NAME: RANSING ASHISH PETER
ROLL NO: 42557
BRANCH: BE AI&DS
BATCH: B3

★ Computer Laboratory II : Information Retrieval ★

⊙ Assignment - 1 ⊙

★ **Title** : Pre-processing of Text-Document.

★ **Problem Statement** :
- Write a program for pre-processing of a text document such as stop word removal, stemming.

★ **Objectives** :
- To understand the concept and importance of pre-processing in Information Retrieval.
- To implement stop word removal in a text document.
- To perform stemming on a text document to reduce words to their root form.

★ **Pre - Requisite** :
- Basic knowledge of text data processing.
- Familiarity with Python programming and understanding of NLP concepts.

★ **Outcomes** :
- At the end of this practical, we will be able to apply - pre-processing techniques to optimize text documents for efficient information retrieval.

★ THEORY :

- Text pre-processing is a crucial step that transforms raw textual data into a structured format suitable for retrieval and analysis.

⊙ Stop Word Removal →

- Stop words are common words such as "the", "is", "in", "and", etc, which do not carry significant meaning for text analysis.
- By removing stop words, we reduce the size of text data and focus on most important words in document.

⊙ Stemming →

- Stemming is a process that reduces words to their root or base form. For instance, the words "running", "runner", and "runs" are reduced to "run".
- Example →
  Words : playing, played, player
  Stemming : play.
- The most common stemming algorithm is Porter stemmer.

★ ALGORITHM / STEPS :

1] Read the given text document or string for pre-processing.

2] Split the input text into individual words (tokens) using space or punctuation as delimiters.

3] Load a list of stop words. Iterate over the tokenized words and remove any words that are present in stop word list.

4] Apply a stemming algorithm [e.g. Porter Stemmer] to each token to reduce words to their base form.

5] The final output will be a list of stemmed tokens with stop words removed, ready for further processing.

★ CONCLUSION :

- Hence, in this practical, we successfully implemented pre-processing techniques on a text document, including tokenization, stop-word removal and stemming.

NAME: RANSING ASHISH PETER
ROLL NO: 42557
BRANCH: BE AI&DS
BATCH: B3

★ Computer Laboratory II : Information Retrieval ★

⊙ Assignment - 2 ⊙

★ **Title** : Document Retrieval using Inverted Indexing.

★ **Problem Statement** :
- Implement a program for retrieval of documents using inverted files.

★ **Objectives** :
- To understand concept of inverted indexing in Information Retrieval.
- To implement a program that creates an inverted index from a set of documents.
- To retrieve documents based on user queries using inverted index.

★ **Pre-Requisite** :
- Basic knowledge of data structures [dictionaries, or Hash maps].
- Understanding of Information Retrieval concepts particularly inverted index.

★ **Outcomes** :
- At the end of this practical, we will be able to implement document retrieval system using inverted indexing.

★ THEORY :

⊙ Inverted Indexing →

- It is a widely used data structure in information systems that allows for fast and efficient document retrieval.
- Unlike traditional indexing methods that store the document in which each term appears, an inverted index maps each unique term to a list of documents where term appears.
- Key Concepts →

1] Inverted Index Structure →

- Two main components :
  a] Term → Unique word extracted from doc
  b] Posting → List of docs in which term appears.

2] Index Creation →

- Index is built by :
  a] Tokenizing text from each doc.
  b] Removing stop words & using stemming.
  c] Adding each unique term to index with its corresponding doc IDs.

3] Query Processing →

a] Parse user query to extract terms.
b] Looks up each term in inverted index.
c] Merges posting lists to provide final set of docs matching query.

★ ALGORITHM / STEPS :

1] Read multiple text documents.
2] For each document, tokenize the text and perform stop word removal.
3] Create an empty dictionary to store the inverted index.
4] If token is not in inverted index, add it with document ID; if present, append the doc ID to posting list.
5] Read the user's query.
6] Tokenize the query and remove the stop words. For each term in query, Lookup the term in the inverted index to retrieve posting list.
7] Combine the posting lists of query terms to find documents that match all terms.
8] Display the list of docs that match the user query.

★ CONCLUSION :

- Hence, in this practical, we successfully implemented a document retrieval system using inverted indexing which enables efficient & rapid retrieval of documents.

