called Okapi BM25 to estimate how relevant documents are to a given search query.) to get similar documents. It may seem a bit strange that here in 2022, I have based on TF-IDF for my thesis which was first formulated long back ago. Key reasons to choose TD-IDF are outlined below.

- 1) After looking at the corpus, which is in a circular domain, the researcher identified most of the circular documents containing one specific topic. Also, most circulars contain one page which is directly talking about the specific

topic. So the researcher decided not to go with topic modeling however use only an improved version of the TF-IDF approach. Here the researcher did a K-Means clustering on the corpus and identified the clustering happens very well when $k=1$ where k is the number of clusters. Which proves the circulars talking only about a specific topic.

- 2) Natural language processing (NLP) has seen many exciting advancements since the 2013 and 2018 breakthroughs of word integration and language models.. In 2018, Google published a text classification framework based on 450K experiences on a couple of different text sets. According to the experiments of 450K, Google found that when the “number of words per document or number of samples” < 1500 , TF-IDF was the best way to represent the text. When you have a small sample size for a fairly common issue, it is helpful to try TF-IDF (**Ann Sebastian (2020)**). Therefore, the researcher decided not to go with word embedding and language models but use only an improved version of the TF-IDF approach.

2.3. Basic Components in an Information Retrieval System

The main aim or the purpose of the IR system is to retrieve documents by indexing them. This will represent the documents that the system will provide users with access to. When a user inputs a request, the system associates the indexed documents with the request and displays the corresponding documents.

2.3.1. Inverted Index

Indexing is an important concept. Creating an index allows you to more quickly find records; without them, you need to go through hundreds or thousands of documents to locate an individual record.

There are two types of indexes. They are forward index and inverted index. For efficient search an inverted Index is preferred over forward Index.

The data structure which is called the forward index stores mappings between documents and words. It directs you to Word from the document. The index data structure which is called the inverted index stores a map of a document's content, such as words or numbers, at its locations in a set of documents. To put it another

way, it is a key-value pair similar to a data structure that takes you from a word document to a webpage or document.

Inverted indexes comes in two different types: **(Baeza-Yates, B. Ribeiro-Neto, et al., 1999)**

- 1) A record-level inverted index contains a list of document references for each word. For example if we take the term “hello” and if that term appears in documents with docIDs 3, 5, 10, 23 and 27, the posting list will contain document frequency of 5 along with docIDs as below.

```
{ "hello" :  
    [5,  
      [3, 5, 10, 23, 27]  
    ]  
}
```

- 2) The inverted index method examines the document word for word and generates an index that includes the word's position. The latter form offers more features, however requires more processing power. In that case, the positions where the term appears in a specific document are also stored at the same time as the docID. For example, if “hello” appears in document 3 at three positions: 120, 125, and 278, we can represent it with the positions and the frequency of the term for each document as below.

```
{ "hello" :  
    [5,  
      [  
        {3 : [3, [120, 125, 278]]},  
        {5 : [1, [28] ] },  
        {10 : [2, [132, 182]]},  
        {23 : [3, [0, 12, 28]]},  
        {27 : [1, [2]]}  
      ]  
    ]  
}
```

We can clearly represent it as figure 2.1.

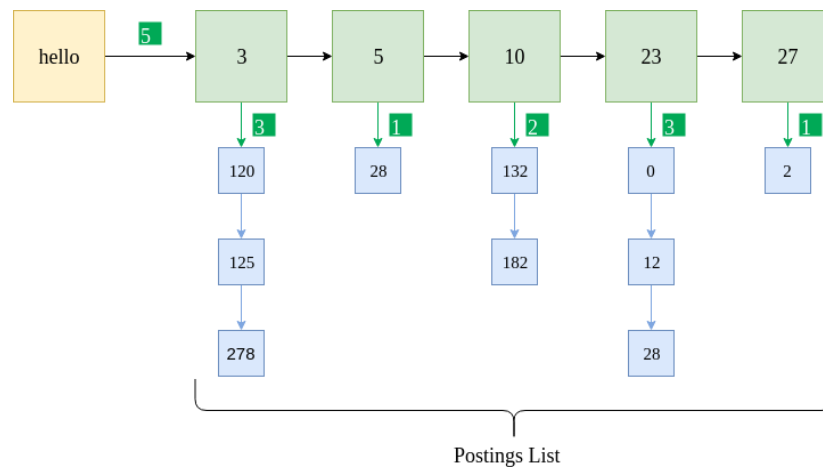


Figure 2.1 - Inverted Index Example

(Source: <https://www.geeksforgeeks.org/python-positional-index>)

A Bag of Words (BoW) array representation needs to be created from raw documents in order to create the index. Document representatives is a common name for these terms. Text is preprocessed in order to produce these representatives for each document.

Pre-processing of text is necessary to transfer human language text to machine-readable format for further processing.

Pretreatment of the document includes:

- Tokenization - tokenization is a process whereby text is taken and broken down into individual words, called tokens. Tokens are used as input to other assignments. The following example illustrates the process. (Kaur, H. and Gupta, V., 2016, August).

The input is: [Blueberries, Raspberries and Moulberries are rich in antioxidants];

The output after tokenization will be: ['Blueberries' , 'Raspberries' , ' and' , 'Moulberries' , 'are' , 'rich' , 'in' , 'antioxidants']

- case folding ignores the case of the query terms and would generally convert them to lower case before comparison.
- Removal of punctuations
- Removal of white spaces

- Removal of stop words
- Stemming - chop a portion of each word so that it might get the "root word". Standard tools are available to do this like "Porter Stemmer".

After that, we count every word's frequency. The word with a frequency greater than a threshold (based on a formula consisting in file size) is chosen as the index term. Collecting all of these words creates our index table (representative document) for this document.

2.3.2. User Query Types

The document set, which includes sentences (a collection of words), the date it was created, the name or names of the author(s), and the type of the document, is paired with a number of keywords during the indexing process. When a user makes a request, the IR system uses these pairings to create an inverted index that is then used to generate results. These indexes are compared to that request in the back-end when a user makes a request. Operators like boolean, conditional statements, and logical statements can be used to narrow down and construct complex requests in IR systems, allowing users to express their search's purpose or requirement in a more stable manner.

From a large data set that has already been created, IR systems provide users with documents that are related to the request they have made. In **2008, Manning, C.D.** stated that search engines can receive user requests—essentially, requirements—that may or may not have a meaning or may even change over time. Most of the time the request says the user requirement in a nut shell. An IR system's querying methods can look like the following.

1) **Keyword Queries:**

To retrieve documents, the user only needs to enter keyword combinations. The logical AND operators connect these keywords. Keyword searches are supported by all retrieval models. This is the simplest querying method.

2) **Boolean Queries:**

The following operators can be used to deal with when the system allows boolean queries.

- The "+" operator is used to individually include required content.
- The "-" operator is used to individually exclude required content.
- The AND operator is used to create requirements queries, which state that the search result must contain the keywords that are combined with this operator.
- The OR operator is used to create queries of requirement, which state that the search result must contain either of the keywords combined with this operator.
- similar to the "-" operator is the NOT operator.

Because a document meets or fails to meet a request like this, there is no matching score involved. An example is provided in Figure 2.2.

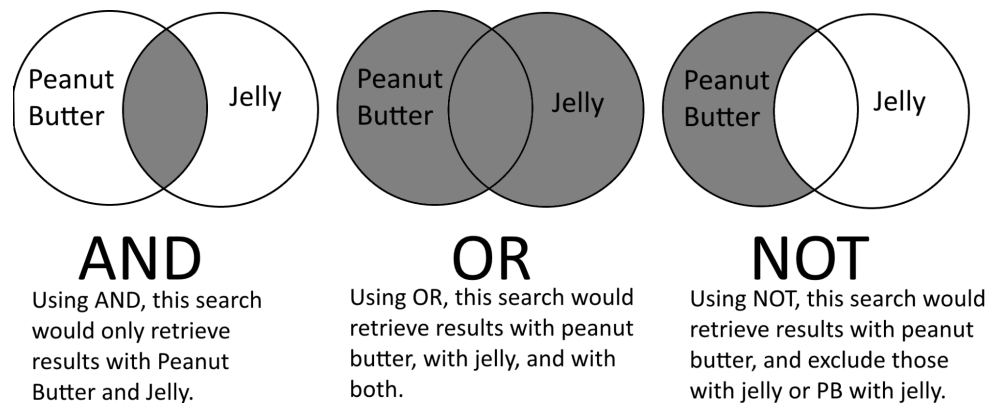


Figure 2.2 -Example of Boolean Queries

(Source: <https://sru.libguides.com/history/librarybasics/booleanoperators>)

When an exact match is found in a document that follows the logic that was used, the document extraction process takes place (**Salton, G., Fox, E.A. and Wu, H., 1983**). The boolean system is relatively simple to implement, however several disadvantages exist. Not all users may be familiar with Boolean queries, and it might be difficult to formulate and optimize the query.

3) Phase Queries:

The relative order of the documents' contents is lost when an inverted keyword index is used to represent them for searching. These phases are either implemented differently or encoded in an inverted index so that precise phase retrieval can be carried out. A phase is the word sequence that makes up this

query. Typically, it is contained within double quotes. (**Gudivada, V.N., Raghavan, V.V., Grosky, W.I. and Kasanagottu, R., 1997**)

4) Proximity Queries:

The search technique known as proximity takes into account how close multiple items in a record ought to be to one another. Phrase searches, which require terms to be arranged precisely, are the most frequently used proximity search option. Terms proximity to one another can be specified by other proximity operators. Some will specify the search terms order.

The names of various operators, such as NEAR, ADJ (adjacent), or AFTER, are used by search engines. These operators are used in proximity queries by search engines.

- NEAR operator - The NEAR operator looks for occurrences of the given word or words within a predetermined number of words of one another, regardless of their order.
- ADJ operator - On real Euclidean space, the ADJ (adjacent) operator mimics the behavior of the flipped version of the given matrix.
- AFTE operator - The AFTER operator is used to compare two keywords.

However, it is more cost-effective for significantly smaller collections of documents than for the web due to the time-consuming document pre-processing required to enable these operators.

