Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7546513

IN-DEPTH ANALYSIS AND COMPARATIVE STUDY OF VARIOUS INFORMATION RETRIEVAL SYSTEMS: AN INSTITUTIONAL PERSPECTIVE

Dr. M.Vikram[1], Asadi Muni Hemanth[2] , Dr. N. Sudhakar Reddy[3], Sampathirao Suneetha[4], Rajesh Chandra Chokkara[5] , Dr. Phani Kumar Solleti [6]

[1] Associate Professor, Department of Computer and Engineering, Sri Venkateswara College of Engineering, Tirupati, AP, India.

[2]Assistant Professor, Department of Computer and Engineering, Sri Venkateswara College of Engineering, Tirupati, AP, India.

[3] Professor, Department of Computer and Engineering, Sri Venkateswara College of Engineering, Tirupati, AP, India.

[4] Research Scholar, Department of CS&SE, AUCE, Andhra University, Visakhapatnam, AP, India (corresponding author),

[5] Lecturer, Department of Mathematics, Ambitus World School, Vijayawada, AP, India.

[6] Assistant Professor, Department of CSE, K. L. Deemed to be University, Vijayawada, AP, India.

Abstract : Information retrieval (IR) is the key technology for finding relevant data from large collections. The main challenge of IR is to collect and manage all the information in the collection. People need to access the information that suits their needs at the right time. However, the excessive availability of information causes information overload and makes it difficult to find the relevant information. To overcome these difficulties, several automated tools are used to search for information that matches the user's needs. The role of IR is to collect and represent the information and enable the retrieval of the relevant information for specific problems in real time. This paper focuses on the different IR models that identify the user's query and retrieve the information from the collection of documents in a specific application domain. If the user's query matches the search engine, it retrieves the relevant information from the collection. This paper presents different IR models with solved examples and analyzes the IR metrics for ranked and unranked models. It also compares the classical and probabilistic models using different parameters.

Keywords: Information retrieval, query, IR Models

1. INTRODUCTION

Information Retrieval (IR) operates as a fully automated process, responding to user queries by analyzing a set of documents and presenting a sorted list of documents deemed relevant to the user's expressed requirements. IR can be described as a software application that handles the organization, storage, retrieval, and assessment of information from document repositories, particularly textual data. While the system aids users in locating the information they seek, it does not explicitly provide answers to questions. Instead, it identifies the existence and whereabouts of documents that may contain the desired information. Documents that meet the user's needs are termed relevant documents. An optimal IR system

retrieves only relevant documents. Figure 1 illustrates the process of Information Retrieval (IR), providing insights into its workings.
The diagram clearly depicts that users begin by formulating a request in the form of a query. Subsequently, the Information Retrieval (IR) system responds by retrieving output in the form of documents that are relevant to the user's query.

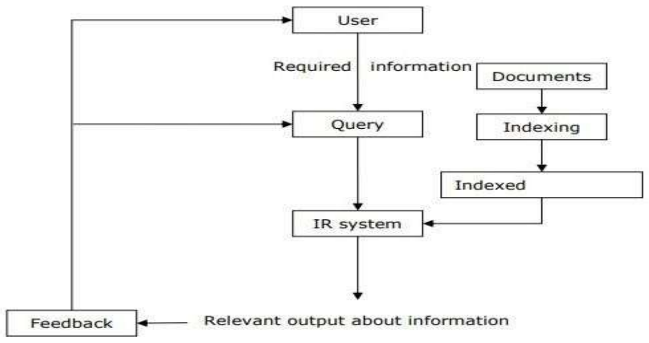


Figure 1: Process of Information retrieval system

1.1 Classical Problem in Information Retrieval (IR) System:

The primary objective of an IR system is to establish a model for efficiently retrieving information from a collection of documents. One classical challenge within this domain is known as the ad-hoc retrieval problem.
In ad-hoc retrieval, users input a query describing the information they seek. The IR system then returns relevant documents related to the specified information. For instance, when searching the Internet, the system may provide pages closely aligned with the user's search, but there might also be non-relevant pages. This phenomenon is attributed to the ad-hoc retrieval problem.
From a mathematical perspective, a retrieval model is composed of: D: Represents documents, R: Represents queries, F: The modeling framework for D and Q, along with the relationship between them.
$R(q, d_i)$: A similarity function that orders the documents concerning the query, also known as ranking.

INFORMATION RETRIEVAL (IR) MODELS

An IR model can be categorized into three models, as depicted in Figure 2.

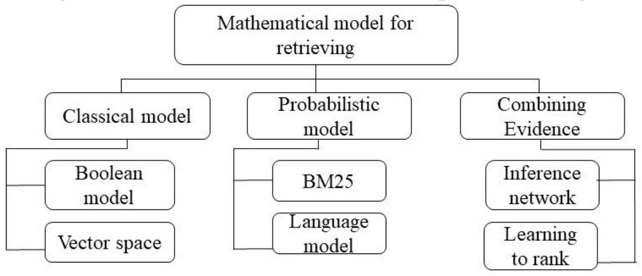


Figure 2: classification of mathematical model for retrieving.[1]

1.2 Classical Models:

2.1.1. Standard Boolean Model: The Standard Boolean model stands as one of the simplest retrieval models, employing exact matching to align documents with a user's request or information query. This model determines document relevance based on matching words in the query. The term "Boolean" indicates the logical relationships among search terms, utilizing operators like AND, OR, and NOT. The Boolean Model is rooted in Boolean algebra and set theory. Each document is represented by the index terms assigned to it, with no indication of term importance (weights are binary, either 0 or 1) [2].

Formulation:

F: Boolean logic over sets of terms and sets of documents

D: set of words (indexing terms) in document that every term is either present (1) or absent (0). Q: A Boolean expression that contain terms that are indexed and operators are AND, OR, NOT

R (q, d i): Predicts relevant documents to a query expression if it satisfies the query expression, that every query terms specifies a group of documents containing the term: And (∧): The intersection of two sets, OR (∨): The union of two sets, Not (¬): Set inverse or really set difference

Example:

Doc1: "Artificial Intelligence is revolutionizing industries and technology."

Doc2: "Machine learning is a subset of Artificial Intelligence."

Doc3: "Robotics involves the design and creation of intelligent machines."

Doc4: "Natural Language Processing enables computers to understand and respond to human language."

