DR. D. Y. Patil Institute of Technology, Pimpri, Pune

Department of Artificial Intelligence and Data Science

LAB MANUAL
Computer Laboratory-II
(BE)
Semester I

Prepared by:
Mrs. Pranjali Bahalkar

Computer Laboratory -II

Course Code	Course Name	Teaching Scheme (Hrs./ Week)	Credits
417526	Computer Laboratory-II: Information Retrieval	4	2

Course Objectives:

- Understand the concepts of information retrieval and web mining
- Understand information retrieval process using standards available tools

Course Outcomes:

- Understand the concepts of information retrieval and web mining
- Understand information retrieval process using standards available tools

Table of Contents

Lab Assignment No.	01
Title	Write a program for pre-processing of a text document such as stop word removal, stemming.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-II
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 01

Title: Write a program for pre-processing of a text document such as stop word removal, stemming.

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

Prerequisite:

Basics of Python

Software Requirements: Jupyter

Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

Learning Objectives:

Learn to remove stop words and stemming

Outcomes:

After completion of this assignment students are able to understand how to remove stop words and stemming

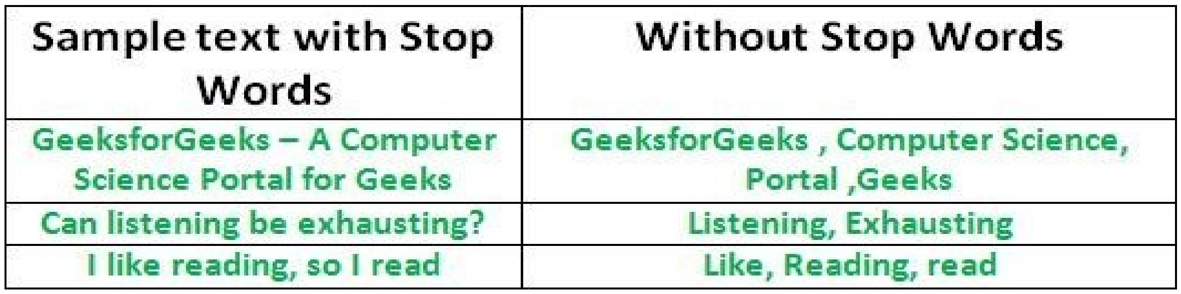
Theory:

The process of converting data to something a computer can understand is referred to as **pre-processing.** One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.

What are Stop words?

Stop Words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages. You can find them in the nltk_data directory. home/pratima/nltk_data/corpora/stopwords are the directory address.(Do not forget to change your home directory name)



Sample text with Stop Words	Without Stop Words
GeeksforGeeks – A Computer Science Portal for Geeks	GeeksforGeeks , Computer Science, Portal ,Geeks
Can listening be exhausting?	Listening, Exhausting
I like reading, so I read	Like, Reading, read

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”. Stemming is an important part of the pipelining process in Natural language processing. The input to the stemmer is tokenized words. How do we get these tokenized words? Well, tokenization involves breaking down the document into different words.

Stemming is a natural language processing technique that is used to reduce words to their base form, also known as the root form. The process of stemming is used to normalize text and make it easier to process. It is an important step in text pre-processing, and it is commonly used in information retrieval and text mining applications.

There are several different algorithms for stemming, including the Porter stemmer, Snowball stemmer, and the Lancaster stemmer. The Porter stemmer is the most widely used algorithm, and it is based on a set of heuristics that are used to remove common suffixes from words. The Snowball stemmer is a more advanced algorithm that is based on the Porter stemmer, but it also supports several other languages in addition to English. The Lancaster stemmer is a more aggressive stemmer and it is less accurate than the Porter stemmer and Snowball stemmer.

Stemming can be useful for several natural language processing tasks such as text classification, information retrieval, and text summarization. However, stemming can also have some negative effects such as reducing the readability of the text, and it may not always produce the correct root form of a word.