5) Wildcard Queries:

Regular expressions and text-based pattern matching searches are supported (**Vechtomova, O., 2009**). Wild card queries are not supported directly by search engines.. Wildcard search support can be implemented in some IR systems. Any of the following scenarios call for the use of wildcard queries:

- I. The user is unsure of how to spell a query term (such as Tokyo vs Tokiyo, which results in the wildcard query Tok*yo)
- II. The user is conscious of the fact that there are multiple spelling variations of a word and searches for documents with any of the spelling variations (for instance, defence vs. defense)

- III. The user is looking for documents that contain variants of a term that would be caught by stemming but is unsure if the search engine uses stemming (for instance, jewelry vs. jewellery, which leads to the wildcard query jewel*)
- IV. The user is unsure of how to spell a foreign word or phrase (such as "Universit* Stuttgart").

6) **Natural Language Queries:**

Without any special syntax or format, a natural language query only contains normal terms in the user's native language. Only a small number of search engines providing natural language processing attempt to comprehend the structure and meaning of queries written in natural language text, typically in the form of questions or narratives. **(Cafarella, M.J. and Etzioni, O., 2005, May)**

From the results it retrieves, the system tries to come up with answers to these questions. This kind of query can be supported by semantic models. **(Strzalkowski, T. ed., 1999)**

2.3.3. **Document Retrieval and Ranking**

- **Boolean Model**

One of the earliest and simplest IR models is the Boolean model. A document is either relevant to a query in the Boolean model or not; there is no ranking or degree of relevance. Classical set theory and Boolean logic are the foundations of the model. When a term is present in a document, its value is either true or one; terms are regarded as Boolean variables, either false or zero, in any case.

- **Vector-Space Model (VSM)**

A term vector is used to represent each document in the VSM. The model does not include a definition for a term; however, terms are typically short phrases, single words, or keywords. The dimension of the vector is the number of words in the vocabulary if words are selected as the terms. In a relatively high-dimensional space, a vector can be used to represent any document as

equation 2.1 below. A query can then be transformed into a vector as well because it is text as equation 2.2 below. The following is a representation for queries and documents.

$$d_j = (w_{1j}, w_{2j}, \dots, w_{ij}) \rightarrow Eq. 2.1$$

$$q = (w_{1,q}, w_{2,q}, \dots, w_{i,q}) \rightarrow Eq. 2.2$$

Where;

w_{ij} - value of the i^{th} term in the vector of the j^{th} document

w_{iq} -value of the i^{th} term in the query vector q .

The cosine value of the angle between the document vectors and the query vector is used to rank the documents during the data retrieval time. The terms may be weighted according to their frequency, TF-IDF score, etc. The cosine of the angle is calculated as the ratio of the inner product of the document and query vectors and the product of the norm of the document vector and the norm of the query vector for each document and query. The system then returns the documents by decreasing the cosine (**Melucci, 2009**).

- **Bag of Words Assumption**

The "bag of words assumption" (also known as the "bag of words model") is a crucial assumption that is essential to not only topic models but also many other text mining models. The assumption known as the "bag of words" (BoW) views each document as a collection of words, ignoring grammar and even word order in favor of multiplicity. Tokenization is another name for this process, which converts the documents into words. The document is represented by a set of words that are extracted. The word order and grammar are ignored by this method. A sorted list of words that best describe the document is what we get. In document classification, the BoW assumption is frequently utilized. (**Zhao, R. and Mao, K., 2017**)

What kind of document does the assumption of the BoW represent? How does it work? Let's use a very simple example to get a better understanding about this. Here are two very brief documents that this model represents. We use two extremely brief documents with only one sentence to simplify the issue.

- **Document 1:** I like Python programming language.
- **Document 2:** My friends prefer Java programming language over Python programming language.

The extraction of vocabulary for the documents is the first step. A vocabulary is a collection of distinct words from the documents. A vocabulary list can be made as follows:

[*"I"* , *"like"* , *"Python"* , *"programming"* , *"language"* , *"My"* , *"friends"* , *"prefer"* , *"Java"* , *"over"*]

All documents are based on vocabulary, which is a vector. To determine whether a word is present in the vocabulary, we must denote 0 or the frequency of the word next to it in order to interpret each document. In the BoW assumption, the following is how we represent these two documents:

- **Document 1:** [1,1,1,1,1,0,0,0,0,0]
- **Document 2:** [0,0,1,2,2,1,1,1,1,1]

Both Document 1 and Document 2 shows the frequency of the words in the vocabulary list prepared. Here 0 (Zero) denotes the unavailability of the specific word in that specific document. If the value shown here is greater than 0, that means the word is available in that document. From this value we can tell the frequency of that word in the document. Each word in the vocabulary is represented using this model in the same way as a histogram. However, this model does not preserve the word order of the original document which means we cannot get the original content from the BoW assumption. The frequency of the words in each document is used to weight them. It is also known as a Document-Term Matrix (DTM). If we transform the collection of documents into a matrix, it can be represented as in table 2.1.

Table 2.1 - Document-Term Matrix (DTM)

	<i>I</i>	<i>like</i>	<i>Python</i>	<i>programming</i>	<i>language</i>	<i>My</i>	<i>friends</i>	<i>prefer</i>	<i>Java</i>	<i>over</i>
Document 1	1	1	1	1	1	0	0	0	0	0
Document 2	0	0	1	2	2	1	1	1	1	1

- **Term Frequency–Inverse Document Frequency (TF-IDF)**

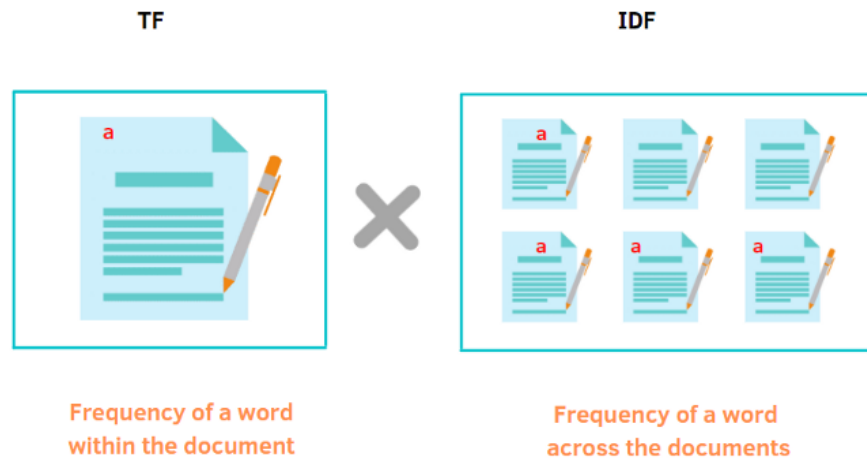


Figure 2.3 - Graphical Representation of Term Frequency-Inverse Document frequency

(Source:

<https://medium.com/codex/document-indexing-using-tf-idf-189afd04a9fc>)

A numerical statistic called TF-IDF is used to show how important a word is to the document. Term frequency is the ratio of the total number of words in a document to the number of times a word appears in the document. The calculation is shown in equation 2.3 below.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \rightarrow Eq. 2.3$$

The weight of unique words across all documents is determined by utilizing inverse document frequency. The calculation is shown in equation 2.4. Rare words have a high IDF score.

$$idf(w) = \log\left(\frac{N}{df_i}\right) \rightarrow Eq. 2.4$$

The two equations above must be combined in order to calculate the TF-IDF score. This combining is shown in Equation 2.5 below.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \rightarrow Eq. 2.5$$

$tf_{i,j}$ = number of occurrences of i in j

df_i = number of documents containing i

N = total number of documents

- **Best Match 25 (BM25)**

Search engines use a ranking function called BM25, to estimate how relevant documents are to a given search query. The BM25 function is thought to be superior to TF-IDF. The structure of BM25 is identical to that of the TF * IDF formula (As in equation 2.6 and equation 2.7); however, the values of the TF and IDF components will be substituted with improvements to those values. In conclusion, simple TF-IDF rewards frequency of terms and penalizes frequency of documents. To account for document length and term frequency saturation, BM25 goes above and beyond (**Seitz, 2020**).

$$BM25(D, Q) = \sum_{i=1}^n IDF(q_i, D) \frac{f(q_i, D) \cdot (k+1)}{f(q_i, D) + k \cdot \left(1 - b + b \cdot \left(\frac{|D|}{d_{avg}}\right)\right)} \rightarrow Eq. 2.6$$

$$IDF(q_i) = \log \frac{N}{n} \rightarrow Eq. 2.7$$

Where:

- $f(q_i, D)$ is the number of times term q_i occurs in document D
- $|D|$ is the number of words in document D
- d_{avg} is the average of words per document
- b and k are hyperparameters for BM25

2.4. Recommender System

One of the most rapidly expanding subfields of Artificial Intelligence (AI), recommendation systems are now a regular part of our lives. Every aspect of our lives is being affected in some way by recommender systems, whether it's personalized ads, search query results, or product bundle recommendations. (**Zhang, Q., Lu, J. and Jin, Y., 2021**)

A Recommender Systems objective is to make useful recommendations to a group of users for items or products that might be of interest to them. Real-world examples of the recommender systems that are in use in the industry include recommendations for movies on Netflix (**Amatriain, X. and Basilico, J., 2015**) and books on Amazon (**Smith, B. and Linden, G., 2017.**).

There are two main types of recommender systems, according to the methods used to prepare suggestions. (**Khatwani, S. and Chandak, M.B., 2016**).

- Non-personalized recommender system - a general recommender system that uses the opinions and feedback of other users to make recommendations.
- Personalized recommender system - maintains a user profile and makes an effort to match products to a user's preferences before recommending them to the user.

The recommender system's most intriguing component is the generation of suggestions based on user ratings and behavior. The following is a summary of some of the most prevalent strategies.

2.4.1. Content Based Recommenders

This family of recommenders, as the name suggests, constructs a recommender system by incorporating the user profile and some form of content, such as reviews or descriptions (**Lops, P., Gemmis, M.D. and Semeraro, G., 2011**).

The most important concepts are as follows:

- Model items based on relevant content-derived attributes,
- Create a user profile using either these implicit actions (clicks, video time spent, etc.) (**Claypool, M., Le, P., Wased, M. and Brown, D., 2001**), explicit actions (like rating, purchase, etc.) or by combining explicit and implicit methods, and finally,
- Utilize these profiles to make suggestions.

One example of content-based recommendations is personalized recommenders. A personalized recommendation system examines user data, including their ratings, purchases, and relationships with other users. That way, individual recommendations will be provided to each user.

