**TYBSc. CS Sem- VI**

**WRITE UP**

**INFORMATION RETRIEVAL**

**PRACTICAL NO 1**

**Aim:** Document Indexing and Retrieval

* Implement an inverted index construction algorithm.
* Build a simple document retrieval system using the constructed index.

**Theory :**

● An Inverted Index is a data structure used in information retrieval

systems to efficiently retrieve documents or web pages containing a

specific term or set of terms.

● In an inverted index, the index is organized by terms (words), and each

term points to a list of documents or web pages that contain that term.

● Inverted indexes are widely used in search engines, database systems, and

other applications where efficient text search is required.

● They are especially useful for large collections of documents, where

searching through all the documents would be prohibitively slow. An

inverted index is an index data structure storing a mapping from content,

such as words or numbers, to its locations in a document or a set of

documents.

**Rules to create an inverted index -**

1) The text of each document is first preprocessed by removing stop words :

Stop words are the most occurring and useless words in documents like

“I”, “the”, “we”, “is”, and “an”.

2) The text is tokenized, meaning that it is split into individual terms.

3) The terms are then added to the index, with each term pointing to the

documents in which it appears.

**Practical No 2**

**Aim: Retrieval Models**

* Implement the Boolean retrieval model and process queries.
* Implement the vector space model with TF-IDF weighting and cosine similarity.

**Theory: A) Boolean Retrieval Model -**

● A Boolean model is a fundamental concept in Information Retrieval (IR) that is used to represent and retrieve documents or information based on Boolean logic.

● In this model, a document is typically represented as a set of terms (words or phrases), and queries are also represented using Boolean operators (AND, OR, NOT) to specify the desired information.

Here's how the Boolean model works in IR:

**1. Document Representation:** Each document in the collection is represented as a set of terms. These terms can be extracted from the document's content and can be single words, phrases, or other units of information.

**2. Query Representation:** Queries are also represented as sets of terms, and Boolean operators (AND, OR, NOT) are used to combine these terms to express the user's information needs. For example, a query might be "cats AND dogs," meaning the user wants documents that contain both "cats" and "dogs."

**3. Boolean Operators:**

● AND: "cats AND dogs,"

both "cats" and "dogs" will be retrieved.

● OR: "cats OR dogs,"

"cats" or "dogs" or both will be retrieved.

● NOT: "cats NOT dogs"

"cats" but not "dogs."

**B) TF-IDF**

● Term Frequency - Inverse Document Frequency (TF-IDF) is a

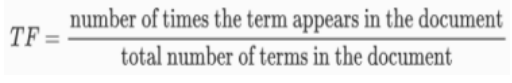
widely used statistical method in information retrieval.

● It measures how important a term is within a document

relative to a collection of documents.

Term Frequency (TF): TF of a term or word is the number of times the

term appears in a document compared to the total number of words in

the document.

**Inverse Document Frequency(IDF):**

● IDF of a term reflects the proportion of documents in the corpus

that contains the term.

IDF = log( N / df ) where,

N= total no. of documents

df = no. of documents containing a term

● The TF-IDF of a term is calculated by multiplying TF and IDF

scores. TF-IDF = TF\*IDE

**C) Cosine Similarity -**

● Cosine similarity is a measure of similarity between two non-

zero vectors defined in an inner product space.

● Cosine similarity is the cosine of the angle between the

vectors.

● The cosine similarity always belongs to the

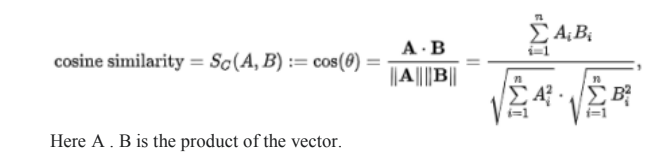
interval [−1,1].

● In cosine similarity, data objects in a dataset are treated as a

vector.

● The formula to find the cosine similarity between two

vectors is -



**PRACTICAL NO 3**

**Aim:** Spelling Correction in IR Systems

* Develop a spelling correction module using edit distance algorithms.
* Integrate the spelling correction module into an information retrieval system.

**Theory:**

**Edit Distance :**

● Edit distance is a measure of the similarity between two strings by calculating the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into the other.

● The smaller the edit distance, the more similar the strings are. Consider two strings str1 and str2 of length M and N respectively.

For finding edit distance there are performed below operations -

**1. Operation 1 (INSERT):** Insert any character before or after any index value

**2. Operation 2 (REMOVE):** Remove a character

**3. Operation 3 (Replace):** Replace a character at any index value with some other character

**Practical no 4**

**Aim:** Evaluation Metrics for IR Systems

* Calculate precision, recall, and F-measure for a given set of retrieval results.
* Use an evaluation toolkit to measure average precision and other evaluation metrics.

**Theory: 1. Precision:**

**●** Precision is the ratio of correctly predicted positive observations to

the total predicted positives.

● It is also called Positive Predictive Value (PPV).

● Precision is calculated using the following formula:

Precision = TP / TP+FP

Where:

• TP (True Positives) is the number of instances correctly predicted as positive.