Errors in Stemming:

There are mainly two errors in

stemming – over-stemming

under-stemming

Over-stemming occurs when two words are stemmed from the same root that are of different stems. Over-stemming can also be regarded as false-positives. Over-stemming is a problem that can occur when using stemming algorithms in natural language processing. It refers to the situation where a stemmer produces a root form that is not a valid word or is not the correct root form of a word. This can happen when the stemmer is too aggressive in removing suffixes or when it does not consider the context of the word.

Over-stemming can lead to a loss of meaning and make the text less readable. For example, the word "arguing" may be stemmed to "argu," which is not a valid word and does not convey the same meaning as the original word. Similarly, the word "running" may be stemmed to "run," which is the base form of the word but it does not convey the meaning of the original word.

To avoid over-stemming, it is important to use a stemmer that is appropriate for the task and language. It is also important to test the stemmer on a sample of text to ensure that it is producing valid root forms. In some cases, using a lemmatizer instead of a stemmer may be a better solution as it takes into account the context of the word, making it less prone to errors.

Another approach to this problem is to use techniques like semantic role labeling, sentiment analysis, context-based information, etc. that help to understand the context of the text and make the stemming process more precise.

Under-stemming occurs when two words are stemmed from the same root that are not of different stems. Under-stemming can be interpreted as false-negatives. Under-stemming is a problem that can occur when using stemming algorithms in natural language processing. It refers to the situation where a stemmer does not produce the correct root form of a word or does not reduce a word to its base form. This can happen when the stemmer is not aggressive enough in removing suffixes or when it is not designed for the specific task or language.

Under-stemming can lead to a loss of information and make it more difficult to analyze text. For example, the word "arguing" and "argument" may be stemmed to "argu," which does not convey
the meaning of the original words. Similarly, the word "running" and "runner" may be stemmed to "run," which is the base form of the word but it does not convey the meaning of the original words.

To avoid under-stemming, it is important to use a stemmer that is appropriate for the task and language. It is also important to test the stemmer on a sample of text to ensure that it is producing.

The correct root forms. In some cases, using a lemmatizer instead of a stemmer may be a better solution as it takes into account the context of the word, making it less prone to errors.

Another approach to this problem is to use techniques like semantic role labeling, sentiment analysis, context-based information, etc. that help to understand the context of the text and makethe stemming process more precise.

Applications of stemming :

Stemming is used in information retrieval systems like search

engines. It is used to determine domain vocabularies in

domain analysis.

To display search results by indexing while documents are evolving into numbers and to map documents to common subjects by stemming.

Sentiment Analysis, which examines reviews and comments made by different users about anything, is frequently used for product analysis, such as for online retail stores. Before it is interpreted, stemming is accepted in the form of the text-preparation mean.

A method of group analysis used on textual materials is called document clustering (also known as text clustering). Important uses of it include subject extraction, automatic document structuring, andquick information retrieval.

Fun Fact: Google search adopted a word stemming in 2003. Previously a search for “fish” would nothave returned “fishing” or “fishes”.

Some Stemming algorithms are:

Porter’s Stemmer algorithm
It is one of the most popular stemming methods proposed in 1980. It is based on the idea that the suffixes in the English language are made up of a combination of smaller and simpler suffixes. This stemmer is known for its speed and simplicity. The main applications of Porter Stemmer include data mining and Information retrieval. However, its applications are only limited to English words. Also, the group of stems is mapped on to the same stem and the output stem is not necessarily a meaningful word. The algorithms are fairly lengthy in nature and are known to be the oldest stemmer.
Example: EED -> EE means “if the word has at least one vowel and consonant plus EED ending, change the ending to EE” as ‘agreed’ becomes ‘agree’.

Advantage: It produces the best output as compared to other stemmers and it has less error rate.

Limitation: Morphological variants produced are not always real words.

Lovins Stemmer-
It is proposed by Lovins in 1968, that removes the longest suffix from a word then the word is recorded to convert this stem into valid words.
Example: sitting -> sitt -> sit
Advantage: It is fast and handles irregular plurals like 'teeth' and 'tooth' etc

Limitation: It is time consuming and frequently fails to form words from

stem. **Dawson Stemmer**
It is an extension of Lovins stemmer in which suffixes are stored in the reversed order indexed by their length and last letter.