Let's try to comprehend this with the figure 2.4 (Nigam, 2022).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
1		baseball	economy	politics	Europe	Asia	soccer	war	security	shopping	family		num-attr		User 1	User 2		Pred1	Pred2	
2	doc1	0.44721	0	0.44721	0	0.44721	0.44721	0	0	0	0.44721		5		1	-1		0.247612	-0.217167	
3	doc2	0	0.5	0.5	0.5	0	0	0	0.5	0	0		4		-1	1		-0.136187	0.329154	
4	doc3	0	0	0	0.57735	0.57735	0.57735	0	0	0	0		3					0.109459	-0.062892	
5	doc4	0	0	0.5	0.5	0	0	0	0.5	0.5	0		4			1		-0.089197	0.240296	
6	doc5	0	0.57735	0	0	0	0	0	0	0.57735	0.57735		3					-0.043527	0.044585	
7	doc6	0.70711	0	0	0.70711	0	0	0	0	0	0		2		1			0.319432	-0.084695	
8	doc7	0	0	0	0	0	0	0	0.70711	0	0.70711		2					-0.058926	0.113531	
9	doc8	0	0	0.5	0.5	0	0	0.5	0	0	0.5		4					-0.04753	0.070575	
10	doc9	0	0	0	0	0	0.70711	0	0	0.70711	0		2					0.179067	-0.120746	
11	doc10	0	0.57735	0	0	0.57735	0	0.57735	0	0	0		3					-0.128031	0.046812	
12	doc11	0	0	0.57735	0	0.57735	0	0	0	0.57735	0		3					0.018752	0.01775	
13	doc12	0.57735	0	0	0	0	0.57735	0.57735	0	0	0		3			-1		0.311648	-0.252852	
14	doc13	0	0	0.5	0.5	0.5	0	0	0.5	0	0		4					-0.057253	0.208553	
15	doc14	0	0.5	0.5	0.5	0	0	0	0	0.5	0		4					-0.053281	0.204154	
16	doc15	0	0	0	0.5	0	0.5	0.5	0.5	0	0		4					0.021184	0.102276	
17	doc16	0.57735	0	0	0	0	0.57735	0	0	0.57735	0		3		1			0.396153	-0.246472	
18	doc17	0	0.5	0.5	0.5	0	0	0	0.5	0	0		4			1		-0.136187	0.329154	
19	doc18	0	0	0	0.70711	0	0	0	0	0.70711	0		2					0.071635	0.096424	
20	doc19	0	0.44721	0.44721	0	0.44721	0	0.44721	0	0	0.44721		5		-1			-0.121533	0.043343	
21	doc20	0	0	0.5	0.5	0	0	0.5	0	0.5	0		4					-0.006291	0.115296	
22																				
23	DF	4	6	10	11	6	6	7	6	7	5									
24																				
25																				
26	User Profiles																			
27	User1	1.73167	-0.94721	-0.5	0.20711	0	1.02456	-0.44721	-0.5	0.57735	0									
28	User2	-1.02456	1	1.05279	1.5	-0.44721	-1.02456	-0.07735	1.5	0	-0.44721									

Figure 2.4 - Personalized Example of Recommendations

(Source:

<https://medium.com/analytics-vidhya/understanding-recommender-systems-introduction-3be54e937625>)

The columns in the preceding example represent attributes (the topic of the article, such as politics, sports, etc.). And the article is represented by the rows. A weighted value for that article's topic can be found in each cell. The user's opinion of a particular article can be found in the two columns user1 and user2.

As a dot product of the user vector and the attribute matrix, user profiles are projections of the user's taste in attribute space. For instance, the dot product of the user vector (O2:O21) and the baseball vector (B2:B21) yields the value in cell B27, which indicates the overall significance of baseball to user1.

After that, a dot product is calculated between the user profile and the document profile to determine the prediction values (pred1 and pred2). For example, a dot product between user1 (B27:K27) and doc1 (B2:K2) yields the value in R2.

The likelihood that the user will like that particular document increases with the prediction value. It's like the more the user shows a likeness towards the attribute, the more he/she can reach to the required content via the suggestions of the recommendation.

2.4.2. Collaborative Filtering

Collaborative filtering can be approached in one of two ways:

1. Memory-based (a.k.a. neighborhood-based) methods **(Yu, K., Schwaighofer, A., Tresp, V., Xu, X. and Kriegel, H.P., 2004)**. They themselves come in two varieties:

- a. User-based collaborative filtering - This filtering identifies customers with similar preferences and makes recommendations to one customer based on a set of new items that are also preferred by other customers with a similar profile.

User-based collaborative filtering strategy is based on the idea that people who watch the same kinds of movies also like the same things. It identifies movies that similar users have watched, however the first user has not, then it identifies movies that similar users have not watched. Based on that, it makes recommendations.

If all travelers could share data in a common location like a website or even social media, this strategy could also be utilized for global tourism industry promotion. **(Jia, Z., Yang, Y., Gao, W. and Chen, X., 2015, February)**

- b. Item-based filtering - This filtering makes recommendations based on a domain-specific knowledge of the content of an item that is similar to those a customer has already purchased. **(Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., 2001, April)**

A movie recommender system would suggest movies with similar characteristics if a user liked movie A with item-based collaborative filtering. These characteristics might include the genre, producer, leading actors, run time, and release date, among other things. **(Kumar, M., Yadav, D.K., Singh, A. and Gupta, V.K., 2015)**

2. Model-based methods - Methods based on models go one step further than collaborative filtering and make use of probabilistic models and machine learning. **(Aggarwal, C.C., 2016)**

2.4.3. Knowledge-based Recommender Systems

Similar to content-based recommender systems, knowledge-based recommender systems use a customized model to identify relevant products for each user. However, rather than constructing that model from the user's rating history, they ask the user to explicitly state their preferences—perhaps through keywords or a series of requirements—in order to build that model. When a user may not immediately know what they want, they are useful in a cold start scenario or in a complex item domain with many attributes (**Burke, R., 2000**).

For items that are rarely purchased, knowledge-based recommenders are utilized. It is simply impossible to recommend such products based on a user's profile or previous purchases. The acquisition of real estate is an excellent illustration of how these kinds of recommender systems can be used.

2.4.4. Hybrid Recommender Systems

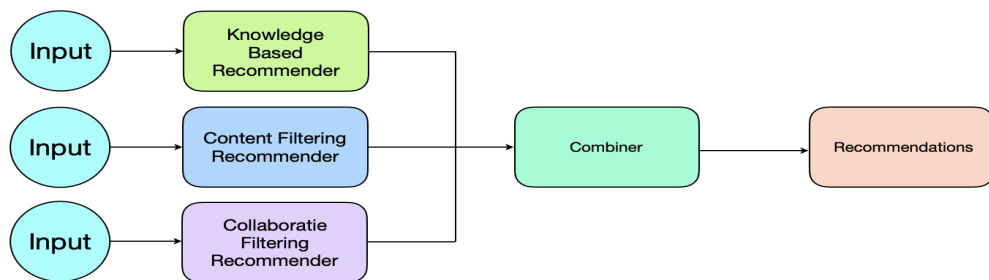


Figure 2.5 - Major Functionalities of a Knowledge-based Recommender System

(Source: <https://goo.by/mcGdy>)

Rarely, if ever, are these fundamental types of recommender systems utilized on their own (As in figure 2.5). The majority of practical recommender systems are hybrid models or combinations of models. These hybrid models can be as simple as taking the weighted average of several models or as complicated as enormous monolithic systems that blur model boundaries. Due to the fact that many of the best recommender systems derive their edge from the way they utilize models collectively, this is an especially rich and useful subject to investigate. (**Burke, R., 2002**).

2.5. Document similarity measures

In a dataset, the term "similarity measure" refers to the distance with dimensions that represent the features of the data object. When this similarity goes lower and lower it provides similarities of higher degrees, however there will be a low degree of similarity if it is large. (Huang, A., 2008, April)

Popular similarity measures include:

- 1) Cosine Similarity,
- 2) Euclidean Distance,
- 3) Jaccard Similarity,
- 4) Manhattan Distance,
- 5) Minkowski Distance and many more.

- **Cosine Similarity**

Cosine similarity can be used in the recommendation system to determine how similar two documents are. Cosine angle between two vectors can be used to identify whether two vectors are closer to each. Since the cosine value is between -1 and 1, if two vectors make an angle of 0, their cosine value would be 1, indicating that the documents are closely related. The sentences would be almost unrelated if the two vectors were orthogonal, or $\cos 90^\circ$ (Xia, P., Zhang, L. and Li, F., 2015). So, by comparing the documents with the user profile, we can recommend users with the circulars that scores are closer to 1. The similarities between two documents which are translated as vectors X and Y using TF-IDF can be calculated by using cosine-similarity formula (Eq. 2.8):

$$\text{Cosine}(X, Y) = \frac{X \cdot Y}{\|X\| \cdot \|Y\|} \rightarrow \text{Eq. 2.8}$$

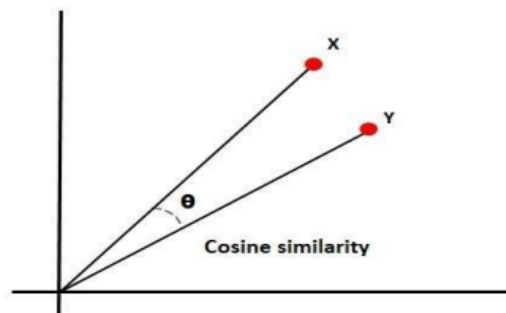


Figure 2.6 - Graphical Representation of Cosine Similarity Calculation

Cosine can be used to identify text document similarities using BoW representations, word vector representations, or TF-IDF values. **(Buck, C. and Koehn, P., 2016, August)**