• FP (False Positives) is the number of instances incorrectly predicted as positive.

High precision indicates that the model has a low rate of false positives. In other words, when the model predicts a positive result, it is likely to be correct.

**2. Recall:**

• Recall is the ratio of correctly predicted positive observations to all observations in actual class.

• Recall is calculated using the following formula:

Recall= TP /TP+FN

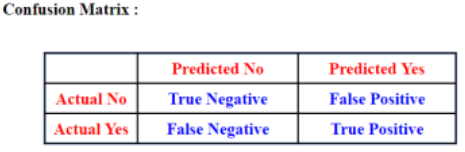
Where:

• TP (True Positives) is the number of instances correctly

predicted as positive.

• FN (False Negatives) is the number of instances incorrectly predicted as negative.

High recall indicates that the model has a low rate of false negatives. In other words, the model is effective at capturing all the positive instances.

****

**3. F-measure:**

**•** The F-measure is a metric commonly used in performance evaluation.

• It combines precision and recall into a single value, providing a balanced measure of a model's performance.

• The formula for F-measure is:

****

**•** The F-measure ranges from 0 to 1, where 1 indicates perfect precision and recall.

**4. Average Precision:** Average Precision is used to find the Average of the model precision based on relevancy of result. • Algorithm:

In order to find Average Precision:

1) Take 2 variables X and Y as 0

2) We will then go through the prediction from left to right:

3) In case the prediction is 0, we will only increment Y by 1 and not find prediction score

4) In case the prediction is 1, we will increment both X and Y by 1

5) After incrementing, we use the formula X/Y to get the current position prediction score.

6) Lastly we will find summation of all prediction scores and divide them by a total number of positive predictions.

**Practical no 5**

**Aim:** Text Categorization

* Implement a text classification algorithm (e.g., Naive Bayes or Support Vector Machines).
* Train the classifier on a labeled dataset and evaluate its performance.

**Theory:**

Text categorization is the process of sorting text documents into one or more predefined categories or classes of similar documents. Differences in the results of such categorization arise from the feature set chosen to base the association of a given document with a given category. Advocates of text categorization recognize that the sorting of text documents into categories of like documents reduces the overhead required for fast retrieval of such documents and provides smaller domains in which the users may explore similar documents. In this paper we are interested in examining whether automatic classification of news texts can be improved by prefiltering the vocabulary to reduce the feature set used in the computations. First we compare artificial neural networks and support vector machine algorithms for use as text classifiers of news items. Secondly, we identify a reduction in the feature set that provides improved results.

**Practical no 6**

**Aim:** Clustering for Information Retrieval

* Implement a clustering algorithm (e.g., K-means or hierarchical clustering).
* Apply the clustering algorithm to a set of documents and evaluate the clustering results.

**Theory: K-Means Clustering:**

● K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters.

● Here K defines the number of predefined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

● It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

● The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

● The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

● The k-means clustering algorithm mainly performs two tasks:

1. Determines the best value for K center points or centroids by an iterative process.

2. Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

**Working of K-means Algorithm -**

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be different from the input

dataset).

Step-3: Assign each data point to their closest centroid, which will form the

predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means assign each datapoint to the new

closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

**Practical No 7**

**Aim:** Web Crawling and Indexing

* Develop a web crawler to fetch and index web pages.
* Handle challenges such as robots.txt, dynamic content, and crawling delays.

**Theory:**

**Crawling:** Google downloads text, images, and videos from pages it found on the internet with automated programs called crawlers.

**Indexing:** Google analyzes the text, images, and video files on the page, and stores the information in the Google index, which is a large database.

**Crawling Process -**

**1) Starting Point:** The crawling process usually begins with a set of seed URLs, which can be provided manually or generated through algorithms. These URLs serve as the starting points for the web crawlers.

**2) Queue of URLs:** Web crawlers maintain a queue of URLs, often referred to the URL frontier. This queue represents the set of URLs that the crawler is yet to visit. New URLs are continuously added to this queue during the crawling process.

**3) Parsing Content:** When a web crawler visits a web page, it parses the HTML content to extract links (URLs) embedded within the page. This process involves examining HTML tags and, in some cases, executing JavaScript to discover additional links.

**4) Respecting Directives:** Crawlers adhere to rules specified in the robots.txt file, which indicates areas of a website that should not be crawled. Additionally, crawlers may implement policies to filter out certain types of content or URLs.

**5) Avoiding Redundancy:** To prevent redundancy and ensure efficient crawling, duplicate URLs are often identified and removed from the crawling queue.

**6) Traversal Strategy:** Crawlers can follow various traversal strategies, such as depth-first or breadth-first, as they explore the web. The chosen strategy determines the order in which pages are crawled.

**7) Politeness:** To avoid overloading web servers with too many requests, crawlers may introduce a crawl delay between successive requests.

**8) Retrieving Web Pages:** Crawlers download the content of web pages, including HTML, text, images, and other resources. The downloaded content is then processed for indexing.

**Indexing Process -**

**1) HTML Parsing:** The content retrieved by the crawler is parsed to extract relevant information. This involves analyzing the HTML structure to identify text, metadata, and other elements on the page.