Advantage: It is fast in execution and covers more

suffices. **Limitation:** It is very complex to implement.

Krovetz Stemmer
It was proposed in 1993 by Robert Krovetz. Following are the steps:
1) Convert the plural form of a word to its singular form.
2) Convert the past tense of a word to its present tense and remove the suffix 'ing'. Example: 'children' -> 'child'

Advantage: It is light in nature and can be used as pre-stemmer for other stemmers.

Limitation: It is inefficient in case of large documents.

Xerox
Stemme
r
Example
:

'children' -> 'child'

'understood' ->

'understand' 'whom'

-> 'who'

'best' -> 'good'

N-Gram Stemmer
An n-gram is a set of n consecutive characters extracted from a word in which similar words will have a high proportion of n-grams in common.
Example: 'INTRODUCTIONS' for n=2 becomes : I, IN, NT, TR, RO, OD, DU, UC, CT, TI, IO, ON, NS, S

Advantage: It is based on string comparisons and it is language dependent.

Limitation: It requires space to create and index the n-grams and it is not time

efficient. <u>Snowball Stemmer:</u>

When compared to the Porter Stemmer, the Snowball Stemmer can map non-English words too. Since it supports other languages the Snowball Stemmers can be called a multi-lingual stemmer. The Snowball stemmers are also imported from the nltk package. This stemmer is based on a programming language called ‘Snowball’ that processes small strings and is the most widely used stemmer. The Snowball stemmer is way more aggressive than Porter Stemmer and is also referred to as Porter2 Stemmer. Because of the improvements added when compared to the Porter Stemmer, the Snowball stemmer is having greater computational speed.

Lancaster Stemmer:

The Lancaster stemmers are more aggressive and dynamic compared to the other two stemmers. The stemmer is really faster, but the algorithm is really confusing when dealing with small words. But they are not as efficient as Snowball Stemmers. The Lancaster stemmers save the rules externally and basically uses an iterative algorithm.

Some more example of stemming for root word "like" include:

-> "likes"

-> "liked"

-> "likely"

-> "liking"

Errors in Stemming: There are mainly two errors in stemming – ***Overstemming*** and ***Understemming***. Overstemming occurs when two words are stemmed from the same root that are of different stems. Under-stemming occurs when two words are stemmed from the same root that is not of different stems.

Applications of stemming are:

Stemming is used in information retrieval systems like search

engines. It is used to determine domain vocabularies in domain

analysis.

Stemming is desirable as it may reduce redundancy as most of the time the word stem and their inflected/derived words mean the same.
Conclusion: The pre-processing of text data not only reduces the dataset size but also helps us to focus on only useful and relevant data so that the future model would have a large percentage of efficiency. With the help of pre-processing techniques like tokenization, stemming, lemmatization, removing stop-words, and part of speech tag we can remove all the irrelevant text from our dataset and make our dataset ready for further processing or model building.

Lab Assignment No.	02
Title	Implement a program for retrieval of documents using inverted files.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-II
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 02

Title: Implement a program for retrieval of documents using inverted files.

Problem Statement: Implement a program for retrieval of documents using

inverted files. **Prerequisite:**

Basics of Python

Software Requirements: Jupyter

Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

Learning Objectives:

Learn to retrieve documents using inverted files

Outcomes:

After completion of this assignment students are able to understand how to retrieve documents using inverted files

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 number s, to its locations in a document or a set of documents. In simple words, it is a hashmap-like data structure that directs you from a word to a document or a web page.
Example: Consider the following documents.
Document 1: The quick brown fox jumped over the lazy
dog. Document 2: The lazy dog slept in the sun.
To create an **inverted index** for these documents, we first tokenize the documents into terms, as follows.
Document 1: The, quick, brown, fox, jumped, over, the
lazy, dog. Document 2: The, lazy, dog, slept, in, the, sun.
Next, we create an index of the terms, where each term points to a list of documents that contain that term, as follows.