CHAPTER 3

METHODOLOGY

Architecture of the Proposed System

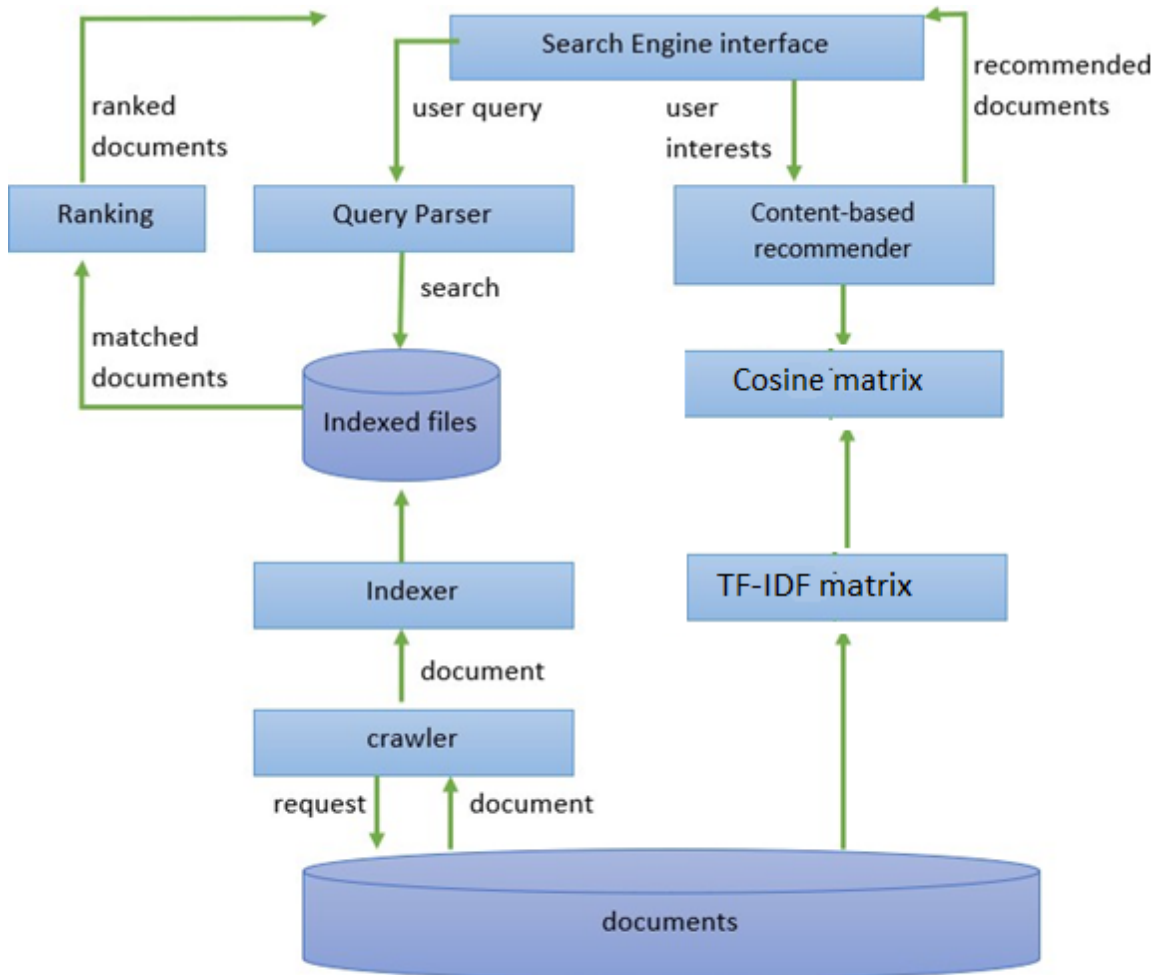


Figure 3.1 - Architecture of the Proposed circular management and recommendation system

3.1. Data Retrieving and Pre-processing

3.1.1. Data Retrieving

The dataset which is going to be used for this research project is downloaded from the Uniform Resource Locator (URL) which is mentioned in the “Introduction” chapter. Even the circulars are in Sinhala, Tamil and English languages this project is implemented only for circulars which are in English language.

The circulars are stored in Portable Document Format (PDF). To process the PDF the first step is to extract text from it. There are many libraries available freely for working with PDFs. They are, PDFMiner, PDFQuery, Tabula.py, Xpdf, pdflib, Slate, PyPDF2,

etc. For this research, to extract text from PDF PDFMiner library (<https://pypi.org/project/pdfminer/>) is used as it gives more accuracy. Uploading circulars to the system is done by admin. There is a separate login for admin users. Once he is logged in he can upload circulars by specifying the tags which that circular refers to. In the system when a user uploads a circular it will be stored in a folder called “incoming_data”.

3.1.2. Bulk Uploading

When there are many circulars to upload it is very difficult to upload one by one from the user interface. To overcome this researcher has implemented a bulk upload facility. The steps in bulk uploading are specified below.

1. Add circulars to the “massUpload” folder.
2. Run the “massUploader.py” script.
3. Foreach file in the ‘massUpload’ folder, it will show the filename and prompt to enter tags which circular refers to.

3.1.3. Adding Tags, Filter with Tags Which PDF belongs to

Tags which PDF belongs to can be added when uploading circulars to the system. When a user is trying to add tags, a drop-down will pop up and suggest previously added tags. This suggested tags list will refresh with every keystroke user enters. Tags which are not there can be added by simply typing the tag name and pressing the”tab” key.

Those tags user added are stored as a simple JavaScript Object Notation (JSON) file. Create and search operations on this JSON file are handled by a python library called “tinyDB”. TinyDB provides a simple way to perform record level transactions with a JSON file. TinyDB acts as a document-oriented database and it is a very powerful library.

Assume two categories start with the same first 3 characters, it will be stored in the same object.

For example, if we take two tags as salary and salary increment, see the first three characters are common for both tags. Therefore, under ‘sal’, both categories are saved.

```

{"sal" : {
    "salary", "salary increment"
}}

```

By doing so it is very easy to provide tags suggestions to the drop-down. All we have to do is to provide the typed letters to the server via Application Programming Interface (API) and retrieve data using JavaScript. Whenever tags are added by admin by uploading a circular, or by bulk uploading method the application will check if a text document named "documentcategory.txt" exists. If that text file does not exist, the app will create a text file named "documentcategory.txt" and insert a record. The inserted record will contain the pdf name, tags which were added for that pdf and pdf size as shown below.

Computer Test Relevant to the Efficiency bar Examination for Officers-1666172634.5892184.txt	"computer test,efficiency bar examination,officers"	217.5
Computer Test Relevant to the Efficiency bar Examination for Officers - 2017-1666172670.1330814.txt	"computer test,efficiency bar examination,officers"	152.4
Deferment of Salary Increments on non-completion of Efficiency Bar Examination-1666172799.0288794.txt	"deferment,salary increments,non-completion,efficiency bar examination,officers"	241.4
Deferment of salary increments on not completing the efficiency bar-1666172822.5231614.txt	"deferment,salary increments,non-completion,efficiency bar examination,officers"	190.0
Reference the course related to the exemption of the officers from the Efficiency Bar Examination-1666172908.0028272.txt	"exemption of officers,efficiency bar examination,course related"	209.3
Agrahara Insurance Scheme for Public Officers-1666172934.8670797.txt	"agrahara insurance,insurance schemes,public officers"	12.7
Extending the benefits under Agrahara Insurance Scheme-1666172969.822767.txt	"benefits,extension of benefits,agrahara insurance,insurance schemes"	66.5

3.1.4. Pre-processing

One of the most significant aspects of this investigation is text pre-processing, also known as text cleaning. For human comprehension, circulars are written in natural language. That data, on the other hand, isn't always simple for computers to process in text mining. We always reduce the size of the text data by removing redundant data like stop words or numbers to speed up the computation. Although careful, laborious, and time-consuming, pre-processing will ultimately yield high-quality data that will fit the

model and produce better results. The steps in preprocessing that might be necessary for any kind of text mining algorithm are as follows:

- **Removal of punctuations and white spaces**

Since computers can't actually read, punctuation and white space are helpful for humans to understand when it comes to grammar. However, computers see them as identical to other characters. Therefore, in order to simplify the document, we can actually eliminate the white spaces and punctuation.

- **Tokenization**

After removing all punctuation, a document is divided into a list of tokens and treated as a string. The documents will become unreadable if punctuation and white space are removed. However, the main ideas or topics are still present, so you can tell what they're trying to say. It is predicated on the supposition that the document is a jumble of words. A word from the topics serves as the basic unit.

- **Conversion to lower cases**

The probabilistic model can benefit from knowing the frequencies of each word, so this step is necessary. Because it is the first word in the sentence, some letters in the documents will be in uppercase. The computer will interpret "data" and "Data" (See the distinction in first letter in quite a while. One is written in uppercase and one in lowercase.) as distinct terms as a result. We can simply change all of the letters to lowercase to avoid this situation.

- **Removal of stopwords**

Stopwords are words that are used to make the sentence easier to read but have no real meaning. For instance, the words "the," "and," and "if" are required in the sentence, but their meaning does not change if they are removed. The majority of English documents use stop words fairly frequently. Although the meaning will not be altered by the removal of the stopwords, it may significantly reduce the document's size and improve the quality of text mining. Stopwords have been pre-defined in Python's libraries. If necessary, we can always examine the situation and decide whether to add or remove some of the stopwords.

- **Stemming**

Although they are expressed in different participles or plurals, some of the words convey the same meaning. For instance, "learned" and "learning," "cat" and "cats," and "joy" and "joyfulness" are all synonyms. Humans can recognize it more easily, but computers need us to stem the words. Stemming is the process of getting a word back to its original form. Additionally, this procedure may improve the quality of text mining while also reducing the amount of time required for computation. To get circulars ready for indexing, the above mentioned steps are applied to the circular data set.

3.1.5. Crawling and Indexing

Search engines work on the concept of Crawling and Indexing.

- Crawling - looks for pages that are new or updated.

User uploaded circulars need to be crawled first before creating an inverted index. In the app all uploaded PDFs are stored in the “incoming_data” folder. Whenever the crawler method runs it will pick all pdf which has not crawled before. The PDFs to be indexed are identified by using the “doclist.txt” file which gets created after indexing. “doclist.txt” file contains all PDFs which were indexed before with their last modified time. So the crawler method can identify what are newly uploaded documents or modified documents by going through “doclist.txt”.

- Indexing - circulars captured from crawling are indexed and stored to be returned later for a search query.

Indexing will happen after the above mentioned pre-processing steps. After preprocessing, the stemmed word list will be returned for the PDF which is given as an input. By using this stemmed list three dictionaries are maintained for the ease of data retrieval. They are,

- **doclist** : This dictionary stores the number of words, tags, year, size and last modified time of the circular corresponding to its name.

