Text Mining, Artificial Neural Networks & Clustering

What is clustering?
Clustering is the task of grouping a set of objects such that objects in the same group are more similar to each other than objects in other groups.

2 main types of clustering discussed in the tutorial are:-

K-means Clustering: - It is an unsupervised algorithm that classifies a given data set into a number of clusters, defined by the letter "k," which is fixed in the first place. The clusters are then positioned as points and all observations or data points associated with the nearest cluster are computed, adjusted and then the process starts to use the new adjustments again and again until the desired result is reached.

Hierarchical Clustering: - It is an algorithm that groups similar objects into groups called 'clusters'. The endpoint is a set of clusters, where each cluster is different from each other cluster, and the objects within each cluster are widely similar to each other.

3 algorithms of hierarchical clustering provided by Python scipy library are as follows:-

1. Single Link(Min)
2. Complete Link(Max)
3. Group Average

What is text mining?
Widely used in knowledge-driven organizations, text mining is the process of examining large collections of documents to discover new information or help answer specific research questions.

Text mining identifies facts, relationships, and assertions that would otherwise remain buried in the mass of textual big data. Once extracted, this information is converted into a structured form that can be further analyzed, or presented directly using clustered HTML tables, mind maps, charts, etc.

What is Countvectorizer?

One-hot encoding (or count vectorization). The idea is very simple. We will be creating vectors that have a dimensionality equal to the size of our vocabulary, and if the text data features that vocab word, we will put a one in that dimension. Every time we encounter that word again, we will increase the count, leaving 0s everywhere we did not find the word even once. The result of this will be very large vectors, if we use them on real text data, however, we will get very accurate counts of the word content of our text data.

Example is as follows-:

```
Vocabulary:

{'one': 12, 'of': 11, 'the': 15, 'most': 9, 'basic': 1, 'ways': 18, 'we': 19,

 'can': 3, 'numerically': 10, 'represent': 13, 'words': 20, 'is': 7,

 'through': 16, 'hot': 6, 'encoding': 5, 'method': 8, 'also': 0,

 'sometimes': 14, 'called': 2, 'count': 4, 'vectorizing': 17}

Full vector:

[[1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1]]

Hot vector:

[[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]

Hot and one:

[[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]]
```

What is TFIDF?

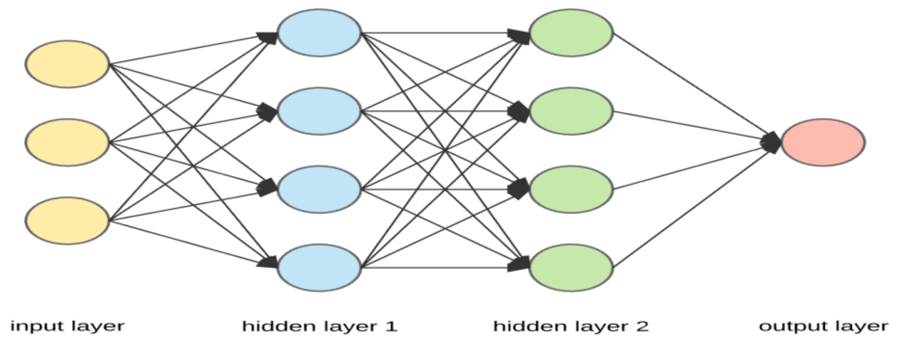
TF-IDF stands for “Term Frequency — Inverse Document Frequency”. This is a technique to quantify a word in documents, we generally compute a weight to each word which signifies the importance of the word in the document and corpus. This method is a widely used technique in Information Retrieval and Text Mining. By vectorizing the documents we can further perform multiple tasks such as finding the relevant documents, ranking, clustering, and so on.

TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF)

The TfidfVectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents. Alternately, if you already have a learned CountVectorizer, you can use it with a TfidfTransformer to just calculate the inverse document frequencies and start encoding documents.

What is the Artificial Neural Network?

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets that look like the figure below. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer.



A given node takes the weighted sum of its inputs, and passes it through a non-linear activation function. This is the output of the node, which then becomes the input of another node in the next layer. The signal flows from left to right, and the final output is calculated by performing this procedure for all the nodes. Training this deep neural network means learning the weights associated with all the edges.

So far we have described the *forward pass*, meaning given an input and weights how the output is computed. After the training is complete, we only run the forward pass to make the predictions. But we first need to train our model to actually learn the weights, and the training procedure works as follows:

- Randomly initialize the weights for all the nodes. There are smart initialization methods which we will explore in another article.
- For every training example, perform a forward pass using the current weights, and calculate the output of each node going from left to right. The final output is the value of the last node.
- Compare the final output with the actual target in the training data, and measure the error using a *loss function.*
- Perform a *backward pass* from right to left and propagate the error to every individual node using *backpropagation*. Calculate each weight's contribution to the error, and adjust the weights accordingly using *gradient descent*. Propagate the error gradients back starting from the last layer.