Table 1: Bag of Words of Query Terms:

Term	**Doc1**	**Doc2**	**Doc3**	**Doc4**
Artificial	1	1	0	0
Intelligence	1	1	1	0
Machine learning	0	1	0	0
Robotics	0	0	1	0
Natural Language	0	0	0	1
Processing	0	0	0	1

New Query Q: (Artificial ∧ Intelligence) ∨ (Machine learning ∧ Robotics)

Updated Results: (Artificial ∧ Intelligence): Doc1, Doc2

(Machine learning ∧ Robotics): No common documents

Final Result: (Artificial ∧ Intelligence) ∨ (Machine learning ∧ Robotics): Doc1, Doc2

2.1.2 Vector Space Model: In information retrieval, the VSM uses vectors of weights to represent both documents and queries. Each weight corresponds to an index term in a document or a query. The weights are calculated from the frequency of the index terms in the document, the query, or the whole collection. When a query is made, the documents are sorted by the cosine similarity between their vectors and the query vector [3]. Let us explore this concept with an example: Consider three documents and a query:

Document 1 (D1): "technology innovation"
Document 2 (D2): "artificial intelligence development"
Document 3 (D3): "machine learning applications"
Query: "innovation in machine learning"

Document Vectors Representation:

The first step is to break down each document into individual words. Then, some words and symbols that are not relevant, such as stop words, punctuations, and special characters like #@$, are removed. After that, each document is converted into a vector of words. The following example shows how the document vectors look like.

D1: (technology, innovation)
D2: (artificial, intelligence, development)
D3: (machine, learning, applications)
Query: (innovation, machine, learning)

The next step is to convert the vectors of words we created into numbers. This is called a term document matrix.

Term Document Matrix: A term document matrix is a way of representing documents and words as numbers in a matrix. Each row of the matrix corresponds to a word, and each column corresponds to a document. The number in each cell indicates how many times the word appears in the document. If a word does not appear in a document, the number is zero. After making the term document matrix, we need to assign weights to each word in each document. The weights help us identify the words that are most relevant for each document [4]. One method to calculate the weights is called term frequency - inverse document frequency (tf-idf) using the formula in equation (1). This method gives higher weights to the words that are frequent in a document, but rare in other documents. It gives lower weights to the words that are common in many documents. This way, we can capture the uniqueness of each document based on its words.

$$\textbf{Tf-idf} = \textbf{tf} \times \textbf{idf} \qquad \textbf{(1)}$$

where, tf (Term Frequency) describes how often a word appears in a document idf (Inverse Document Frequency) and idf = log(N/df), where df is the number of documents that have the word.

Tf-idf is a way of measuring how important a word is in a document, compared to other documents. It gives a higher score to words that are frequent in a document, but rare in other documents. It gives a lower score to words that are common in many documents.

Table 2: TF scores of terms:

	technology	innovation	artificial	intelligence	development	machine	learning	applications

D1	1	1	0	0	0	0	0	0
D2	0	0	1	1	1	0	0	0
D3	0	1	0	0	0	1	1	1

Using the formula TF-IDF = TF * IDF, where TF is the term frequency, and IDF is the inverse document frequency. Calculate IDF values for each term:
technology: log2(3/1) = 1.584, innovation: log2(3/2) = 0.584, artificial: log2(3/1) = 1.584, intelligence: log2(3/1) = 1.584, development: log2(3/1) = 1.584, machine: log2(3/2) = 0.584, learning: log2(3/2) = 0.584, applications: log2(3/1) = 1.584

Table 3: TF-IDF scores of terms

	technology	innovation	artificial	intelligence	development	machine	learning	applications
D1	1.584	0.584	0	0	0	0	0	0
D2	0	0	1.584	1.584	1.584	0	0	0
D3	0	0.584	0	0	0	0.584	0.584	1.584

Table 4: TF-IDF Vector for the Query:

	technology	innovation	artificial	intelligence	development	machine	learning	applications
Query	0	0.584	0	0	0	0.584	0.584	0

Similarity Measures: cosine similarity:

Cosine similarity is a way of measuring how similar two vectors are by looking at the angle between them. A vector is a list of numbers that represents some data, such as ratings, features, or words. The angle between two vectors shows how much they point in the same direction. The smaller the angle, the more similar the vectors are. The larger the angle, the more different the vectors are [4].

To calculate the cosine similarity, we use the formula in equation (2):

$$\mathbf{sim(x, y)} = \frac{\mathbf{x \cdot y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \qquad \mathbf{(2)}$$

where x and y are the two vectors, x·y is the dot product of the vectors, and ǁxǁ and ǁyǁ are the lengths of the vectors. The dot product is the sum of the products of the corresponding elements of the vectors. The length of a vector is the square root of the sum of the squares of its elements.

The cosine similarity ranges from -1 to 1. A cosine similarity of 1 means that the vectors are identical, pointing in the same direction. A cosine similarity of 0 means that the vectors are perpendicular, pointing in opposite directions. A cosine similarity of -1 means that the vectors are opposite, pointing in opposite directions. The closer the cosine similarity is to 1, the more similar the vectors are. The closer the cosine similarity is to -1, the more different the vectors are.

Cosine Similarity Calculation: Calculate the cosine similarity between each document vector and the query vector.

CosSim(D1, Query) = 0.586, CosSim(D2, Query) = 0.586, CosSim(D3, Query) = 0.292

The final order of documents presented as results to the query based on cosine similarity values would be: D1, D2, D3.

2.2 Probabilistic models:

2.2.1 The BM25 Algorithm: BM25 is a way of measuring how relevant a document is to a query, based on the terms they share. It is derived from a mathematical model of how likely a document is to be relevant, created by Stephen E. Robertson, Karen Spärck Jones, and others in the 1970s and 1980s. The name BM25 stands for best matching, and it was first used in a system called Okapi, hence the name Okapi BM255. BM25 assigns a score to each document based on how many times the query terms appear in it, and how important those terms are in the whole collection of documents. It does not care about the order or the distance of the terms in the document, only their frequency [6]. The formula for BM25 is in equation (3):

$$\mathbf{score}(D,Q)=\sum_{i=1}^{n}\mathrm{IDF}(q_i)\cdot\frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\mathrm{avgdl}}\right)} \qquad (3)$$

Where D is a document, Q is a query, n is the number of terms in the query, qi is the i-th term in the query, IDF(qi) is the inverse document frequency of the term, which measures how rare or common it is in the collection, f(qi,D) is the term frequency of the term in the document, which measures how often it appears in the document, k1 and b are parameters that control the influence of term frequency and document length on the score, |D| is the length of the document, measured by the number of words and avgdl is the average document length in the collection.

The formula sums up the contribution of each query term to the document score, weighted by their inverse document frequency and adjusted by their term frequency and document length.

Example:

D1: amazing smart phone, good camera, good battery life and have good specification. D2: Good picture quality and camera handling is good.