The -> Document 1,
Document 2 Quick ->
Document 1
Brown ->
Document 1
Fox ->
Document 1
Jumped ->
Document 1
Over ->
Document 1
Lazy -> Document 1, Document 2
Dog -> Document 1,
Document 2 Slept ->
Document 2
In ->
Document 2
Sun ->
Document 2

To search for documents containing a particular term or set of terms, the search engine queries the inverted index for those terms and retrieves the list of documents associated with each term. The search engine can then use this information to rank the documents based on relevance to the query and present them to the user in order of importance.
There are two types of inverted indexes:

- **Record-Level Inverted Index:** Record Level Inverted Index contains a list of references to documents for each word.
- **Word-Level Inverted Index:** Word Level Inverted Index additionally contains the positions of each word within a document. The latter form offers more functionality but needs more processing power and space to be created.

Suppose we want to search the texts "hello everyone, " "this article is based on an inverted index, " and "which is **hashmap-like data structure**". If we index by (text, word within the text), the index with a location in the text is:

hello	(1, 1)
everyone	(1, 2)
this	(2, 1)
article	(2, 2)
is	(2, 3); (3, 2)
based	(2, 4)
on	(2, 5)
inverted	(2, 6)
index	(2, 7)
which	(3, 1)
hashmap	(3, 3)
like	(3, 4)
data	(3, 5)
structure	(3, 6)

The word “hel o” is in document 1 (“hello everyone”) starting at word 1, so has an entry (1, 1), and the word “is” is in documents 2 and 3 at ‘3rd’ and ‘2nd’ positions respectively (here position is based on the word).
The index may have weights, frequencies, or other indicators.

Steps to Build an Inverted Index

- **Fetch the Document:** Removing of Stop Words: Stop words are the most occurring and useless words in documents like “I”, “the”, “we”, “is”, and “an”.

- **Stemming of Root Word:** Whenever I want to search for "cat", I want to see a document that has information about it. But the word present in the document is called "cats" or "catty" instead of "cat". To relate both words, I'**l** chop some part of every word I read so that I could get the "root word". There are standard tools for performing this like "Porter's Stemmer".
- **Record Document IDs:** If the word is already present add a reference of the document to index else creates a new entry. Add additional information like the frequency of the word, location of the word, etc.

Example:

Words

Document ant	doc1
demo	doc2
world	doc1, doc2

Implementing Inverted Index

```
  Define the documents
document1 = "The quick brown fox jumped over the lazy dog."
document2 = "The  lazy dog slept in the sun."

  Step 1: Tokenize  the documents
  Convert each document to lowercase and split it into wordstokens1
= document1.lower().split()
tokens2 =
document2.lower().split()

  Combine the tokens into a list of unique
terms terms = list(set(tokens1  +
tokens2))

  Step 2: Build the inverted index
  Create an empty dictionary  to store the inverted
index inverted_index = {}

  For each term, find the documents that contain
itfor term in terms:
    documents  = []
    if term in tokens1:
        documents.append("Docum
        ent 1")
    if term in tokens2:
        documents.append("Docum
        ent 2")
    inverted_index[term]  = documents

  Step 3: Print the inverted index
for term, documents in
    inverted_index.items(): print(term,  "->", ",
    ".join(documents))
```

Advantages of Inverted Index

- The inverted index is to allow fast full-text searches, at a cost of increased processing when a document is added to the database.
- It is easy to develop.
- It is the most popular data structure used in document retrieval systems, used on a large scale for example in search engines.

Disadvantages of Inverted Index

- Large storage overhead and high maintenance costs on updating, deleting, and inserting.
- Instead of retrieving the data in decreasing order of expected usefulness, the records are retrieved in the order in which they occur in the inverted lists.

Features of Inverted Indexes

- **Efficient search:** Inverted indexes allow for efficient searching of large volumes of text-based data. By indexing every term in every document, the index can quickly identify all documents that contain a given search term or phrase, significantly reducing search time.
- **Fast updates:** Inverted indexes can be updated quickly and efficiently as new content is added to the system. This allows for near-real-time indexing and searching for new content.
- **Flexibility:** Inverted indexes can be customized to suit the needs of different types of information retrieval systems. For example, they can be configured to handle different types of queries, such as Boolean queries or proximity queries.
- **Compression:** Inverted indexes can be compressed to reduce storage requirements. Various techniques such as delta encoding, gamma encoding, variable byte encoding, etc. can be used to compress the posting list efficiently.
- **Support for stemming and synonym expansion:** Inverted indexes can be configured to support stemming and synonym expansion, which can improve the accuracy and relevance of search results. Stemming is the process of reducing words to their base or root form, while synonym expansion involves mapping different words that have similar meanings to a common term.
- **Support for multiple languages:** Inverted indexes can support multiple languages, allowing users to search for content in different languages using the same system.