**2) Isolating Textual Content:** From the parsed content, the crawler isolates the textual information, such as the body of the page, headings, and other relevant textual data.

**3) Breaking into Tokens:** The textual content is tokenized, breaking it down into smaller units, typically words or phrases. Tokenization facilitates efficient indexing and retrieval based on keywords.

**4) Data Organization:** The extracted information is organized into an index, which is a structured database allowing for fast and efficient retrieval. The index includes details about keywords, their locations, and other relevant metadata.

**5) Optimizing for Retrieval:** The index is often organized as an inverted index, mapping each term to the documents or web pages where it appears. This structure enables quick retrieval of documents containing specific terms.

**6) Additional Information:** In addition to textual content, metadata such as page title, URL, and other relevant details may be included in the index to enhance the search experience.

**7) Real-time Changes:** Search engines continuously update their indexes to reflect changes on the web. This ensures that the search results remain current and accurate.

**Practical No 8**

**Aim: Link Analysis and PageRank**

* Implement the PageRank algorithm to rank web pages based on link analysis.
* Apply the PageRank algorithm to a small web graph and analyze the results.

**Theory: Link Analysis:**

● Link analysis is a method used to examine relationships and connections between entities in a network.

● It involves studying the links or connections between different elements to uncover patterns, structures, and insights.

● Link analysis is commonly applied in various fields, including information retrieval, social network analysis, fraud detection, and recommendation systems.

**PageRank Algorithm -**

● The PageRank algorithm is an algorithm used by the Google search engine to rank web pages in its search results.

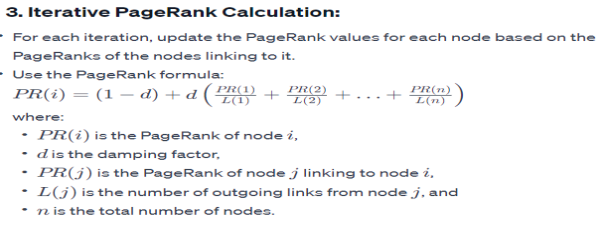
● It was developed by Larry Page and Sergey Brin, the co-founders of Google, and is named after Larry Page.

● PageRank is based on the idea that the importance of a webpage is determined by the number and quality of other pages that link to it.

**Working -**

**1) Initialize PageRank Values:** Set an initial PageRank value for each node. Commonly, this is initialized to 1 divided by the total number of nodes, making the sum of all PageRank values equal to 1.

**2) Define Damping Factor and Iterations:** Choose a damping factor (typically 0.85) to model the probability that the user will continue navigating through the web by following links. Decide on the maximum number of iterations for the algorithm.



**4) Convergence Check:** After each iteration, check for convergence. If the difference between the new and previous PageRank values falls below a certain threshold, the algorithm has converged, and you can stop iterating.

**5) Repeat Iterations:** Continue iterating until the maximum number of iterations is reached or until convergence is achieved.

**6) Final PageRank Values:** The final PageRank values represent the importance of each node in the graph based on the link structure.

**Practical No 9**

**Aim:** Learning to Rank

* Implement a learning to rank algorithm (e.g., RankSVM or RankBoost).
* Train the ranking model using labeled data and evaluate its effectiveness.

**Theory:** Normalized Discounted Cumulative Gain (NDCG) is a ranking quality metric. It compares rankings to an ideal order where all relevant items are at the top of the list.

NDCG at K is determined by dividing the Discounted Cumulative Gain (DCG) by the ideal DCG representing a perfect ranking.

DCG measures the total item relevance in a list with a discount that helps address the diminishing value of items further down the list.

You can aggregate the NDCG results across all users to get an overall measure of the ranking quality in the dataset.

NDCG can take values from 0 to 1, where 1 indicates a match with the ideal order, and lower values represent a lower quality of ranking.

**Step1:** We first generate synthetic labeled data using make\_classification. Replace this with your actual labeled data.

**Step2:** We split the data into training and testing sets using train\_test\_split.

**Step3:** We normalize the features using StandardScaler.

**Step4:** We train a RankSVM model using SVC (Support Vector Classifier) with a linear kernel.

**Step5:** We predict rankings on the test set using the trained model's decision function.

**Step6:** Finally, we evaluate the effectiveness of the model using NDCG (Normalized Discounted Cumulative Gain), a popular metric for ranking evaluation, using ndcg\_score

**Code:**

**from sklear**n.datasets import make\_classification

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import ndcg\_score

# Generate synthetic labeled data (replace with your actual labeled data)

X, y = make\_classification(n\_samples=1000, n\_features=10, n\_classes=2, random\_state=42)

# Split data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Normalize features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Train RankSVM model

ranksvm\_model = SVC(kernel='linear')

ranksvm\_model.fit(X\_train\_scaled, y\_train)

# Predict rankings on test set

y\_pred = ranksvm\_model.decision\_function(X\_test\_scaled)

# Evaluate model effectiveness using NDCG (Normalized Discounted Cumulative Gain)

ndcg = ndcg\_score(y\_test.reshape(1, -1), y\_pred.reshape(1, -1))

print("NDCG Score:", ndcg)