D3: Good sound quality, good battery life, good for gaming and good performance. Query: (good, and, amazing).

Total tokens: 32 and Document length (dl): D1- 12, D2- 8, D3- 12; **qi** is the ith query term.

From above example our query is good, and, amazing where q0 is good, q1 is ‘and’ and q2 is amazing.

IDF(gi) stands for inverse document frequency of the i-th term in a query. It is a way of calculating how rare or common a term is in a set of documents. The formula for IDF(gi) is in equation (4):

$$\mathbf{IDF}(g_i) = \ln\left(1 + \frac{\mathbf{docCount} - f(q_i) + 0.5}{f(q_i) + 0.5}\right) \qquad \mathbf{(4)}$$

where docCount is the number of documents in the set, and f(qi) is the number of documents that contain the i-th term. For example, if the term "good" appears in all 3 documents in the set, then f(qi) is 3, and IDF(gi) is: IDF("good") = ln(1+((3-3+0.5))/(3+0.5)) =0.1335, IDF("and") = ln(1+((3-3+0.5))/(3+0.5)) = 0.1335
IDF("amazing") = ln(1+((3-1+0.5))/(1+0.5)) = 0.4259
The BM25 formula has several parts that affect how a document is scored for a query. These parts are:

- |D| / avgdl: This is the ratio of the document length to the average document length in the collection. It reflects how long or short a document is compared to others. A longer document will have a larger ratio, which will lower its score. A shorter document will have a smaller ratio, which will increase its score. This is because a longer document is more likely to have terms that are not relevant to the query, while a shorter document is more focused on the query terms.
- b: This is a parameter that controls how much the document length ratio influences the score. It ranges from 0 to 1. A larger b means that the document length ratio has more effect on the score. A smaller b means that the document length ratio has less effect on the score. If b is 0, then the document length ratio has no effect on the score. The default value of b is 0.75.
- f(qi,D): This is the term frequency of the i-th query term in the document. It counts how many times the query term appears in the document. A higher term frequency means that the query term is more important in the document, and it will increase the score. A lower term frequency means that the query term is less important in the document, and it will decrease the score.
- k1: This is a parameter that controls how much the term frequency influences the score. It is usually a positive number. A larger k1 means that the term frequency has more effect on the score. A smaller k1 means that the term frequency has less effect on the score. The default value of k1 is 1.2.

2.2.2 Language Model: A language model is a way of estimating how likely a document is to produce a query, based on the words it contains. We want to calculate P(Q|Md), which is the probability of the query given the document's language model [7].

To do this, we first need to estimate the probability of each word in the query under the document's word distribution. A simple way to do this is to use the maximum likelihood estimate, which is in equation (5):

$$\hat{p}_{ml}(t \mid Md) = \frac{tf(t,d)}{dld} \qquad \mathbf{(5)}$$

where tf(t,d) is how many times the word t appears in the document d, and dld is the total number of words in the document d. We assume that the words in the query are independent of each other, so we can multiply their probabilities to get the ranking score for the document as in equation (6):

$$\prod_{t\in Q}\hat{p}_{ml}(t \mid Md) \qquad (6)$$

However, this method has some problems. One problem is that it gives zero probability to any document that does not have all the words in the query, which is too harsh. Another problem is that it does not account for how common or rare the words are in the whole collection of documents. For example, cft is how many times the word t appears in the whole collection, and cs is the total number of words in the collection. A word that is very rare in the collection should have a higher probability than a word that is very common, because it is more informative.

To solve these problems, we need to use a more robust estimate of the word probability, which is in equation (7):

$$\hat{p}_{avg}(t) = \frac{\sum_{d:t\in d}\hat{p}_{ml}(t|Md)}{dft} \qquad (7)$$

where dft is how many documents contain the word t. This estimate is based on the average probability of the word across all the documents that have it, which is more reliable than the probability from a single document. However, this estimate also has a problem. It does not distinguish between documents that have different frequencies of the word. For example, a document that has the word t ten times should have a higher probability than a document that has the word t only once.

To address this problem, we need to use a risk function that measures how risky it is to use the average probability for a given document. The risk function is in equation (8):

$$\hat{R}_{t,d} = \left(\frac{1}{1+ft}\right)^{tf(t,d)-ft}\cdot(1+ft) \qquad (8)$$

where ft is the average frequency of the word t in the documents that have it, which is p^avg (t)·dld. The risk function is based on the geometric distribution, which models how likely it is to observe a certain number of successes in a sequence of independent trials. The intuition is that the more the document's frequency of the word deviates from the average frequency, the riskier it is to use the average probability.

Now, we can use the risk function to combine the maximum likelihood estimate and the average estimate, using the following formula in equation (9):

$$\hat{p}(t|Md) = \begin{cases} \hat{p}_{ml}(t|Md)\cdot(1-\hat{R}_{t,d}) + \hat{p}_{avg}(t)\cdot\hat{R}_{t,d} & \text{if } tf(t,d) > 0 \\ \frac{cft}{cs} & \text{otherwise} \end{cases} \qquad (9)$$

This formula gives a smoothed estimate of the word probability, which balances between the document-specific and the collection-wide information. If the word is present in the document, it uses a weighted average of the two estimates, where the weight is determined by the risk function. If the word is absent from the document, it uses the collection frequency as a fallback.

Finally, we can use this smoothed estimate to calculate the probability of the query given the document's language model, using the following formula in equation (10):

$$P(Q \mid Md) = \prod_{t\in Q}\hat{p}(t \mid Md)\cdot\prod_{t\notin Q}(1-\hat{p}(t \mid Md)) \qquad (10)$$

This formula considers both the words that match and the words that do not match the query. We use this formula to rank the documents according to their relevance to the query.
Example:
D1: amazing smart phone, good camera, good battery life and have good specification. D2: Good picture quality and camera handling is good.
D3: Good sound quality, good battery life, good for gaming and good performance. Query: (good, and, amazing). Total tokens: 32, Document length (dld) : D1- 12, D2- 8, D3- 12; Here we shows calculations for some tokens:

Maximum likelihood estimates of the probability of term 'good' in document d_i:

Pml(good|md1) = 3/12= 0.25, Pml(camera|md1) =1/12= 0.083 Pml(good|md2) = 2/8= 0.25, Pml(camera|md2) = 1/8= 0.125, Pml(good|md3) = 4/12 = 0.33, Pml(camera|md3) =0/12=0(because no word camera in d3)

Estimate: Pavg(good)= [(3/12)+(2/8)+(4/12)] / 3 = 0.277, Pavg(camera) =[(1/12)+(1/8)] / 2 = 0.104

Risk factor:

R(good|d1) = (1.0/(1.0+(0.277 x 12))) x ((0.277 x 12)/ (1+(0.277 x 12))^{3}) = 0.105
R(good|d2) = (1.0/(1.0+(0.277 x 8))) x ((0.277 x 8)/ (1+(0.277 x 8))^{2}) = 0.148
R(good|d3) = (1.0/(1.0+(0.277 x 12))) x ((0.277 x 12)/ (1+(0.277 x 12))^{4}) = 0.081
R(camera|d1) = (1.0/(1.0+(0.104 x 12))) x ((0.104 x12)/ (1+(0.104 x 12))) = 0.247
R(camera|d2) = (1.0/(1.0+(0.104 x 8))) x ((0.104 x8)/ (1+(0.104 x 8))) = 0.249

Now we will use this risk function as a mixing parameter in our calculation of P(t|md1).