Conclusion: - This way, Implemented a program for inverted files.

Lab Assignment No.	3
Title	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 (You can use Java/Python ML library classes/API.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-II
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 03

Title: 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 (You can use Java/Python ML library classes/API.

Problem Statement: 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 (You can use Java/Python ML library classes/API.

Prerequisite:

Basics of Python

Software Requirements: Jupyter

Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

Learning Objectives:

Learn to construct a Bayesian network considering medical data.

Outcomes:

After completion of this assignment students are able to understand how to construct a Bayesian network considering medical data.

Theory:

A Bayesian network is a directed acyclic graph in which each edge corresponds to a conditiona l dependency, and each node corresponds to a unique random variable.

Bayesian network consists of two major parts: a directed acyclic graph and a set of conditional probability distributions

- The directed acyclic graph is a set of random variables represented by nodes.
- The conditional probability distribution of a node (random variable) is defined for everypossible outcome of the preceding causal node(s).

For illustration, consider the following example. Suppose we attempt to turn on our computer,but the computer does not start (observation/evidence). We would like to know

which of the possible causes of computer failure is more likely. In this simplified illustration, we assume only two possible causes of this misfortune: electricity failure and computer malfunction.
The corresponding directed acyclic graph is depicted in below figure.

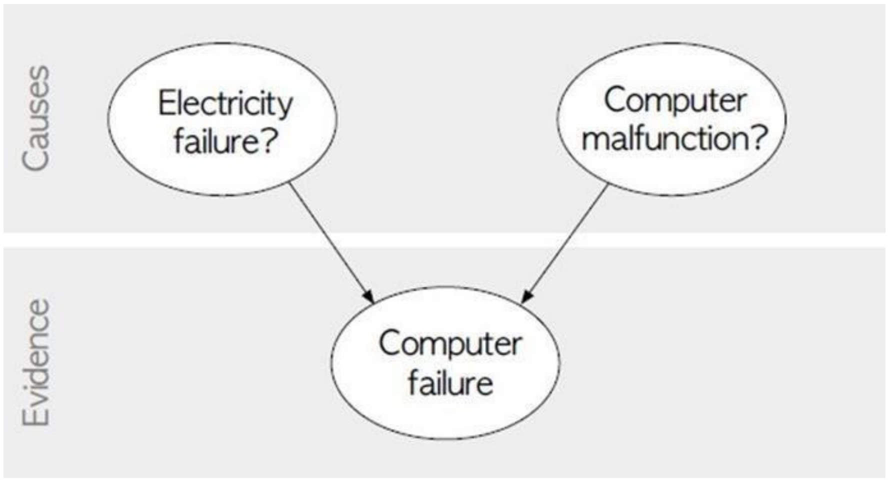


Fig: Directed acyclic graph representing two independent possible causes of a computer failure.

The goal is to calculate the posterior conditional probability distribution of each of the possible unobserved causes given the observed evidence, i.e. P [Cause | Evidence].

Data Set:

Title: Heart Disease Databases

The Cleveland database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "Heartdisease" field refers to the presence of heart diseasein the patient. It is integer valued from 0 (no presence) to 4.