Computer Test Relevant to the Efficiency bar Examination for Officers-1666172634.5892184.txt "computer test,efficiency bar

```
examination,officers" 217.5
```

```
Computer Test Relevant to the Efficiency bar Examination for Officers -  
2017-1666172670.1330814.txt "computer test,efficiency bar  
examination,officers" 152.4
```

```
Deferment of Salary Increments on non-completion of Efficiency Bar  
Examination-1666172799.0288794.txt "deferment,salary  
increments,non-completion,efficiency bar examination,officers" 241.4
```

- **wordfile** : This dictionary stores how many documents containing the stemmed word. And also it contains the number of occurrences of a stemmed word with its circular name foreach file the stemmed word contains.

```
"common": {  
    "predoc": 1,  
    "ScrapedPDFs/Deferment of Salary Increments on non-completion of  
    Efficiency Bar Examination-1666172799.0288794.txt": 1  
},  
"law": {  
    "predoc": 1,  
    "ScrapedPDFs/Deferment of Salary Increments on non-completion of  
    Efficiency Bar Examination-1666172799.0288794.txt": 1  
},  
"order": {  
    "predoc": 1,  
    "ScrapedPDFs/Deferment of Salary Increments on non-completion of  
    Efficiency Bar Examination-1666172799.0288794.txt": 1  
},
```

- **wordloclist** : This dictionary stores all the locations of a stemmed word in a file. The location includes the line number and position of the word in that line. This will help to show a small description to the user where the search query exists in the document for matching search results.

```
"combin": {  
    "ScrapedPDFs/Computer Test Relevant to the Efficiency bar  
    Examination for Officers-1666172634.5892184.txt": [  
        [ 1, 34 ],  
        [ 61, 56 ],  
    ]  
}
```

Whenever a user query is submitted scanning is happening only through those dictionaries but not the whole circular documents. This will increase the speed of user search because user search does not need to scan through large text which are in circular PDFs.

3.1.6. Searching and Ranking

System will allow the following user query types.

- **keyword queries** - For finding keywords, a wordlist dictionary file created earlier is useful.

```
keyword-Matcher(userquery, year, tag):
words = preprocess(userquery):
  for word in words:

    // check if user entered word exists in wordlist dictionary
    // which created earlier in indexing step

    if word in self.wordlist:

      // if exists, from wordloclist dictionary get the circular names with their positions

      pos = self.wordloclist[word]

      // add each document word position in doclist variable

      for docs in pos:

        // check if document exist in doclist variable,
        // if document exists append word and its position for that document

        if docs in doclist:
          if word in doclist[docs]:
            continue
          else:
            doclist[docs][word] = pos[docs]
          else:

            // if document not exists in doclist according to the filtering criteria
            // add document to doclist variable with word and its position for that document

            if year and category:

              // if matches both year and category

              doclist[docs] = {}
              doclist[docs][word] = pos[docs]

            elif year:
              // if matches only year
```

```

doclist[docs] = {}
doclist[docs][word] = pos[docs]

elif category:

    // if matches for category

    doclist[docs] = {}
    doclist[docs][word] = pos[docs]
else:

    // if filtering criteria not added add document

doclist[docs] = {}
doclist[docs][word] = pos[docs]

```

- **phrase queries** - For finding phrases, wordloclist dictionary files created earlier are used. As this file stores the locations of the words this scenario implementation became possible.

```

Phrasequery-Matcher(userquery, year, tag):
    words = preprocess(userquery)

    // to add documents which matches filter criteria for every keyword in phrase

    wordll=[]
    for word in words:

        // check if user entered word exists in wordlist dictionary which created earlier in
        // indexing step

        if word in self.wordlist:

            // to store documents which matches filter criteria for only keyword

            filteredDoc = []

            // store each document which contains that word into variable app

            app = [x for x in self.wordlist[word]]

            // for each document which contains the keyword

            for doc in app:
                if year and category:

                    // if matches both year and category add to filteredDoc

                    filteredDoc.append(doc)
            elif year:

```

```

        // if matches both year

        filteredDoc.append(doc)
    elif category:

        // if matches category

        filteredDoc.append(doc)
    else:

        // if filtering criteria not added add document

        filteredDoc.append(doc)
    wordll.append(filteredDoc)

// finaldocs is derived by intersecting lists in wordll. only if phrase exists in all the
lists that
// document will be added to finaldoc
finaldocs = self.intersectlists(wordll)

// store each word positions of each document in doclist variable

for word in words:
    pos = self.wordloclist[word]
    for docs in finaldocs:
        doclist[docs][word] = pos[docs]
for docs in doclist:

    // the below code is used to check foreach document, the words occurs
    // in the way user entered
    // counter variable is used to deduct counter value from current word
    // position and identify the keywords occur nearby

    counter = 0
    dummy = []
    for word in words:
        for z in doclist[docs][word]:
            locationdict[z[0]] = z[1]
            z[0] = z[0] - counter

        // in a dummy variable the current word position stores. because we have
deducted
        // counter value if the order of document words matches for user phrase,

        // z[0] value has same value

        dummy.append([z[0] for z in doclist[docs][word]])
        counter += 1

    // resultant is derived by intersecting lists in dummy. only if word position
matches
    // in all the lists that word position will be added to resultant

    resultant = self.intersectlists(dummy)

```


For ranking the matched documents, the Okapi BM25 formula is used. PDF documents can be retrieved efficiently using algorithms like BM25 over TF-IDF.

The implementation of Okapi BM25 algorithm to rank phrase query matching documents are as follows.

```
OKAPI-BM25(doclist, k=1.2, b=0.75, delta=1.0):

    // get the no of document in corpus and average no of words for a document
    avgdl, n = self.total_length()

    // create a counts dictionary with size doclist length
    counts = dict([(docid, 0) for docid in doclist])

    // no of documents which contains the matching phrase
    val = len(doclist)

    for docs in doclist:

        // get the document length from doclist dictionary which created in indexing step
        doc_length = int(self.doclist[docs]['wordcount'])

        // variable initialize which needs for OKAPI-BM25

        numerator = (k + 1) * doclist[docs]
        denominator = doclist[docs] + k * (1 - b + b * (doc_length / avgdl))

        idf = log2(n/val)

        // calculate the score value

        score = idf * (delta + numerator / denominator)
        counts[docs] = score

    // keep the score between 0-1 range by dividing the max score value

    ncounts = self.normalizescores(counts)
    return ncounts
```

3.1.7. Recommendation System

Since the data set contains only item data (in our context it is circulars), we would focus on creating a prediction system which is content - based. The system which is built predicts a circular considering the time the user spent on reading (which is calculated by the time the user focuses on the browser page) a particular circular. Therefore, when

recommending circulars, a high recommendation score will be given to the similar circulars which the user has spent a long time reading in the past.

To create the recommendation system TF-IDF matrix and cosine similarity matrix were used. To create a TF-IDF matrix, TfidfVectorizer function is used from sklearn feature_extraction text package. This function accepts several parameters and those are needed when customizing the output. To get more information on this function go to (<https://goo.by/vgyXh>)

It is used to represent a document corpus as its TF-IDF matrix. When calling the TfidfVectorizer function we can pass several parameters to customize the TF-IDF matrix output (**Chaudhary, 2020**). Cosine similarity matrix is used to get the document similarity. To create this matrix, “cosine_similarity” function is used from the sklearn metrics pairwise package (<https://goo.by/oVnSG>). To create a cosine similarity matrix we have passed our TF-IDF matrix as the input for the cosine_similarity function.

The algorithm for recommendation generation is below.

```
GET-RECOMMENDATIONS(viewedDoclist, int matchingMinPercent, int
matchingMaxPercent):

    // get all documents name, document texts as lists in corpus
    documents, train_texts = load_data(crawlPath)

    // get the tf-idf matrix by passing train texts
    tfidf_mtx = tfidf_vector(train_texts)

    // create cos_sim matrix
    cos_sim = cosine_similarity(tfidf_mtx , tfidf_mtx )

    // store the viewed document ids
    idx = [document.id for document in documents]

    // loop through each viewed document id and get the cosine similarity list for that
    // document and store in sim_scores_lst

    for i in range(len(idx)):
        simi_scores = list ( enumerate (cos_sim [idx[i]]))
        for i in rng ( len (simi_scores)):
            sim_scores_lst.append(simi_scores [i])

    # the articles are sorted by the sim scores in desc
    sim_scores = sim_scores_lst sort in desc
```

```

# store scores for very simi articles, giving start index as length of viewed
# documents because no need to show viewed documents in recommendation list
sim_scores = sim_scores_lst[len(documents):len(sim_scores_lst)]
# get only the similarity score between matchingMinPercent and matchingMaxPercent
# which user entered
sim_scores = [sim_score for sim_score in sim_scores if minVal <= sim_score[1] <= maxVal

// return the top 5 most similar documents to front end

lstRecommendations= []
for element in sim_scores:
    if(i==5):
        break
    lstRecommendations.Add(element)
return lstRecommendations

```

3.1.8. Presentation of Circulars

Presentation of the results is done by a web app which is built using python Jinja templates. List of circulars which are relevant to the search terms are generated by the algorithms in the back-end. The results returned from the backend can be mainly divided into two sections. They are,

1. **Query result content** – user query result data directly served here. Result is generated to the browser by parsing searched query parameters to the backend. Jinja templates handle the creation of custom web pages.
2. **Personalized content** – circulars are recommended based on user past activities served here. Once the web page is loaded, a JavaScript fetch request is made to the server endpoint (acting as an API) with user past activity (time spent by user on reading PDF). Back-end server returns matching circulars by considering past user activity in JSON format and JavaScript rendering them as HyperText Markup Language (HTML) in the web page.

Users can view the matching PDFs by clicking on them. If the circular is viewed directly as a PDF there is no way to track time users are actively engaged in reading the PDF. This issue is addressed by opening the PDF as an embedded object in a HTML page so the time spent on the page can be tracked. Simple and lightweight JavaScript library called TimeMe.js is used in the front-end to capture the time that the user spent actively on the PDF. The PDF can be shared via email, instant messaging (such as

WhatsApp), social media (such as Facebook and Instagram), or instant messaging (such as WhatsApp).

CHAPTER 4

EVALUATION AND RESULTS

4.1. Introduction

Current curricula repository allows the user to search the required curricular by,

1. keywords (required)
2. curricular number (optional)
3. year (optional)
4. curricular type (optional)

Curricula repository contains a large number of documents. There is no way to omit unnecessary results which contain the same keywords but related to other curricular types. So the system made by me serves as an advanced searching tool that allows the user to get specific curricular/curricula required by searching for occurrences of the searched keyword in PDF files. The magic is made to happen with an information retrieval process called Okapi BM25 method. The system provides a simple yet powerful interface for the user to get what they want with less effort.

This system has to interact with two kinds of people. They are,

01. the system admin who is in charge of the file uploads and system maintenance
&
02. the users that search for curricula.