P(camera|md1)= (0.083)$^{(1.0 -0.247)}$ x (0.104)$^{(0.247)}$ = 0.087, P(camera|md2)= (0.125)$^{(1.0 -0.249)}$ x (0.104)$^{(0.249)}$ = 0.119, P(camera|md3)= 2/32 =0.063

Like this we must calculate for every token in whole corpus. The results are shown in Table-5.

Table 5: Score of terms in document at each stage of LM model

Token	Pml(t|m d1)	Pml(t|m d2)	Pml(t|m d3)	Pavg (t)	R(t|d 1)	R(t|d 2)	R(t|d 3)	P(t|m d1)	P(t|m d2)	P(t|m d3)
Amazing	0.083	0	0	0.083	0.25	-	-	0.083	0.031	0.031
Smart	0.083	0	0	0.083	0.25	-	-	0.083	0.031	0.031
Phone	0.083	0	0	0.083	0.25	-	-	0.083	0.031	0.031
Good	0.25	0.25	0.33	0.277	0.105	0.148	0.081	0.252	0.254	0.325
Camera	0.083	0.125	0	0.104	0.249	0.249	-	0.087	0.119	0.063
Battery	0.083	0	0.083	0.083	0.25	-	0.25	0.083	0.063	0.083
Life	0.083	0	0.083	0.083	0.25	-	0.25	0.083	0.063	0.083
And	0.083	0.125	0.083	0.097	0.136	0.246	0.249	0.084	0.093	0.083
Have	0.083	0	0	0.083	0.25	-	-	0.083	0.031	0.031
Specificat	0.083	0	0	0.083	0.25	-	-	0.083	0.031	0.031

ion										
Picture	0	0.125	0	0.125	-	0.25	-	0.031	0.125	0.031
Quality	0	0.125	0.083	0.069	-	0.229	0.248	0.063	0.086	0.079
Handling	0	0.125	0	0.125	-	0.25	-	0.031	0.125	0.031
Is	0	0.125	0	0.125	-	0.25	-	0.031	0.125	0.031
Sound	0	0	0.083	0.083	-	-	0.25	0.031	0.031	0.083
For	0	0	0.083	0.083	-	-	0.25	0.031	0.031	0.083
Gamming	0	0	0.083	0.083	-	-	0.25	0.031	0.031	0.083
Performance	0	0	0.083	0.083	-	-	0.25	0.031	0.031	0.083

$P(Q|md1) = (0.083 \times 0.252 \times 0.084) \times (1-0.083)^6 \times (1- 0.087) \times (1- 0.031)^7 \times (1-0.063) = 0.00073$

$P(Q|md2) = (0.031 \times 0.254 \times 0.093) \times (1-0.031)^8 \times (1-0.119) \times (1-0.063)^2 \times (1-0.125)^3 \times (1-0.086) = 0.00027$, $P(Q|md3) = (0.031 \times 0.325 \times 0.083) \times (1-0.031)^7 \times (1-0.063) \times (1-0.079) \times (1-0.083)^6 = 0.00035$

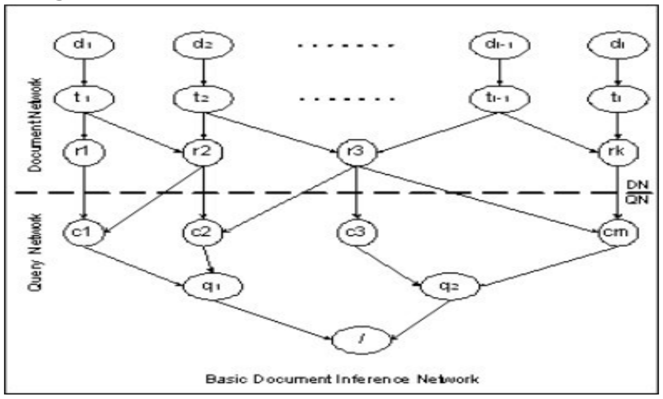
From this calculation our given query is more relevant to document1. Then document 3, then document2.

2.3 Combining Evidence

2.3.1 Inference Network Model: A Bayesian inference network is a graphical model that uses Bayesian logic to compute the probabilities of different outcomes based on the available evidence. It is composed of nodes and edges, where nodes represent variables or constants, and edges represent causal or conditional relationships between variables. A node p that causes or affects another node q is connected by a directed edge (->) from p to q. The graph is acyclic, meaning that there are no loops or cycles in the network.

A Bayesian inference network can be used for ranking purposes, where it combines multiple sources of evidence to determine the relevance of different items. As shown in Figure 3, the network consists of two sub-networks: the document network and the query network [8].

Figure 3: Basic document inference network



Document Network: The document network represents the items that we want to rank, such as documents, web pages, or products. It has three types of nodes: document nodes (di’s), document content nodes (ti’s), and concept nodes (rk’s). The document nodes are the items that we want to rank. The document content nodes are the words or features that describe the items. The concept nodes are the abstract or latent topics that capture the meaning of the items. The edges between the nodes have weights or probabilities that indicate how strongly the nodes are related. The value of a node depends on the values of its parent nodes and the probabilities on the edges [15].

Query Network: The query network represents the user’s information need, which is expressed as a text query. It has three types of nodes: query concept nodes, query operator nodes, and a final leaf node. The query concept nodes are the words or features that describe the user’s need. The query operator nodes are the logical operators that combine the query concepts, such as AND, OR, or NOT. The final leaf node is the user’s information need, which is the goal of the network [1].