NAME: RANSING ASHISH PETER
ROLL NO: 42557
BRANCH: BE AI&DS
BATCH: B3

★ Computer Laboratory II : Information Retrieval ★

⊙ Assignment - 3 ⊙:

★ **Title** : Diagnosis of Heart Disease Using Bayesian Networks.

★ **Problem Statement** :
- Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set.

★ **Objectives** :
- To understand the fundamentals of Bayesian networks and their applications in medical diagnosis.
- To construct a Bayesian network using Heart Disease Dataset.

★ **Pre-Requisite** :
- Basic knowledge of probability theory.
- Familiarity with Bayesian networks and their components.

★ **Outcome** :
- At the end of this practical, we will be able to construct a Bayesian network model to diagnose heart disease.

* THEORY:

① Bayesian Network →

- A Bayesian network is a probabilistic graphical model that represents a set of variables and their conditional dependencies through a directed acyclic graph [DAG].
- Key Concepts:

1] Nodes and Edges →

- Nodes represent random variables (e.g. symptoms, test results.)
- Directed edges indicate causal relationships between these variables.

2] Conditional Probability Tables [CPTs] →

- Each node has an associated CPT that quantifies the effect of the parent nodes on the node.
- For example, a node representing the likelihood of heart disease may depend on various factors such as as, cholesterol levels and blood pressure.

★ ALGORITHM / STEPS :

1] Import the Heart Disease Data set and preprocess it (handle missing values, encode categorical variables).
2] Identify relevant variables and their relationships based on medical knowledge.
3] Create a directed acyclic graph [DAG] representing these relationships.
4] For each node, define the conditional probabilities based on the data or expert knowledge.
5] Use the Bayesian network to perform inference based on patient data.
6] Input the symptoms or test results and compute posterior probabilities for heart disease diagnosis.
7] Interpret the results and provide a diagnosis based on probabilities computed by model.

★ CONCLUSION :

- Hence, in this practical, we constructed a Bayesian network model to diagnose heart disease using medical data from Heart Disease Data set.

NAME: RANSING ASHISH PETER
ROLL NO: 42557
BRANCH: BE AI&DS
BATCH: B3

★ COMPUTER LABORATORY II: INFORMATION RETRIEVAL ★

◎ ASSIGNMENT - 4 ◎

★ **TITLE :** Email Spam Filtering using Text Classification

★ **PROBLEM STATEMENT :**
- Implement email spam filtering using text classification algorithm with appropriate dataset.

★ **OBJECTIVES :**
- To understand the concept of text classification in e-mail spam filtering.
- To implement a text classification algorithm (e.g. Naive Bayes, SVM) for distinguishing between spam and non spam emails.

★ **PRE-REQUISITE :**
- Basic knowledge of text pre-processing techniques
- Familiarity with machine learning algorithm especially text classification methods like Naive Bayes or Support Vector Machines (SVM).

★ **OUTCOME :**
- At the end of this practical, we will be able to implement a text classification algorithm for email spam filtering.

* THEORY:

- Email spam filtering is a common application of text classification, where the objective is to categorize emails as either spam (unsolicited or unwanted emails) or ham (legitimate emails).
- Various machine learning algorithms can be used for this purpose, with text classification algorithms like Naive Bayes, Support Vector Machines [SVM], and Logistic Regression being popular choices.
- Key concepts:

1] Text pre-processing →
- Methods like tokenization, stop word removal and stemming / Lemmatization are important here.

2] Feature Extraction →
- Methods like Bag-of-Words (BoW) model or TF-IDF (Term Frequency-Inverse Document Frequency) are used to convert text into numerical form.

3] Classification Algorithm →
- Naive Bayes or Support Vector Machines (SVMs) are used for effective classification operation.

★ ALGORITHM:

1] Load the dataset containing e-mails labeled as spam or ham.
2] Perform text pre-processing (tokenization, stop word removal, stemming).
3] Convert the pre-processed text into numerical features using the Bag-of-Words or TF-IDF model.
4] Split the data into training and testing sets for model training.
5] Train a Naive Bayes classifier (or any other text classification algorithm) on the training data.
6] Use the trained model to predict whether the emails in the test set are spam or ham.
7] Evaluate the performance of the model using accuracy, precision, recall and F1-score.

★ CONCLUSION:

- Hence, in this practical, we successfully implemented a text classification algorithm to filter emails into spam and non-spam categories using a suitable dataset.

