**Clustering News Headlines**

Extract/read data using BeautifulSoup

Clean, tokenize and stem data using NLTK

Convert words to vector space using TFIDF matrix

Calculate Cosine Similarity and generate the distance matrix

Generate Clusters using different clustering algorithms

KMeans clustering

Hierarchical clustering using Ward method

Dimensionality reduction using MDS

Visualization of clusters using Matplotlib

The various techniques are explained below:-

**BeautifulSoup**

We may sometimes need to gather data from different websites for analytic purposes that we are working on. One of the techniques to accomplish this is scraping websites using Beautiful Soup. Web scraping automatically extracts data and presents it in a format that can easily make sense. Beautiful Soup is a python package that helps us parse html pages and extract the desired content. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping. Beautiful Soup relies on parsers like lxml, html.

**Tokenization**

Tokenization is a process of breaking the sequence of strings into pieces of words, phrases and other elements. These pieces here are referred as tokens and are input to other process like text mining, stemming etc.

**Stemming**

Stemming is a process of removing and replacing word suffixes to arrive at a common root form of the word. For example, responsive, responsitivity, responsiveness - all refer to a common stem ‘respons’. The suffixes are removed from these words to arrive at their stem.

**NLTK - Natural Language ToolKit**

Natural Language ToolKit, commonly known as NLTK is python package for working with natural language processing. It helps us in text mining processes and techniques. NLTK has been used to tokenize, remove stop words and stem words in our text data.

**TFIDF - Term Frequency Inverse document Frequency**

The **tf-idf** is a weighing scheme that is intended to measure how important a word is to a document in context to a collection (or corpus) of documents, For example, a word to one website in a collection of websites or a term to one news headline in a collection of news headlines.

**Term Frequency** is a weight that represents how important a word is in the document. It counts the number of occurrence of each word in a given document.

There are different measures for Term frequency represented as

tf.png

It is similar to 'bag of words' model where the exact ordering of the words in the document is ignored. Eg - 'Chetan runs faster than Manu' is identical to ' Manu runs faster than Chetan'.

The **inverse document frequency** is a measure of how much information the word provides, that is, whether the word is common or rare across all documents. The inverse document frequency diminishes the weight of words that occurs very frequently in the corpus of documents and instead increases the weight of words that occur rarely. Thus, the idf of rare words is high.

Similar to the TF, there are different weighing schemes of inverse document frequency. It is represented as

IDF.png

Together TF-IDF weighing scheme assigns a composite weight to the word in each document.

TFIDF.png

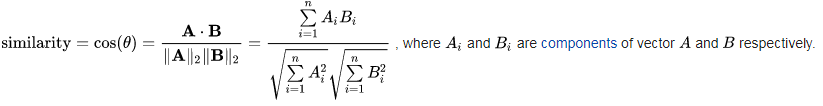
The TFIDF weight assigned is such that

* is highest when the word occurs many times within a small number of documents.
* lower when the word occurs fewer times in the document, or occurs in many documents.
* lowest when the word occurs in virtually all documents.

Thus using TFIDF, we have converted our word documents to a vector space and we can now apply different metric(s) to find the out similarity or categorization of different documents in a collection (corpus) of documents.

**Cosine Similarity**

We can calculate the similarity between pairs of the documents using ‘cosine similarity’ algorithm. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The documents are represented as vectors whose cosine of the angle is determined on a set of the TF-IDF values in the vector space. Cosine Similarity generates a metric that says how related are two documents by looking at the angle instead of the magnitude. The arrows pointing towards to Document A and Document B are the ‘vectors’. If the two vectors are pointing in a similar direction then we can say these two documents are similar.



Cosine similarity is particularly used in positive space, where the outcome is in range between 0 and 1.

CS represents word documents in high dimensional space instead of two dimensional space. By using all the dimensions (or terms in this case) and their corresponding values (or TF-IDF values in this case) each document’s position is determined in the high dimensional space.

**Comparison to other metric functions**

There is one other metric function to find the similarity or categorization of data such as Euclidean distance. Euclidean distance measures the ordinary distance between two points in the space. It calculates the square root of the sum of squares of the difference b/w corresponding points.

Euclidean distance is similar to using a ruler to actually measure the distance which in case of word documents / text mining may not be much relevant. While cosine looks at the angle between two vectors (thus not taking into regard their weight or magnitude).

Cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter. This happens for example when working with text data represented by word counts.

Like for example, a word community occurs more in one text document as it is longer in length than the other text document which is shorter in length. In this case, the weight of the word community might be larger for first document than the second but they appear to be similar documents. In such cases, cosine similarity would be a better metrics.

Clustering is way to partition or group data into meaningful classes based on some measures.

**K-Means Clustering**

K-Means Clustering is an unsupervised learning algorithm which creates k clusters or groups of unlabeled data based on the similarity of data to each other within the same group than the other. Here K is the number of classes in which we want our data to be grouped.

The algorithm works iteratively to assign each data point to one of the K groups based on the features that are provided. K-Means minimizes the variance within the cluster and thus Euclidean distance is used as measurement.

The algorithms starts with initial estimates for the Κ centroids, which can either be randomly generated or randomly selected from the data set.

* Each centroid defines one of the clusters each data point is assigned to its nearest centroid, based on the squared Euclidean distance.
* The centroids of the clusters are recomputed by finding the mean of all the data points within that cluster. The mean becomes the new centroid in the cluster.
* The above two steps are repeated till none of the cluster assignment changes.

K means may fall in local minima, that’s why it can be useful to restart it several times.

For actual clustering tasks, i.e. when you want to analyze the resulting clusters manually, people usually use more advanced methods than k-means. K-means is more of a data simplification technique.

**Choosing K**

Some of the methods to an optimal k are:-

* Elbow method
* Silhouette Analysis
* Using Dendogram
* V-measure
* Adjusted Rand Index

**Dimensionality Reduction**

Using Cosine similarity, the word documents are represented in a high dimensional space. A simple approach to visualize multi-dimensional data is to select two or three dimensions and plot the data seen in that plane. For this, we use the technique known as MDS.

Multidimensional scaling (MDS) is a means of visualizing the level of similarity of individual cases of a dataset.

**Hierarchical Clustering**

**Visualization**