To rank the items, the query network is attached to the document network, forming a complete inference network. The attachment is done by matching the concepts in both networks. Then, the network is evaluated for each document node, computing the probability of the document being relevant to the query. The evaluation is done by setting one document node to one and the rest to zero, and propagating the values through the network. The probability of document relevance is obtained from the final node I, and this is used to rank the documents [4].

For example, Let us say we have a collection of documents about different topics, such as sports, politics, science, etc. We want to rank the documents according to how relevant they are to a query, such as “Who won the Nobel Prize in Physics in 2020?”.

The document network has three types of nodes: document nodes (D1, D2, …, Dn), topic nodes (T1, T2, …, Tm), and word nodes (W1, W2, …, Wk). The document nodes represent the documents that we want to rank. The topic nodes represent the latent topics that the documents belong to, such as sports, politics, science, etc. The word nodes represent the words that appear in the documents, such as “Nobel”, “Prize”, “Physics”, etc.

The edges between the nodes have probabilities that indicate how strongly the nodes are related. For example, the edge from T1 to D1 has a probability of 0.8, which means that document D1 has an 80% chance of belonging to topic T1. The edge from W1 to T1 has a probability of 0.6, which means that the word W1 has a 60% chance of appearing in a document that belongs to topic T1[2].

The query network has two types of nodes: query word nodes (Q1, Q2, …, Ql) and a final leaf node (I). The query word nodes represent the words that appear in the query, such as “Who”, “won”, “Nobel”, etc. The final leaf node represents the user’s information need, which is the goal of the network.

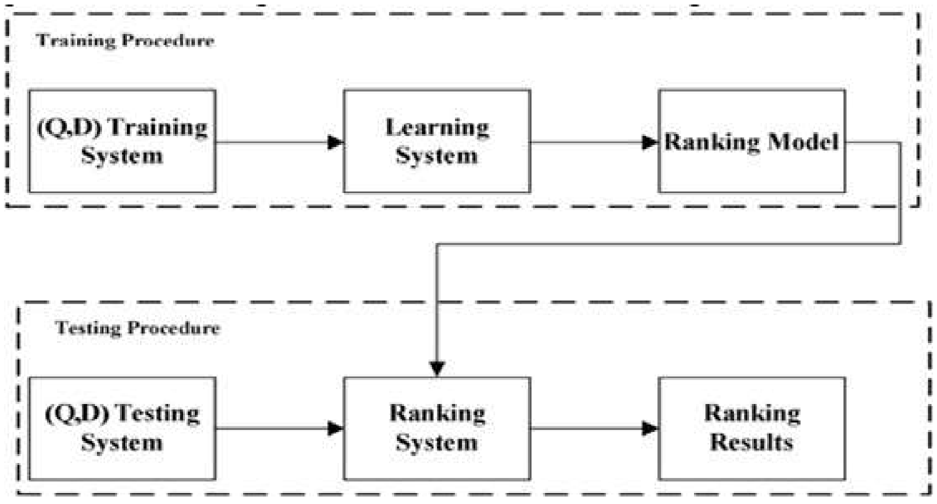
The edges between the nodes have probabilities that indicate how likely the query words are to match the document words. For example, the edge from Q1 to W1 has a probability of 0.9, which means that the query word Q1 has a 90% chance of matching the document word W1.

To rank the documents, the query network is attached to the document network, forming a complete inference network. The attachment is done by matching the query words to the document words. For example, Q1 is matched to W1, Q2 is matched to W2, etc [6].

Then, the network is evaluated for each document node, computing the probability of the document being relevant to the query. The evaluation is done by setting one document node to one and the rest to zero, and propagating the values through the network. The probability of document relevance is obtained from the final node I, and this is used to rank the documents. For example, to evaluate the relevance of document D1, we set D1 to one and the rest of the document nodes to zero. Then, we calculate the values of the topic nodes, the word nodes, the query word nodes, and the final node I, using the probabilities on the edges. The value of the final node I is the probability of document D1 being relevant to the query, which is 0.324 in this case. We repeat this process for each document node and rank them according to their probabilities [8].

2.3.2 Learning to Rank Model: Learning to rank (LTR) is a branch of machine learning that deals with ranking problems in search relevance. Ranking problems are those where the goal is to order a set of items according to some criteria, such as relevance to a query, popularity, or quality. LTR methods use supervised learning techniques to learn a ranking function F from training data, which consists of queries and documents with some relevance scores or judgments. The ranking function F can then be used to rank new documents for new queries, based on some performance measures P and some requirements R [9]. The general process of learning to rank model can be illustrated in Figure -4.

Figure-4: procedure to building learn to rank model



There are three main types of LTR methods, depending on how they treat the ranking problem: pointwise, pairwise, and listwise. Pointwise: Pointwise methods treat the ranking problem as a regression or classification problem, where each document is assigned a score or a label based on how relevant it is to the query. The documents are then sorted by their scores or labels to produce a ranking list. The advantage of pointwise methods is that they are simple and efficient, but the disadvantage is that they ignore the dependencies and interactions between the documents in the ranking list. Pairwise: Pairwise methods treat the ranking problem as a preference learning problem, where the goal is to learn the relative order of pairs of documents for a given query. The documents are compared in pairs, and a binary judgment is made on which one is more relevant [9]. The ranking function F is then learned to minimize the number of disagreements or inversions between the predicted and the true preferences.

The advantage of pairwise methods is that they capture the ordinal nature of the ranking problem, but the disadvantage is that they still ignore the global structure of the ranking list [22]. Listwise methods treat the ranking problem as a structured learning problem, where the goal is to learn the optimal ranking list for a given query. The ranking function F is learned to directly optimize a list-level performance measure, such as normalized discounted cumulative gain (NDCG) or expected reciprocal rank (ERR). The advantage of listwise methods is that they consider the whole ranking list as a unit, but the disadvantage is that they are more complex and computationally expensive than the other methods [7,8].

3. COMPARISION OF INFORMATION RETRIEVAL MODELS

The provided table outlines a comparison of four different Information Retrieval (IR) models: Boolean Model, Vector Space Model (VSM), BM25, and Language Modeling (LM). Detailed description of each model is shown in Table 6:

Table-6: comparison of IR models

Properties	**Boolean**	**VSM**	**BM25**	**LM**
Matching w.r.t query	Exact matching	Partial matching	Partial matching	Partial matching
Precision	Retrieve document only if query exactly matches to that document.	Retrieve according to document weight.	Retrieve according to term occurrence probability w.r.t individual documents	Retrieve according to term occurrence probability w.r.t individual documents and total documents
Ranking	There is no ranking in Boolean model	Ranking provided according to similarity between document and query	ranks a set of documents based on the query terms appearing in each document	ranks a set of documents based on the query terms appearing in each document and whole corpus.
Real time applications	Boolean model gives average performances in real time applications	Vector space model give better performance than Boolean model in real time system because it uses partial matching.	BM25 model gives good performance in real time applications.	LM model gives better performance than BM25 model because there is no zero probability in document score.