The admin of the system needs a reliable way to add PDF files to the system while users need a reliable way to get what they are looking for. So the testing should be done by keeping in mind the point of view of these people.

There is another helpful feature in this system which suggests to the user about curricula related to what the user checked in the past.

4.2. Features to be Tested

4.2.1. Admin Side Tests

1. Capability to admin to add the documents to the system with tags (**Test ID: 1AT1**)

4.2.2. User Side Tests

1. Normal search

Table 4.1 - Expected Outcomes of Normal Search at Every Possibilities

<i>Test ID</i>	<i>Search Keyword</i>	<i>Year</i>	<i>Tags</i>	<i>Expected outcome</i>
2NS01	0	0	0	No keywords, no search
2NS02	0	0	1	
2NS03	0	1	0	
2NS04	0	1	1	
2NS05	1	0	0	All related documents
2NS06	1	0	1	Results with documents matching the keyword AND tags entered
2NS07	1	1	0	Results with documents matching the keyword AND the year entered
2NS08	1	1	1	Results with documents matching the keyword AND tags entered AND the year entered

Sates,

- Search keyword (1) – search keyword entered
- Search keyword (0) – search keyword not entered
- Year (1) – year entered
- Year (0) – year not entered
- Tags (1) – tags entered
- Tags (0) – tags not entered

2. Advance search

Table 4.2 - Expected Outcomes of Advance Search at Every Possibilities

<i>Test ID</i>	<i>Search keyword</i>	<i>Words to include</i>	<i>Words to exclude</i>	<i>Year</i>	<i>Tag</i>	<i>Expected outcome</i>
2AS01	0	0	0	0	0	No keywords, no search
2AS02	0	0	0	0	1	
2AS03	0	0	0	1	0	
2AS04	0	0	0	1	1	
2AS05	0	0	1	0	0	
2AS06	0	0	1	0	1	
2AS07	0	0	1	1	0	
2AS08	0	0	1	1	1	
2AS09	0	1	0	0	0	
2AS10	0	1	0	0	1	
2AS11	0	1	0	1	0	
2AS12	0	1	0	1	1	
2AS13	0	1	1	0	0	
2AS14	0	1	1	0	1	
2AS15	0	1	1	1	0	
2AS16	0	1	1	1	1	
2AS17	1	0	0	0	0	All related documents
2AS18	1	0	0	0	1	Documents matching the keyword AND the tags entered
2AS19	1	0	0	1	0	Results with documents matching the keyword AND the year entered
2AS20	1	0	0	1	1	Results with documents matching the keyword AND year AND the tags entered
<i>“Table 4.2. continued”</i>						

<i>"Table 4.2. concluded"</i>						
2AS21	1	0	1	0	0	Documents matching the keyword but which doesn't include the excluded words
2AS22	1	0	1	0	1	Documents matching the keyword AND the tags entered but which doesn't include the excluded words
2AS23	1	0	1	1	0	Documents matching the keyword AND the year entered but which doesn't include the excluded words
2AS24	1	0	1	1	1	Results with documents matching the keyword AND year entered AND the tags entered but which doesn't include the excluded words
2AS25	1	1	0	0	0	Results with documents matching the keyword AND matching the exact words in the words to include
2AS26	1	1	0	0	1	Results with documents matching the keyword AND the tags entered AND matching the exact words in the words to include
2AS27	1	1	0	1	0	Results with documents matching the keyword AND the year entered AND matching the exact words in the words to include
<i>"Table 4.2. continued"</i>						

<i>"Table 4.2. concluded"</i>						
2AS28	1	1	0	1	1	Results with documents matching the keyword AND year entered AND the tags entered AND matching the exact words in the words to include
2AS29	1	1	1	0	0	Results with documents matching the keyword AND matching the exact words in the words to include but which doesn't include the excluded words
2AS30	1	1	1	0	1	Results with documents matching the keyword AND the tags entered AND matching the exact words in the words to include but which doesn't include the excluded words
2AS31	1	1	1	1	0	Documents matching the keyword AND the year entered AND matching the exact words in the words to include but which doesn't include the excluded words
2AS32	1	1	1	1	1	Results with documents matching the keyword AND tags entered AND the year entered AND matching the exact words in the words to include but which doesn't include the excluded words

States,

- Search keyword (1) – search keyword entered
- Search keyword (0) – search keyword not entered
- Year (1) – year entered
- Year (0) – year not entered
- Tags (1) – tags entered
- Tags (0) – tags not entered
- Words to include (1) – words to include entered
- Words to include (0) – words to include not entered
- Words to exclude (1) – words to exclude entered
- Words to exclude (0) – words to exclude not entered

3. Recommendations (Test ID: 2REC01)

4. Share a document via social media, instant messenger, email (Test ID: 2SHR01)

4.2.3. System Side Tests

1. Okapi BM25 document data retrieval accuracy

a. Year (**Test ID: 3SS01**)

b. Words occurrences (**Test ID: 3SS02**)

2. Recommendations accuracy (Test ID: 3SSREC01)

4.3. Test Scenarios

Let's check a simple scenario. Say someone is looking for the curriculum of "Efficiency Bar Examination" of officers in the current system. They receive some results containing the required curricular plus deferments of salary increments of those officers who completed the examination and deferments of salary increments of those officers who did not complete the examination. The system made by me has given the facility to narrow the results by removing the results matching keywords given by the user as not required.

There are two ways to search using this system. The user can eliminate undesirable results from the result list using these two approaches, making it easier to use and requiring less effort.

- I. A normal search where user can search by,
 - keyword,
 - year and
 - related tags.

- II. An advance search where a combination of normal search methods plus a way to further narrow down the results by letting the user to enter,
 - words required in the results and
 - words which are not required to be in results.

Since the document's tags play a crucial role in search methods, only the administrator has the ability to add tags, giving this system some real-world guidance for preparing search results. Additionally, it assists in limiting the results.

4.4. Testing Approach

4.4.1. Upload Test

1. Uploading the sample “Efficiency Bar Examination” related PDFs with relevant tags

Table 4.3 - PDFs to Upload Under “Efficiency Bar Examination”

<i>PDF ID</i>	<i>PDF name</i>	<i>Tags</i>	<i>Year in document</i>
EBE01	Computer Test Relevant to the Efficiency bar Examination for Officers.pdf	computer test, efficiency bar examination, officers	2017
EBE02	Computer Test Relevant to the Efficiency bar Examination for Officers – 2017.pdf	computer test, efficiency bar examination, officers	2017
EBE03	Deferment of Salary Increments on non-completion of Efficiency Bar Examination.pdf	deferment, salary increments, non-completion, efficiency bar examination, officers	2018
EBE04	Deferment of salary increments on not completing the efficiency bar.pdf	deferment, salary increments, non-completion, efficiency bar examination, officers	2019
<i>“Table 4.3. continued”</i>			

<i>“Table 4.3. concluded”</i>			
EBE05	EBE of class III officers.pdf	efficiency bar examination, officers, class III, public management, assistants service	2014
EBE06	Reference the course related to the exemption of the officers from the Efficiency Bar Examination.pdf	exemption of officers, efficiency bar examination, course related	2022

2. Uploading the sample “Agrahara Insurance Scheme” related PDFs with relevant tags

Table 4.4 - PDFs to Upload Under “Agrahara Insurance Scheme”

<i>PDF ID</i>	<i>PDF name</i>	<i>Tags</i>	<i>Year in document</i>
AIS01	Agrahara Insurance Scheme for Public Officers.pdf	agrahara insurance, insurance schemes, public officers	2003
AIS02	Extending the benefits under Agrahara Insurance Scheme.pdf	benefits, extension of benefits, agrahara insurance, insurance schemes	2005
AIS03	Claims under AIS for COVID 19 positive officers.pdf	payment of claims, agrahara insurance, insurance scheme, public officers, covid-19, pandemic, covid-19 positive officers	2021

3. Normal search test cases

Table 4.5 - Normal Search Test Cases

<i>Test ID</i>	<i>Search keyword</i>	<i>Year</i>	<i>Tags</i>	<i>Expected result</i>
2NS01	-	-	-	Cannot proceed without a search keyword
2NS02	-	-	agrahara insurance	
<i>“Table 4.5. continued”</i>				

“Table 4.5. concluded”

2NS03	-	2003	-	Cannot proceed without a search keyword
2NS04	-	2003	agrahara insurance	
2NS05	agrahara	-	-	Results with PDF IDs - AIS01, AIS02, AIS03
2NS06	agrahara	-	benefits	Results with PDF ID - AIS02
2NS07	agrahara	2003	-	Results with PDF ID - AIS01
2NS08	agrahara	2021	covid-19	Results with PDF ID - AIS03

4. Advance search test cases

Table 4.6 - Advance Search test Cases

<i>Test ID</i>	<i>Search keyword</i>	<i>Words to include</i>	<i>Words to exclude</i>	<i>Year</i>	<i>Tag</i>	<i>Expected result</i>
2AS01	-	-	-	-	-	Cannot proceed without a search keyword
2AS02	-	-	-	-	officers	
2AS03	-	-	-	2019	-	
2AS04	-	-	-	2019	officers	
2AS05	-	-	agrahara	-	-	
2AS06	-	-	agrahara	-	officers	
2AS07	-	-	agrahara	2019	-	
2AS08	-	-	agrahara	2019	officers	
2AS09	-	bar examination	-	-	-	
2AS10	-	bar examination	-	-	officers	

“Table 4.6. continued”

"Table 4.6. concluded"						
2AS11	-	bar examination	-	2019	-	Cannot proceed without a search keyword
2AS12	-	bar examination	-	2019	officers	
2AS13	-	bar examination	agrahara	-	-	
2AS14	-	bar examination	agrahara	-	officers	
2AS15	-	bar examination	agrahara	2019	-	
2AS16	-	bar examination	agrahara	2019	officers	
2AS17	bar examination	-	-	-	-	Results with PDF IDs - EBE01, EBE02, EBE03, EBE04, EBE05, EBE06
2AS18	bar examination	-	-	-	computer test	Results with PDF IDs - EBE01, EBE02
2AS19	bar examination	-	-	2017	-	Results with PDF IDs - EBE01, EBE02
"Table 4.6. continued"						

"Table 4.6. concluded"						
2AS20	bar examination	-	-	2014	class III	EBE05
2AS21	bar examination	-	computer test	-	-	Results with PDF IDs - EBE03, EBE04, EBE05, EBE06
2AS22	bar examination	-	computer test	-	salary increments	Results with PDF IDs - EBE03, EBE04
2AS23	bar examination	-	computer test	2014	-	EBE05
2AS24	bar examination	-	computer test	2019	salary increments	EBE04
2AS25	bar examination	computer test	-	-	-	Results with PDF IDs - EBE01, EBE02
2AS26	bar examination	class III	-	-	deferment	EBE05
2AS27	bar examination	non - completion	-	2018	-	EBE03
2AS28	bar examination	exemption of officers	-	2022	exemption of officers	EBE06
"Table 4.6. continued"						

"Table 4.6. concluded"						
2AS29	bar examination	officers	computer test	-	-	Results with PDF IDs – EBE03, EBE04, EBE05, EBE06
2AS30	bar examination	officers	computer test	-	efficiency bar examination	Results with PDF IDs – EBE03, EBE04, EBE05, EBE06
2AS31	bar examination	officers	computer test	2018	-	EBE03
2AS32	bar examination	officers	computer test	2014	efficiency bar examination	EBE05

4.4.2. Recommendations Test

1. Open a PDF related to "Efficiency Bar Examination" from search results and check the recommendations when opened document history shows only that file.