Observations:-

1] In Text Mining, we have used Hash Vectorizer apart from TFIDF Vectorizer.The main difference is that HashingVectorizer applies a hashing function to term frequency counts in each document, where TFIDF Vectorizer scales those term frequency counts in each document by penalising terms that appear more widely across the corpus. Hash functions are an efficient way of mapping terms to features; it doesn't necessarily need to be applied only to term frequencies.

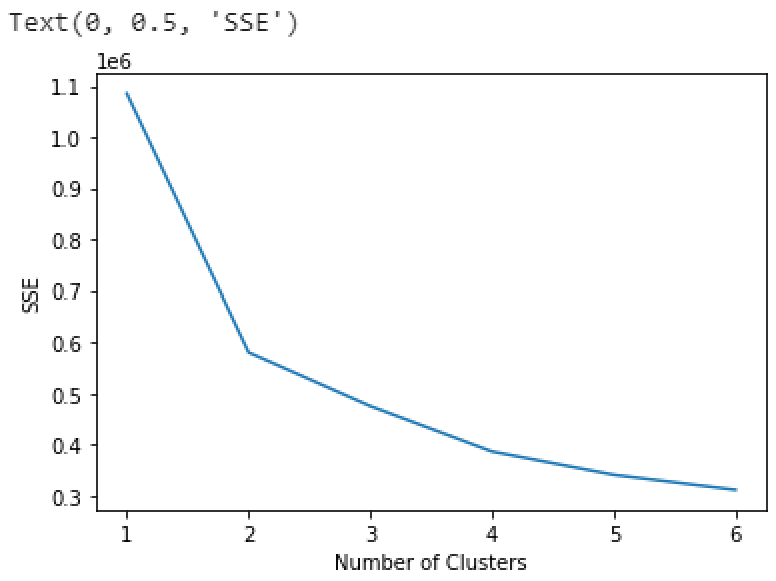
Secondly, for ANN models we have used **'Admission_Predict.csv'** which gives out a good accuracy measure of 0.9 or nearly 90% which is good compared to other classification models such as KNN, SVM. But still KNN and SVM works out to be better.

It's best to match the algorithm to the problem, so you can simply test to see which algorithm works better. But to start with, I would suggest SVM: it works better than KNN with small train sets, and generally easier to train then ANN, as there are less choices to make.

2] In Clustering, we have used dataset named '**IMDB.csv**'. It describes a list of several movies with their imdb ratings. There were many columns like *title, title_type, genre, runtime, mpaa_rating, studio, thtr_rel_year, dvd_rel_year, imdb_rating, critics_score, best_pic_win, best_actor_win, director, actor1, imdb_url* and so on. Several columns which had string values and were not necessary to predict accurate clustering so they were dropped using the *drop()* function.

K-means clustering:

So the final columns were *runtime, thtr_rel_month , thtr_rel_day*, *dvd_rel_month*, *imdb_rating*, *critics_score and audience_score.* We applied the K-means clustering algorithm on the dataset and got Cluster *ID's* in the form *1* and *0* corresponding to every *Movie_name*. We decided the number of cluster values to be *6*.

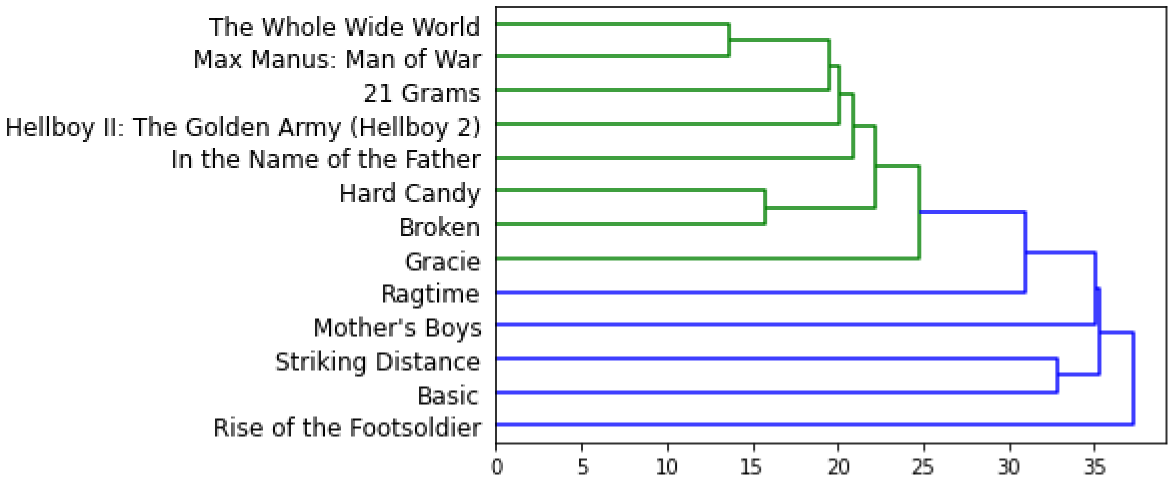


```
Text(0, 0.5, 'SSE')
```

As shown in the above figure, for the cluster value 6, the Sum of Squared Error is 1.1 which is highest.