Database:	0	1	2	3	4	Total
Cleveland:	164	55	36	35	13	303

Attribute Information:

1. age: age in years

2. sex: sex (1 = male; 0 = female)
3. cp: chest pain type
    - Value 1: typical angina
    - Value 2: atypical angina
    - Value 3: non-anginal pain
    - Value 4: asymptomatic
4. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
5. chol: serum cholestoral in mg/dl
6. fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg: resting electrocardiographic results
    - Value 0: normal
    - Value 1: having ST-T wave abnormality (T wave inversions and/ or ST elevationor depression of > 0.05 mV)
    - Value 2: showing probable or definite left ventricular hypertrophy by Estes'criteria
8. thalach: maximum heart rate achieved
9. exang: exercise induced angina (1 = yes; 0 = no)
10. oldpeak = ST depression induced by exercise relative to rest
11. slope: the slope of the peak exercise ST segment
    - Value 1: upsloping
    - Value 2: flat
    - Value 3: downsloping
12. ca = number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
14. Heartdisease: It is integer valued from 0 (no presence) to 4. Diagnosis of heart disease(angiographic disease status)

Some instance from the dataset:

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	Heartdisease
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
60	1	4	130	206	0	2	132	1	2.4	2	2	7	4

```
     given below-

     age  sex cp  trestbps     ...slope ca
thal heartdisease
0    63     1    1      145       ...  3    0      6              0
1    67     1    4      160       ...  2    3      3              2
2    67     1    4      120       ...  2    2      7              1
3    37     1    3      130       ...  3    0      3              0
4    41     0    2      130       ...  1    0      3              0

[5 rows x 14 columns]

Learning CPD using Maximum likelihood

estimatorsInferencing with Bayesian Network:


1. Probability of HeartDisease given Age=28
╒════════════════╤═════════════════════╕
│ heartdisease   │   phi(heartdisease) │
╞════════════════╪═════════════════════╡
│ heartdisease_0 │              0.6791 │
├────────────────┼─────────────────────┤
│ heartdisease_1 │              0.1212 │
├────────────────┼─────────────────────┤
│ heartdisease_2 │              0.0810 │
├────────────────┼─────────────────────┤
│ heartdisease_3 │              0.0939 │
├────────────────┼─────────────────────┤
│ heartdisease_4 │              0.0247 │
╘════════════════╧═════════════════════╛
```

```
2. Probability of HeartDisease given cholesterol=100
╒════════════════╤═════════════════════╕
│ heartdisease   │   phi(heartdisease) │
╞════════════════╪═════════════════════╡
│ heartdisease_0 │              0.5400 │
├────────────────┼─────────────────────┤
│ heartdisease_1 │              0.1533 │
├────────────────┼─────────────────────┤
│ heartdisease_2 │              0.1303 │
├────────────────┼─────────────────────┤
│ heartdisease_3 │              0.1259 │
├────────────────┼─────────────────────┤
│ heartdisease_4 │              0.0506 │
╘════════════════╧═════════════════════╛
```

Conclusion: This way constructed a Bayesian network considering medical data.

Lab Assignment No.	04
Title	Implement Agglomerative hierarchical clustering algorithm using appropriate dataset.
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-II
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 04

Title: I+++.

Problem Statement: Implement Agglomerative hierarchical clustering algorithm using appropriate dataset

Prerequisite:

Basics of Python

Software Requirements: Jupyter

Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

Learning Objectives:

Learn to Implement Agglomerative hierarchical clustering algorithm using appropriate dataset

Outcomes:

After completion of this assignment students are able to understand how to Implement Agglomerative hierarchical clustering algorithm using appropriate dataset

Theory:

In data mining and statistics, hierarchical clustering analysis is a method of clustering analysis that seeks to build a hierarchy of clusters i.e. tree-type structure based on the hierarchy.

In machine learning, clustering is the unsupervised learning technique that groups the data based on similarity between the set of data. There are different-different types of clustering algorithms in machine learning. **Connectivity-based clustering:** This type of clustering algorithm builds the cluster based on the connectivity between the data points. Example: Hierarchical clustering

- **Centroid-based clustering:** This type of clustering algorithm forms around the centroids of the data points. Example: K-Means clustering, K-Mode clustering
- **Distribution-based clustering:** This type of clustering algorithm is modeled using statistical distributions. It assumes that the data points in a cluster are generated from a particular probability distribution, and the algorithm aims to estimate the parameters of the distribution to group similar data points into clusters Example: Gaussian Mixture Models (GMM)

- **Density-based clustering:** This type of clustering algorithm groups together data points that are in high-density concentrations and separates points in low-concentrations regions. The basic idea is that it identifies regions in the data space that have a high density of data points and groups those points together into clusters. Example: DBSCAN(Density-Based Spatial Clustering of Applications with Noise)

Hierarchical clustering

Hierarchical clustering is a connectivity-based clustering model that groups the data points together that are close to each other based on the measure of similarity or distance. The assumption is that data points that are close to each other are more similar or related than data points that are farther apart.