Procedure,

- Delete opened document history from "Recently opened documents".
- Open PDF ID – EBE01 and close it
- Search for any document
- Check the recommendations shown
- Expected recommendations,
 - Results with PDF IDs - EBE02, EBE03, EBE04, EBE05, EBE06 (Not showing EBE01 since it is already shown.

Table 4.7 - Scores that were calculated in the recommendation system when the PDF "EBE01" was first opened

<i>PDF</i>	<i>Score</i>
Computer Test Relevant to the Efficiency bar Examination for Officers - 2017-1668945981.4800775.pdf	0.3434428381623789
Deferment of Salary Increments on non-completion of Efficiency Bar Examination-1668946003.0828907.pdf	0.06708065499397613
Deferment of salary increments on not completing the efficiency bar-1668946042.8416.pdf	0.06647273692493798
EBE of class III officers-1668946059.80287.pdf	0.059335227554227016
Reference the course related to the exemption of the officers from the Efficiency Bar Examination-1668946095.5181093.pdf	0.04305171344744481

- Open a PDF related to "Agrahara Insurance Scheme" from search results and checking the recommendations when opened document history shows only that file.

Procedure,

- Delete opened document history from "Recently opened documents".
- Open PDF ID – AIS01 and close it
- Search for any document
- Check the recommendations shown
- Expected recommendations,
 - Results with any PDF IDs - AIS02, AIS03 (Not showing AIS01 since it is already shown.)
 - Results with any PDF IDs - EBE01, EBE02, EBE03, EBE04

Table 4.8 - Scores that were calculated in the recommendation system when the PDF "AIS01" was first opened

<i>PDF</i>	<i>Score</i>
Claims under AIS for COVID 19 positive officers-1666172992.1150625.pdf	0.037536188926928296
Extending the benefits under Agrahara Insurance Scheme-1666172969.822767.pdf	0.01931392960783257
Computer Test Relevant to the Efficiency bar Examination for Officers-1666172634.5892184.pdf	0.006783031266493108
Computer Test Relevant to the Efficiency bar Examination for Officers - 2017-1668945981.4800775.pdf	0.006683940215877608
Deferment of Salary Increments on non-completion of Efficiency Bar Examination-1668946003.0828907.pdf	0.005714313002450784

- Open a PDF related to "Agrahara Insurance Scheme" and a PDF related to "Efficiency Bar Examination" from search results and checking the recommendations when opened document history shows only those two files.

Procedure,

- Delete opened document history from "Recently opened documents".
- Open PDF ID – EBE01 and close it
- Open PDF ID – AIS01 and close it
- Search for any document
- Check the recommendations shown
- Expected recommendations,
 - Results with any PDF IDs - AIS02, AIS03 (Not showing AIS01 since it is already shown.)
 - Results with any PDF IDs - EBE02, EBE03, EBE04, EBE05, EBE06 (Not showing EBE01 since it is already shown.)
 - Results can be of any 5 PDFs from above results

Table 4.9 - Scores Calculated in Recommendation System when PDF “EBE01” and PDF “AIS01” was Opened Before

<i>PDF</i>	<i>Score</i>
Computer Test Relevant to the Efficiency bar Examination for Officers - 2017-1668945981.4800775.pdf	0.3434428381623789
Deferment of Salary Increments on non-completion of Efficiency Bar Examination-1668946003.0828907.pdf	0.06708065499397613
Deferment of salary increments on not completing the efficiency bar-1668946042.8416.pdf	0.06647273692493798
EBE of class III officers-1668946059.80287.pdf	0.059335227554227016
Reference the course related to the exemption of the officers from the Efficiency Bar Examination-1668946095.5181093.pdf	0.04305171344744481

4.4.3. PDFs That We Have Used To Test

1. Efficiency Bar Examination related
 - a. Computer Test Relevant to the Efficiency bar Examination for Officers.pdf
 - b. Computer Test Relevant to the Efficiency bar Examination for Officers – 2017.pdf
 - c. Deferment of Salary Increments on non-completion of Efficiency Bar Examination.pdf
 - d. Deferment of salary increments on not completing the efficiency bar.pdf
 - e. EBE of class III officers.pdf
 - f. Reference the course related to the exemption of the officers from the Efficiency Bar Examination.pdf

2. Agrahara Insurance scheme related
 - a. Agrahara Insurance Scheme for Public Officers.pdf
 - b. Extending the benefits under Agrahara Insurance Scheme.pdf
 - c. Claims under AIS for COVID 19 positive officers.pdf

4.5. The Resources Allocated to Test The App

- Local test – Local computer

4.6. Test Result

Table 4.10 - Results of the Test are shown below

<i>Test ID</i>	<i>Expected Result</i>	<i>Actual Result</i>
1AT1	Capability to admin to add the documents to the system with tags	The admin was able to upload documents with tags, so the result was as expected.
2NS01	No keywords, no search	No keywords, no search, so the result was as expected.
2NS02		
2NS03		
2NS04		
2NS05	Shows all related documents	Showed all related documents, so the results were as expected.
2NS06	Documents that match the keywords and tags entered in the search results	All documents matching the keywords and tags entered appeared in the search results, so the results were as expected.
2NS07	Documents that match the keywords and the year entered in the search results	All documents matching the keywords and the year entered appeared in the search results, so the results were as expected.
2NS08	Documents that match the keywords, tags and the year entered in the search results	All documents matching the keywords, tags and the year entered appeared in the search results, so the results were as expected.
<i>“Table 4.10 continued”</i>		

<i>"Table 4.10 concluded"</i>		
2AS01	No keywords, no search	No keywords, no search, so the result was as expected.
2AS02		
2AS03		
2AS04		
2AS05		
2AS06		
2AS07		
2AS08		
2AS09		
2AS10		
2AS11		
2AS12		
2AS13		
2AS14		
2AS15		
2AS16		
2AS17	Shows all related documents	Showed all related documents
2AS18	Documents that match the keywords and tags entered in the search results	All documents matching the keywords and tags entered appeared in the search results, so the results were as expected.
2AS19	Documents that match the keywords and the year entered in the search results	All documents matching the keywords and the year entered appeared in the search results, so the results were as expected.
2AS20	Documents that match the keywords, year and the tags entered in the search results	All documents matching the keywords, year and the tags entered appeared in the search results, so the results were as expected.
<i>"Table 4.10 continued"</i>		

<i>"Table 4.10. concluded"</i>		
2AS21	Results with documents matching the keyword but which doesn't include the excluded words	Documents that matched the keywords but did not contain the excluded words in the result, so the results were as expected.
2AS22	Results with documents matching the keyword AND the tags entered but which doesn't include the excluded words	Documents that matched the keyword and the tags but did not contain the words that were excluded appeared in the result, so the results were as expected.
2AS23	Results with documents matching the keyword AND the year entered but which doesn't include the excluded words	Documents that matched the keyword and the year but did not contain the words that were excluded appeared in the result, so the results were as expected.
2AS24	Results with documents matching the keyword AND year entered AND the tags entered but which doesn't include the excluded words	Documents that matched the keyword, year and tags but did not contain the words that were excluded appeared in the result, so the results were as expected.
2AS25	Results with documents matching the keyword AND matching the exact words in the words to include	Documents matching the keyword and having words in exact words were appearing in the result, so the results were as expected.
2AS26	Results with documents matching the keyword AND the tags entered AND matching the exact words in the words to include	Documents matching the keyword, tags entered and having exact words were appearing in the result, so the results were as expected.
2AS27	Results with documents matching the keyword AND the year entered AND matching the exact words in the words to include	Documents matching the keyword, year and having words in exact words were appearing in the result, so the results were as expected.
<i>"Table 4.10 continued"</i>		

<i>"Table 4.10 concluded"</i>		
2AS28	Results with documents matching the keyword AND year entered AND the tags entered AND matching the exact words in the words to include	Documents matching the keyword, year and tags entered and having words in exact words were appearing in the result, so the results were as expected.
2AS29	Results with documents matching the keyword AND matching the exact words in the words to include but which doesn't include the excluded words	Documents that matched the keyword and words in exact words, but did not contain the excluded words were appearing in the result, so the results were as expected.
2AS30	Results with documents matching the keyword AND the tags entered AND matching the exact words in the words to include but which doesn't include the excluded words	Documents matching the keyword, tags entered and having exact words, but did not contain the words that were excluded were appearing in the result, so the results were as expected.
2AS31	Results with documents matching the keyword AND the year entered AND matching the exact words in the words to include but which doesn't include the excluded words	Documents matching the keyword, tags entered and having exact words, but did not contain the words that were excluded were appearing in the result, so the results were as expected.
2AS32	Results with documents matching the keyword AND tags AND year AND matching the exact words in the words to include but which doesn't include the excluded words	Documents matching the keyword, tags, year entered and having exact words, but did not contain the words that were excluded were appearing in the result, so the results were as expected.
<i>"Table 4.10 continued"</i>		

<i>“Table 4.10 concluded”</i>		
2REC01	Assuming the client has opened any PDF previously, suggestion ought to show related in the proposal segment. A message stating that there are currently no recommendations should appear if nothing has been viewed.	When no PDFs were opened, no suggestions were displayed, while when PDFs were opened, suggestions were displayed, so the results were as expected.
2SHR01	The opened document should have a link that can be shared via email, instant messaging, and social media.	A link that can be shared via email, instant messaging, and social media is in the opened document, so the results were as expected.
3SS01	The year listed in each document should appear in the documentcategory.txt file, as shown in tables 1 and 2.	Tables 1 and 2 in the documentcategory.txt file correspond to the years listed, so the results were as expected.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Conclusions

Searching for a required circular over a circular repository becomes a very tedious and time consuming task. Therefore, the absence of a well-functioning circular storage and retrieval system has led the researcher to build such a system. Implementation of this system will help users to quickly and easily retrieve the circulars that they are looking for.