Pros	Clear formalism and easy to implement.	Simple and fast model based on linear algebra and allows computing a continuous degree of similarity between queries and documents.	It performs very well in many ad-hoc retrieval tasks, especially those designed by TREC.	Conceptually simple and explanatory, formal mathematical model and Natural use of collection statistics, not heuristics (almost …)
Cons	Exact matching may retrieve too many or too less numb of documents, Difficult to rank and Difficult to translate a query into a Boolean expression.	Long documents are poorly represented, because they have poor similarity values and documents with similar context but different term vocabulary won't be associated, resulting in a "false negative match".	It is full of heuristics. This fact makes it hard to extend this framework. This is why the language modeling framework becomes popular.	Our language models are accurate representations of the data. have some sense of term distribution

4. IR Metrics:

Information retrieval (IR) is used to fetching the relevant documents to given query. Here we discus about list evaluation metric that can be used to measure performance of IR system [10].

4.1 Metrics for unranked retrieval systems:

All these definitions are same as that for classification. However they have some specific characteristics for IR system.

Precision: Precision is defined as the fraction of the number of relevant and retrieved documents to the number of total retrieved documents from the query [11].

Precision = (relevant items retrieved) / (retrieved items)

Recall: Recall is defined as fraction of the number of retrieved and relevant documents to the number of possible relevant documents.

Recall = (relevant items retrieved)/ (relevant items)

F-Measure: F-Measure is a weighted harmonic mean of precision and recall.

$$F = 2PR/ (P+R)$$

where P represents Precision and R represents recall.

4.2 Metrics for ranked retrieval system:

Precision recall curves: In classification we plot ROC by changing the threshold values of binary classification. In IR system we plot it by changing no of documents retrieved.

11 point interpolated average precision: For the recall values of (0,0.1,0.2,…,0.9,1.0) we have to find the precision and average it.

NDCG: Normalized Discounted Cumulative Gain (NDCG), It contains cumulative gain at each position which is discount by position and is normalized. NDCG range in between 0 to 1. For perfect ranking model value of NDCG is 1.

5. RESULTS:

In this paper we use phone reviews datasets of size 500 and 1000 documents, which are collected from Kaggle. Unranked IR metrics for all models shown in Table 7:

Table 7: Results comparison of IR models using unranked metrics

Algorithm s/ Metrics	Boolean		VSM		BM25		LM	
	1000 docs	500 docs	1000 docs	500 docs	1000 docs	500 docs	1000 docs	500 docs
Prcesion	0.17	0.20	0.35	0.30	0.52	0.50	0.56	0.53
Recall	0.25	0.25	0.52	0.37	0.67	0.44	0.68	0.51
f-measure	0.21	0.22	0.42	0.33	0.58	0.47	0.59	0.52
Relavent	450	321	450	321	450	321	450	321
Retrived	664	399	664	280	580	290	597	310
Rrd	115	81	238	140	300	161	308	166

The table compares the performance of four Information Retrieval (IR) models (Boolean, VSM, BM25, LM) across different document sizes (1000, 500) using unranked metrics such as Precision, Recall, and F-measure. BM25 consistently shows higher Precision and Recall values, indicating balanced performance. The Relevant and Retrieved documents (Rrd) column provides insights into the overlap between relevant and retrieved documents[26]. These metrics offer a concise evaluation of each model's effectiveness under varying conditions, aiding in informed model selection for specific retrieval goals [11].

Results for ranked IR models:

Table-8 presents a comprehensive comparison of the ranked precision and recall values for three Information Retrieval (IR) models, VSM, BM25, and LM across two scenarios with 500 and 1000 documents. Each column corresponds to a specific rank, while the rows depict precision and recall values for each model at that rank. Notably, VSM consistently achieves the highest precision in the 500-document scenario, while BM25 exhibits strong and consistent performance, particularly in the 1000-document setting. LM also demonstrates competitive precision and recall values, showcasing its effectiveness. The table offers a nuanced view of how these models perform at different ranks, aiding in the understanding of their strengths and weaknesses in ranked information retrieval tasks [15].

Table-8: Rank wise precision and recall values of different IR models.

	500 documents	1000 documents

Rank	VSM		BM25		LM		VSM		BM25		LM	
	precision	Recall	precision	Recall	precision	Recall	precision	recall	precision	recall	precision	Recall
1	1.0	0.20	1.0	0.14	1.0	0.14	1	0.25	1	0.17	1	0.14
2	0.50	0.20	0.50	0.14	0.50	0.14	0.5	0.25	0.50	0.17	1	0.29
3	0.33	0.20	0.33	0.14	0.67	0.28	0.67	0.5	0.33	0.17	1	0.43
4	0.25	0.20	0.25	0.14	0.75	0.42	0.75	0.75	0.50	0.33	1	0.57
5	0.40	0.40	0.40	0.28	0.60	0.42	0.6	0.75	0.60	0.50	0.80	0.57
6	0.50	0.60	0.50	0.42	0.67	0.57	0.5	0.75	0.67	0.67	0.83	0.71
7	0.42	0.60	0.57	0.57	0.71	0.71	0.43	0.75	0.71	0.83	0.71	0.71
8	0.50	0.80	0.62	0.71	0.62	0.71	0.38	0.75	0.62	0.83	0.75	0.85
9	0.55	1.0	0.66	0.85	0.67	0.85	0.44	1	0.56	0.83	0.77	1
10	0.50	1.0	0.70	1.0	0.70	1.0	0.4	1	0.60	1	0.70	1

11 point interpolated average precision

Table-9 provides the precision-recall values at specific recall points for two scenarios with 500 and 1000 documents, representing the performance of ranked IR models—Vector Space Model (VSM) and BM25. The recall values range from 0.0 to 1.0, and corresponding precision values for each model are reported

at these recall points. Notably, the table showcases how the precision-recall curve evolves for both models[12]. The 11-point Interpolated Average Precision (AP) is calculated to summarize the overall performance. BM25 consistently demonstrates higher precision values across various recall points, resulting in a higher overall AP compared to VSM. The values indicate the effectiveness of BM25 in balancing precision and recall at different thresholds, as reflected in the precision-recall curve depicted in Figure 5. This detailed analysis provides insights into the models' performance across the spectrum of recall levels, aiding in a nuanced evaluation of their effectiveness in ranked retrieval tasks [24].