NAME: RANSING ASHISH PETER
ROLL NO: 42557
BRANCH: BE AI&DS
BATCH: B3

★ Computer Laboratory II: Information Retrieval ★

⊙ Assignment - 5 ⊙

★ **Title:** Agglomerative Hierarchical Clustering Algorithm.

★ **Problem Statement:**
- Implement Agglomerative Hierarchical clustering algorithm using appropriate dataset.

★ **Objectives:**
- To understand the concept and steps of Agglomerative Hierarchical Clustering algorithm.
- To implement the Agglomerative HCA using suitable dataset.

★ **Pre-Requisite:**
- Basic knowledge of clustering techniques and unsupervised learning.
- Familiarity with Python programming and data manipulation libraries (e.g. pandas, numpy).

★ **Outcome:**
- At the end of this practical, we will be able to implement the Agglomerative Hierarchical Clustering algorithm.

* THEORY :

- Agglomerative Hierarchical Clustering is an unsupervised learning algorithm used for clustering data points.
- It is a bottom-up approach that starts with each data points as an individual cluster and merges them iteratively based on similarity until all points belong to a single cluster or until a stopping criterion is met.
- Key Concepts :

1] Clustering Process → a] Initialization
b] Distance Calculation
c] Merging
d] Repeat.

2] Distance Metrics → a] Euclidean distance
b] Manhattan distance
c] Cosine similarity.

3] Linkage Criteria → a] Single Linkage
b] Complete Linkage
c] Average Linkage
d] Ward's Method.

★ ALGORITHM / STEPS :

1] Load the dataset and preprocess it. (Handle the missing values, etc.)
2] Compute the pairwise distance between all data points using an appropriate distance metric.
3] Initialize the cluster, start with each data point as its own cluster.
4] Find the two closest clusters and merge them.
5] Update the distance matrix.
6] Continue merging clusters until the desired number of clusters is reached or all points are in one cluster.
7] Create a dendrogram to visualize the clustering process.

★ CONCLUSION :

- Hence, in this practical, we implemented the Agglomerative Hierarchical Clustering algorithm to group similar data points using relevant dataset.

NAME: RANSING ASHISH PETER
ROLL NO: 42557
BRANCH: BE AI&DS
BATCH: B3

★ Computer Laboratory II : Information Retrieval ★

⊙ Assignment - 6 ⊙

★ **Title :** PageRank Algorithm.

★ **Problem Statement :**
- Implement Page Rank Algorithm (Use python or beautiful soup for implementation).

★ **Objectives :**
- To understand the concept of PageRank algorithm and its significance in ranking web pages.
- To implement the PageRank algorithm using a sample directed graph of web pages.

★ **Pre-Requisites :**
- Basic understanding of graph theory and web scraping.
- Familiarity with Python programming.
- Knowledge of directed graphs and their representation.

★ **Outcome :**
- At the end of this practical, we will be able to implement the PageRank algorithm.

* THEORY:

- PageRank is an algorithm used by search engines to rank web pages in their search results. It works on the principle that the importance of a web page can be determined by the quantity and quality of links pointing to it.
- The algorithm is based on following concepts →

1] Graph Representation →

- The web can be represented as a directed graph where web pages are nodes and hyperlinks between them are directed edges.

2] Rank Calculation →

- Each page is assigned a rank.
- The PageRank of page is calculated using the formula:-

$$PR(A) = (1-d) + d \sum_{i=1} \frac{PR(B_i)}{C(B_i)}$$

where,

- PR(A) is PageRank of page A.
- d is a damping factor (0.85)
- Bi are pages linking to A.
- C(Bi) is the number of links on page Bi.

3] Damping Factor →
- It accounts for probability that a user continues clicking on links. [Generally d = 0.85].

4] Convergence →
- The algorithm iteratively updates the PageRank scores until the converge.

★ ALGORITHM / STEPS :
1] Use web scraping to extract links between web pages or define a directed graph.
2] Assign an initial PageRank score to each page (usually 1/N, where N is total number of pages).
3] Update the PageRank scores iteratively using formula until convergence.
4] Display the final PageRank scores and visualize them.

★ CONCLUSION :
- Hence, in this practical, we successfully implemented the PageRank algorithm to compute importance of web pages based on their link structure.