In this thesis, we discussed the literature of IR and Recommendation Systems and its working principles. We then explained in detail how to create an inverted index, apply the Okapi BM25 document ranking model and implement a user based content based recommendation model to the circular corpus after performing pre-processing steps on circular documents. In the evaluation section the researcher demonstrated that the final system worked effectively on circular documents. Therefore, as the final output of this research, a circular retrieval and recommendation system is implemented which allows users to search circulars easily and quickly.

5.2. Limitations

Though the researcher made considerable effort, the limitations that still exists in this thesis can be explained as below,

1. Only supports the English language.
2. Some old circulars are uploaded as images. Those are not supported for search.

5.3. Future Work

Therefore, future work can be done on,

1. Making this system to support multi-language.
2. To read the text from circulars which are uploaded as images and enable to search by circulars uploaded as images also.
3. Current recommendation system is built on the time the user spent on reading a specific circular. The researcher identified when working with a large corpus of circulars better accuracy can be obtained by using circular tags + the time the user

spent on reading circulars. Therefore, this will allow supervised machine learning to be used on this system.

REFERENCES

- Aggarwal, C.C., 2016. Model-based collaborative filtering. In *Recommender systems* (pp. 71-138). Springer, Cham.
- Alexandros Ntoulas, Junghoo Cho, and Christopher Olston. 2004. What's new on the web? the evolution of the web from a search engine perspective. In Proceedings of the 13th international conference on World Wide Web (WWW '04). Association for Computing Machinery, New York, NY, USA, 1–12. <https://doi.org/10.1145/988672.988674>
- Amatriain, X. and Basilico, J., 2015. Recommender systems in industry: A netflix case study. In *Recommender systems handbook* (pp. 385-419). Springer, Boston, MA.
- Anders ME, Evans DP. Comparison of PubMed and Google Scholar literature searches. *Respiratory care*. [Comparative Study].2010;55(5):578-83
- Ann Sebastian (2020). A Gentle Introduction To Calculating The TF-IDF Values. *Medium*. [online] 16 Jul. Available at: <https://towardsdatascience.com/a-gentle-introduction-to-calculating-the-tf-idf-values-9e391f8a13e5>.
- Antai, Roseline. (2011). The Use of Latent Semantic Indexing to Cluster Documents into their Subject Areas. 10.13140/2.1.2225.6000.
- Baeza-Yates, R. and Ribeiro-Neto, B., 1999. *Modern information retrieval* (Vol. 463). New York: ACM press.
- Baker, K., 2005. Singular value decomposition tutorial. *The Ohio State University*, 24.
- Buck, C. and Koehn, P., 2016, August. Quick and reliable document alignment via tf/idf-weighted cosine distance. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers* (pp. 672-678).
- Burke, R., 2000. Knowledge-based recommender systems. *Encyclopedia of library and information systems*, 69(Supplement 32), pp.175-186.
- Burke, R., 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4), pp.331-370.

Cafarella, M.J. and Etzioni, O., 2005, May. A search engine for natural language applications. In *Proceedings of the 14th international conference on World Wide Web* (pp. 442-452).

Castillo, C., 2005, June. Effective web crawling. In *Acm sigir forum* (Vol. 39, No. 1, pp. 55-56). New York, NY, USA: Acm.

Chaudhary, M. (2020). TF-IDF Vectorizer scikit-learn. [online] Medium. Available at: <https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a>.

Chimah, J. and Unagha, A. (2010). Information Retrieval in Libraries and Information Centres: Concepts, Challenges and Search Strategies. [online] 20 -Journal of Applied Information Science and Technology, p.4. Available at: https://www.jaistonline.org/ChimahUnaghaNwokocha_2k10.pdf [Accessed 30 Nov. 2020].

Claypool, M., Le, P., Wased, M. and Brown, D., 2001, January. Implicit interest indicators. In *Proceedings of the 6th international conference on Intelligent user interfaces* (pp. 33-40).

Cvitanic, T., Lee, B., Song, H.I., Fu, K. and Rosen, D., 2016, January. LDA v. LSA: A comparison of two computational text analysis tools for the functional categorization of patents. In *International Conference on Case-Based Reasoning*.

Dhruvil Karani (2018). Introduction to Word Embedding and Word2Vec. [online] Medium. Available at: <https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa>.

Diao, Y., Jamjoom, H. and Loewenstern, D., 2009, September. Rule-based problem classification in it service management. In *2009 IEEE International Conference on Cloud Computing* (pp. 221-228). IEEE.

Falagas ME, Pitsouni EI, Malietzis GA, Pappas G. Comparison of PubMed, Scopus, Web of Science, and Google Scholar: strengths and weaknesses. *FASEB J*. 2008 Feb;22(2):338-42. doi: 10.1096/fj.07-9492LSF. Epub 2007 Sep 20. PMID: 17884971.

Gudivada, V.N., Raghavan, V.V., Grosky, W.I. and Kasanagottu, R., 1997. Information retrieval on the world wide web. *IEEE Internet Computing*, 1(5), pp.58-68.

- Gupta, D., Bhatia, K. and Sharma, A. (01 2009) "A Novel Indexing Technique for Web Documents using Hierarchical Clustering," *International Journal of Computer Science and Network Security*, 9 No 9, p. 16
- Huang, A., 2008, April. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, Christchurch, New Zealand (Vol. 4, pp. 9-56).
- Jia, Z., Yang, Y., Gao, W. and Chen, X., 2015, February. User-based collaborative filtering for tourist attraction recommendations. In *2015 IEEE international conference on computational intelligence & communication technology* (pp. 22-25). IEEE.
- Kapronczay, M. (2021). The beginners guide to language models. [online] Medium. Available at: <https://towardsdatascience.com/the-beginners-guide-to-language-models-aa47165b57f9>.
- Kaur, H. and Gupta, V., 2016, August. Indexing process insight and evaluation. In *2016 International Conference on Inventive Computation Technologies (ICICT)* (Vol. 3, pp. 1-5). IEEE.
- Khatwani, S. and Chandak, M.B., 2016, September. Building Personalized and Non Personalized recommendation systems. In *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)* (pp. 623-628). IEEE.
- Kumar, M., Yadav, D.K., Singh, A. and Gupta, V.K., 2015. A movie recommender system: Movrec. *International Journal of Computer Applications*, 124(3).
- Lops, P., Gemmis, M.D. and Semeraro, G., 2011. Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pp.73-105.
- Lydia, L. et al. (04 2018) "Document Clustering Based On Text Mining K-Means Algorithm Using Euclidean Distance Similarity," *Journal of Advanced Research in Dynamical and Control Systems*.
- Manning, C.D., 2008. *Introduction to information retrieval*. Syngress Publishing.
- Melucci, M. (2009). Vector-Space Model. *Encyclopedia of Database Systems*, [online] pp.3259–3263. Available at: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_918.

Menéndez, M.L., Pardo, J.A., Pardo, L. and Pardo, M.C., 1997. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334(2), pp.307-318.

Michael Gordon, Praveen Pathak, Finding information on the World Wide Web: the retrieval effectiveness of search engines, *Information Processing & Management*, Volume 35, Issue 2, 1999, Pages 141-180, ISSN 0306-4573, [https://doi.org/10.1016/S0306-4573\(98\)00041-7](https://doi.org/10.1016/S0306-4573(98)00041-7)

Murtagh, F. and Contreras, P., 2012. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1), pp.86-97.

Nigam, V., 2022. Understanding Recommender Systems: Introduction. [online] Medium. Available at: <https://medium.com/analytics-vidhya/understanding-recommender-systems-introduction-3be54e937625> [Accessed 26 April 2022].

Salton, G., Fox, E.A. and Wu, H., 1983. Extended boolean information retrieval. *Communications of the ACM*, 26(11), pp.1022-1036.

Saritha, R.C. and Kulkarni, A., 2013. Text Clustering Using Mixture Models. *Publications of Problems & Application in Engineering Research*.

Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., 2001, April. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285-295).

Seitz, R., 2020. Understanding TF-IDF and BM-25. [online] KMW Technology. Available at: <https://kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm-25/> [Accessed 22 April 2022].

Smith, B. and Linden, G., 2017. Two decades of recommender systems at Amazon. com. *Ieee internet computing*, 21(3), pp.12-18.

Strzalkowski, T. ed., 1999. *Natural language information retrieval* (Vol. 7). Springer Science & Business Media.

Tober, Markus. (2011). PubMed, ScienceDirect, Scopus or Google Scholar – Which is the best search engine for effective literature research in laser medicine?. *Medical Laser Application*. 26. 139-144. 10.1016/j.mla.2011.05.006.

Tong, Zhou & Zhang, Haiyi. (2016). A Document Exploring System on LDA Topic Model for Wikipedia Articles. *The International Journal of Multimedia & Its Applications*. 8. 01-13. 10.5121/ijma.2016.8401

Toppr-guides. (2018). Circulars: Definition, Advantages, Samples and Solved Question. [online] Available at:

<https://www.toppr.com/guides/business-correspondence-and-reporting/official-communication/circulars/> [Accessed 16 April 2022]

Vechtomova, O., 2009. Book Review: Introduction to Information Retrieval by Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Computational Linguistics*, 35(2).

W. Ahmad and R. Ali, "Textual content based information retrieval from Twitter," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, 2016, pp. 2668-2672, doi: 10.1109/ICACCI.2016.7732462.

Xia, P., Zhang, L. and Li, F., 2015. Learning similarity with cosine similarity ensemble. *Information Sciences*, 307, pp.39-52.

Yu, K., Schwaighofer, A., Tresp, V., Xu, X. and Kriegel, H.P., 2004. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, 16(1), pp.56-69.

Zhang, Q., Lu, J. and Jin, Y., 2021. Artificial intelligence in recommender systems. *Complex & Intelligent Systems*, 7(1), pp.439-457.

Zhao, R. and Mao, K., 2017. Fuzzy bag-of-words model for document representation. *IEEE transactions on fuzzy systems*, 26(2), pp.794-804.