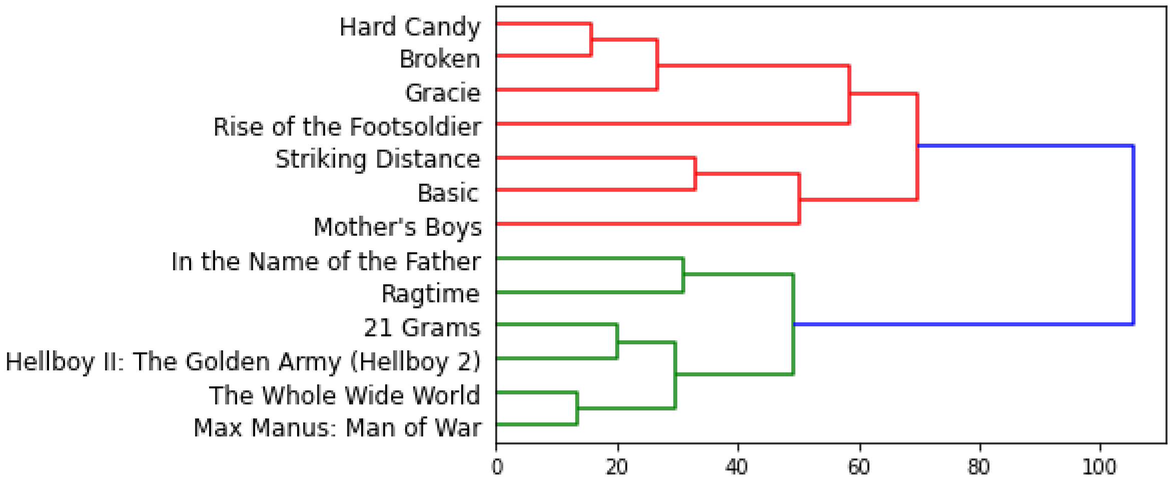
Hierarchical Clustering: In hierarchical clustering, we applied the following algorithms

Single Link (Min):
We considered 2 columns *Movie_name* and *genre*. The movie names were classified based on the genre.



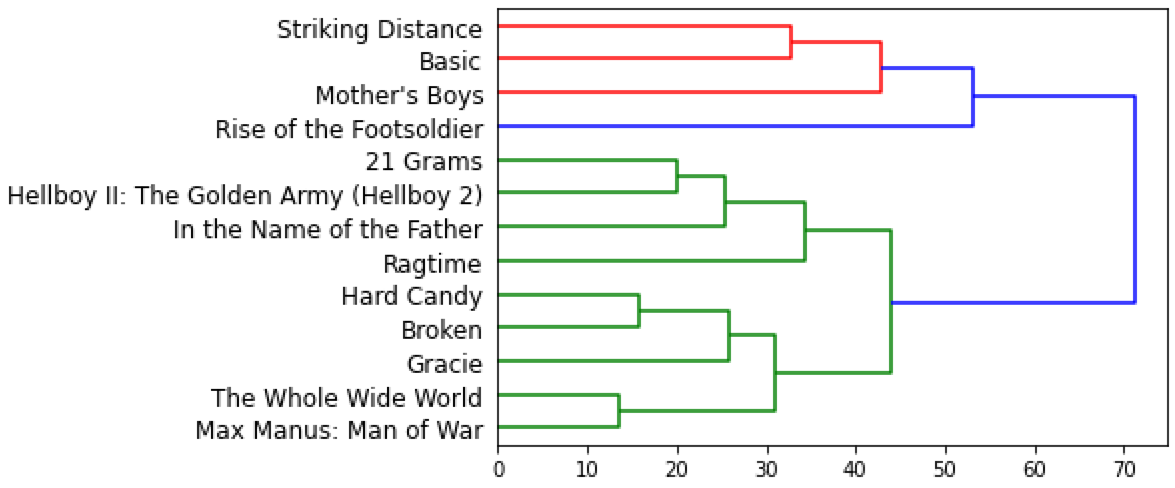
As shown in the above figure, we get the dendogram in which the hierarchy is arranged according to Single Link algorithm.

Complete Link (Max):



As shown in the above figure, we get the dendogram in which the hierarchy is arranged according to Complete Link algorithm.

Group Average:



As shown in the above figure, we get the dendogram in which the hierarchy is arranged according to Group Average.

By Suraj Rawat and Khushali Upadhyay

Code + Documentation (50 %)

Code + Documentation (50 %)