Table -9: 11-point precision recall values of ranked IR models

	500 documents	1000 documents		500 documents	1000 documents	
Recall	VSM	BM25	Recall	VSM	BM25	Recall
0.0	1	1	0.0	1	1	0.0
0.1	1	1	0.1	1	1	0.1
0.2	0.55	0.70	0.2	0.55	0.70	0.2
0.3	0.55	0.70	0.3	0.55	0.70	0.3
0.4	0.55	0.70	0.75	0.75	0.71	1
0.5	0.55	0.70	0.71	0.60	0.71	0.83

0.6	0.55	0.70	0.71	0.50	0.71	0.83
0.7	0.55	0.70	0.71	0.44	0.71	0.77
0.8	0.55	0.70	0.70	0.44	0.71	0.77
0.9	0.55	0.70	0.70	0.44	0.62	0.77
1.0	0.5	0.70	0.70	0.40	0.62	0.70
AP	0.63	0.75	0.77	0.64	0.75	0.87

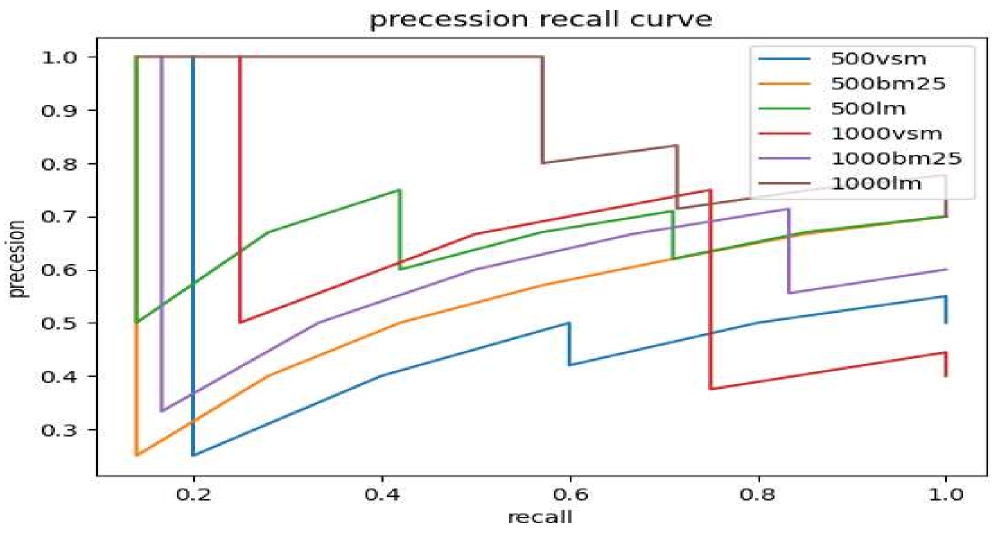


Figure 5: precision-recall curve

Table 10 presents a metrics comparison of Information Retrieval (IR) models—VSM, BM25, and LM. based on the NDCG score. The NDCG scores are calculated for the top 10 ranking documents of each model across different document counts, ranging from 100 to 1000. The scores indicate the effectiveness of each model in retrieving relevant documents as the dataset size increases. BM25 consistently achieves high NDCG scores across various document counts, suggesting its robust performance in ranking relevant documents. VSM and LM also exhibit competitive scores, with all models demonstrating an improvement in NDCG as the document count increases. The accompanying Figure 6, a bar chart, visually represents this metrics comparison, providing a clear overview of how the NDCG scores vary for each IR model across different dataset sizes. Overall, this analysis offers valuable insights into the models' relative effectiveness in handling varying document volumes [20,25].

Table 10: Metrics Comparison of IR models with different count of dataset.

Doc count/ Algorithms	100	200	300	400	500	600	700	800	900	1000
VSM	0.874	0.880	0.898	0.940	0.938	0.953	0.967	0.971	0.986	0.993
BM25	0.888	0.890	0.942	0.945	0.956	0.967	0.983	0.986	0.984	0.993
LM	0.893	0.897	0.944	0.949	0.967	0.971	0.983	0.989	0.989	0.997

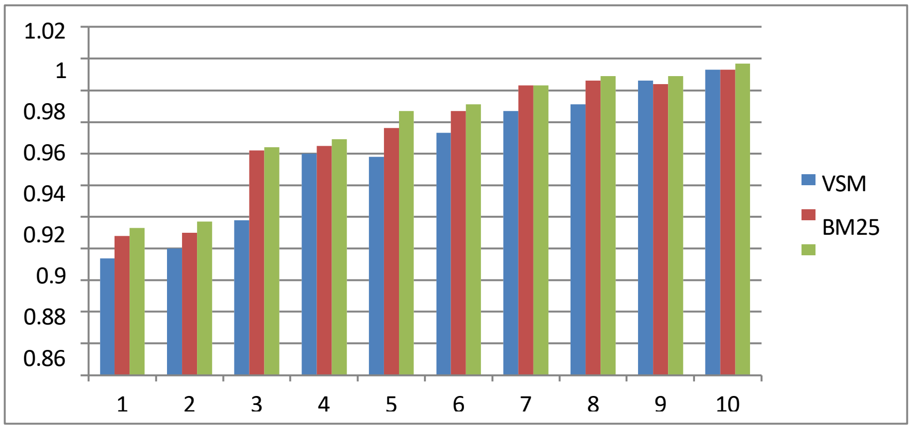


Figure 6: Bar chat of metrics comparison of IR models

6. CONCLUSION

In conclusion, Information Retrieval (IR) plays a crucial role in efficiently retrieving data from extensive document collections based on user needs. This paper extensively explores various IR models employed for this purpose. The initial focus involves defining Information Retrieval and outlining the classification of IR models. Subsequently, each category is detailed, providing an explanation of methods and elucidating Boolean and probabilistic models through illustrative examples. A comparative analysis between Boolean and probabilistic models is then conducted. Practical implementation of these models is demonstrated using Python on datasets containing phone reviews of varying sizes, and the results are thoroughly analyzed. The comparison involves both ranked and unranked IR metrics. The findings indicate that the Boolean model is effective for small to medium-sized datasets, while probabilistic models outperform Boolean models. Specifically, the BM25 model demonstrates faster retrieval compared to the LM model, but LM retrieves a higher number of relevant documents compared to BM25. This comprehensive exploration sheds light on the strengths and limitations of different IR models, providing valuable insights for their practical application in real-world scenarios.