A dendrogram, a tree-like figure produced by hierarchical clustering, depicts the hierarchical relationships between groups. Individual data points are located at the bottom of the dendrogram, while the largest clusters, which include all the data points, are located at the top. In order to generate different numbers of clusters, the dendrogram can be sliced at various heights.

The dendrogram is created by iteratively merging or splitting clusters based on a measure of similarity or distance between data points. Clusters are divided or merged repeatedly until all data points are contained within a single cluster, or until the predetermined number of clusters is attained.

We can look at the dendrogram and measure the height at which the branches of the dendrogram form distinct clusters to calculate the ideal number of clusters. The dendrogram can be sliced at this height to determine the number of clusters.

Types of Hierarchical Clustering

Basically, there are two types of hierarchical Clustering:

1. Agglomerative Clustering
2. Divisive clustering

Hierarchical Agglomerative Clustering

It is also known as the bottom-up approach or hierarchical agglomerative clustering (HAC). A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to prespecify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.

Agglomerative Clustering is one of the most common hierarchical clustering techniques. Dataset – Credit Card Dataset.

Assumption: The clustering technique assumes that each data point is similar enough to the other data points that the data at the starting can be assumed to be clustered in 1 cluster. Step 1: Importing the required libraries

Conclusion :- This way Implemented Agglomerative hierarchical clustering algorithm using appropriate dataset

Lab Assignment No.	5
Title	Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-II
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 5

Title: Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).

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

Prerequisite:

Basics of Python

Software Requirements: Jupyter

Hardware Requirements:

PIV, 2GB RAM, 500 GB HDD

Learning Objectives:

Learn to Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).

Outcomes:

After completion of this assignment students are able to understand how to Implement Page Rank Algorithm. (Use python or beautiful soup for implementation).

Theory:

PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google:

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

It is not the only algorithm used by Google to order search engine results, but it is the first algorithm that was used by the company, and it is the best-known. The above centrality measure is not implemented for multi-graphs.

Algorithm

The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called "iterations", through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

Simplified algorithm

Assume a small universe of four web pages: A, B, C, and D. Links from a page to itself, or multiple outbound links from one single page to another single page, are ignored. PageRank is initialized to the same value for all pages. In the original form of PageRank, the sum of PageRank over all pages was the total number of pages on the web at that time, so each page in this example would have an initial value of 1. However, later versions of PageRank, and the remainder of this section, assume a probability distribution between 0 and 1. Hence the initial value for each page in this example is 0.25.

The PageRank transferred from a given page to the targets of its outbound links upon the next iteration is divided equally among all outbound links.

If the only links in the system were from pages B, C, and D to A, each link would transfer 0.25 PageRank to A upon the next iteration, for a total of 0.75.

Suppose instead that page B had a link to pages C and A, page C had a link to page A, and page D had links to all three pages. Thus, upon the first iteration, page B would transfer half of its existing value, or 0.125, to page A and the other half, or 0.125, to page C. Page C would transfer all of its existing value, 0.25, to the only page it links to, A. Since D had three outbound links, it would transfer one-third of its existing value, or approximately 0.083, to A. At the completion of this iteration, page A will have a PageRank of approximately 0.458.

In other words, the PageRank conferred by an outbound link is equal to the document's own PageRank score divided by the numberof outbound links L( ).

In the general case, the PageRank value for any page u can be expressed as:

i.e. the PageRank value for a page u is dependent on the PageRank values for each page v contained in the set Bu (the set containing all pages linking to page u), divided by the number L(v) of links from page v. The algorithm involves a damping factor for the calculation of the PageRank. It is like the income tax which the govt extracts fromone despite paying him itself.

Conclusion:- Thus, this way the centrality measure of Page Rank is calculated for the given graph.