7. REFERENCES

[1] Doaa Mabrouk, Sherine Rady, Nagwa Badr and M.E.Khalifa; ”A Survey on Information Retrieval Systems’ Modeling using Term Dependencies and term weighting”; IEEE International Conference on Intelligent Computing and Information Systems (ICICIS 2017).

[2] Arash Habibi Lashkari, Fereshteh Mahdavi and Vahid Ghomi; “A Boolean Model in Information Retrieval For Search Engines”; 2009 International Conference on Information Management and Engineering; IEEE DOI 10.1109/ICIME.2009.101.

[3] Naol Bakala; “Information Retrieval System By Using Vector Space Model”; International Journal Of Scientific & Technology Research Volume 8, Issue 10, October 2019 ISSN 2277- 8616.

[4] “Information retrieval document search using vector space model in R” ; Posted by data perspective on November 15, 2017

[5] Stephen Robertson and Hugo Zaragoza; “The Probabilistic Relevance Framework: BM25

and Beyond", Foundations and Trends in Information Retrieval Vol. 3, No. 4 (2009) 333–389 @ 2009 S. Robertson and H. Zaragoza DOI: 10.1561/1500000019.

[6] Shane Connelly; "Practical BM25 - Part 2: The BM25 Algorithm and its Variables"; APRIL 2018.

[7] Jay M.Ponte and W.Bruce Croft; " A Language Modeling Approach to Information Retrieval"; SIGIR'98, Mellbourne, Australia @ 1998 ACM 1-58113-015-58/98.

[8] Adi Wahyu and Zainal Arifin Hasibuan; "Implementing Inference Networks for Information Retrieval System in Indonesian Language"; Conference: iiWAS'2003 - The Fifth International Conference on Information Integrationand Web-based Applications Services, 15-17 September 2003, Jakarta, Indonesia.

[9] Xishuang Dong, Xiaodong Chen, Yi Guan, Zhiming Xu and Sheng Li; "An Overview of Learning to Rank for Information Retrieval"; 2009 World Congress on Computer Science and Information Engineering; 978-0-7695-3507-4/08 $25.00 © 2008 IEEE DOI 10.1109/CSIE.2009.1090.

[10] datastoriesweb.wordpress.com/2019/11/20/ir-metrics

[11] towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52.

[12] Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. Journal of machine Learning research 2003, 3, 993–1022.

[13] Robertson, S.E.; Walker, S.; Jones, S.; Hancock-Beaulieu, M.M.; Gatford, M.; others. Okapi at TREC-3. Nist Special Publication Sp 1995, 109, 109.

[14] Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; Hullender, G. Learning to rank using gradient descent. Proceedings of the 22nd international conference on Machine learning, 2005, pp. 89–96.

[15] Burges, C.; Ragno, R.; Le, Q. Learning to rank with nonsmooth cost functions. Advances in neural information processing systems 2006, 19.

[16] Guo, J.; Fan, Y.; Ai, Q.; Croft, W.B. A deep relevance matching model for ad-hoc retrieval. Proceedings of the 25th ACM international on conference on information and knowledge management, 2016, pp. 55–64.

[17] Mitra, B.; Diaz, F.; Craswell, N. Learning to match using local and distributed representations of text for web search. Proceedings of the 26th international conference on world wide web, 2017, pp. 1291–1299.

[18] Zhou, J.; Agichtein, E. Rlirank: Learning to rank with reinforcement learning for dynamic search. Proceedings of The Web Conference 2020, 2020, pp. 2842–2848.

[19] MacAvaney, S.; Yates, A.; Cohan, A.; Goharian, N. CEDR: Contextualized embeddings for document ranking. Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, 2019, pp. 1101–1104.

[20] Li, J.; Zeng, H.; Peng, L.; Zhu, J.; Liu, Z. Learning to rank method combining multi-head self-attention with conditional generative adversarial nets. Array 2022, 15, 100205.

[21] P. K. Lakineni, R. Singh, B. Mandaloju, S. Singhal, M. D. Bajpai and M. Tiwari, "A Cloud-Based Healthcare Diagnosis Support Network for Smart IoT for Predicting Chronic Kidney Failure," 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2023, pp. 1858-1863, doi: 10.1109/ICACITE57410.2023.10183151.

[22] P. K. Lakineni, K. M. Nayak, H. Pallathadka, K. Gulati, K. Pandey and P. J. Patel, "Fraud Detection in Credit Card Data using Unsupervised & Supervised Machine Learning-Based Algorithms," 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), Chennai, India, 2022, pp. 1-4, doi: 10.1109/ICSES55317.2022.9914287.

[23] V. Dankan Gowda, K. Prasad, R. Shekhar, R. Srinivas, K. N. V. Srinivas and P. K. Lakineni, "Development of a Real-time Location Monitoring App with Emergency Alert Features for Android Devices," 2023 4th IEEE Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2023, pp. 1-8, doi: 10.1109/GCAT59970.2023.10353310.

[24] Lakineni, P.K., Balamanigandan, R., Rajesh Kumar, T., Sathyendra Kumar, V., Mahaveerakannan, R., Swetha, C. (2023). Securing the E-records of Patient Data Using the Hybrid Encryption Model with Okamoto–Uchiyama Cryptosystem in Smart Healthcare. In: Swaroop, A., Polkowski, Z., Correia, S.D., Virdee, B. (eds) Proceedings of Data Analytics and Management. ICDAM 2023. Lecture Notes in Networks and Systems, vol 788. Springer, Singapore. https://doi.org/10.1007/978-981-99-6553-3_38

[25] D. G. V, V. N. Prasad, K. Prasad, V. S. Prasad, Y. Mahajan and S. Suneetha, "A Cloud-Based UV Monitoring System for Remote Real-Time UV Exposure Tracking," 2023 4th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2023, pp. 1764-1770, doi: 10.1109/ICOSEC58147.2023.10276360.

[26] R. M, Reeta, R. Singh, N. K. Gupta, S. Adhav and S. Suneetha, "Comparative Analysis Of Machine Learning And Its Powered Technologies Applications In The Banking Sector," 2023 6th International Conference on Contemporary Computing and Informatics (IC3I), Gautam Buddha Nagar, India, 2023, pp. 2415-2420, doi: 10.1109/IC3I59117.2023.10397904.

[27] S. Suneetha and S. V. Row, "Aspect-based sentiment analysis: A comprehensive survey of techniques and applications", *J. Data Acquisition Process.*, vol. 38, no. 3, pp. 177-203, 2023.

[28] L. Prasanna Kumar and S. Suneetha, "Optimization Scheme for Storing and Accessing Huge Number of Small Files on HADOOP Distributed File System", *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 4, no. 2, pp. 315-319, 2016.