Clustering of Online Encyclopedia

Sakthi Kripa S, Akshaya Mahesh
USML Project Report – Fall 2022

Abstract

In today's digital word text data plays an important role in organizations. Documents are created for each process in the organization and referred to at multiple times. Over the period, as the documents increase in number categorizing and retrieval becomes tedious. By using clustering techniques manual intervention can be avoided. The algorithm can go through the document, look for prominent words using Natural Language Processing techniques and cluster them into the respective categories. The project explores on the performance of clustering algorithms for text data and how tuning the parameters can improve the performance of the algorithms. The project uses articles from Wikipedia to test and validate our algorithms and discusses the results in detail.

Introduction

The project focuses on clustering the articles retrieved from Wikipedia based on their content. The goal of this project is to cluster similar topics together using clustering algorithms. The process carried out is as follows, obtain data by selecting a few topics which are related, retrieve documents from Wikipedia, clean and preprocess the data, apply clustering algorithms, tune their parameters to improve the performance, compare the performance of the algorithms and select the best suited algorithm for the data. Silhouette score is used to compare different models and metrics like homogeneity, completeness, adjusted rand index and v-measure are also used to understand the models in detail.

The project intends to use the following unsupervised machine learning algorithms that help cluster the unlabeled data into different categories:

- Agglomerative
- K-means
- DBSCAN
- Spectral

These algorithms are widely used in fields such as health care, finance, etc. More details on these algorithms will be discussed in the following topics.

Background

The data for this project was acquired using Python's Wikipedia module. This API allows to scrap data from the Wikipedia pages (Online Encyclopedia). The wikipedia.search() function can be used to search a query supplied. In this case, the query will be the category of interest, for example: social media. The output will be topics related to social media like WeChat, Facebook, Instagram, Snapchat, etc. This project includes 8 different categories and about 3-4 topics for each of the categories. The wikipedia.page('Title') is used to retrieve the content or articles for each topic selected which is preprocessed as input for the algorithms.

The NLTK library in python is used to clean the data and remove noise. This module helps in removing stop words, html/url patterns, tokenization, stemming, lemmatization, etc. from the data.

To implement the clustering algorithms sklearn.cluster module from python is used. The algorithms implemented using this module can take different kind of matrices as inputs. This module is very helpful as it also provides various tuning parameters for each algorithm which may impact their performance.

The sklearn.metrics library is used to calculate the performance metrics like v-measure, homogeneity, silhouette score, adjusted rand index and completeness. They take in two arguments, the predicted label, and the actual label to calculate the metrics.

Related Work

This project emphasis on performance of clustering algorithms on text data. Even though many experiments have been carried out in this setting, this project serves as a tool to understand the algorithms in detail. Since the data is not huge it helps visualize how even small change in the parameters changes how the clusters are formed. Furthermore, trying different algorithms on the same data reveals more on the working of these algorithms. With this understanding it becomes easier to expand to larger datasets and real-world applications.

Project Description

The following sections will discuss in detail how the data was acquired for this project, the steps taken to clean the data and make the data prepared for the clustering process, different algorithms which were carried out, their tuning parameters and the performance metrics observed.

Data Acquisition

As mentioned earlier, python's Wikipedia module was used to scrape the data from the Wikipedia pages. 8 categories completely unrelated were chosen namely, data science, finance, sports, religion, global warming, superheroes, social media, and festivals. Under these categories 3-4 topics were chosen in random and articles related to the topic were scrapped using the wikipedia.page('Title'). For example, the category festival has topics such as Diwali, Christmas, Pongal and Thanksgiving and articles related to this topic will be scrapped using the function above. The content retrieved from these articles ranges from 10-50 sentences.

The original label is created by assigning numbers to each category for example, topics related to data science are labelled 0, topics that are related to finance are labelled 1 and so on. These original labels and their content are stored in a dataframe parallel to each other which is used for calculating the performance metrics.

Data Cleaning

After acquiring the data, it must be cleaned to reduce the noise in the data. This is an important step as clustering algorithms are very sensitive to noise and outliers. When it comes to cleaning of text data, it involves natural language processing techniques. This project uses the NLTK library provided by python to clean the data.

First, the data is converted into lower case letter after which URL and html patterns if any are removed. The data is then tokenized and scanned for stop words. Post which the tokenized words go through stemming and lemmatizing.

Stemming is a natural language processing technique which reduces the word to its root form. This is important as the same word may be modified to suit many grammatical circumstances, including tense, case, gender, etc. For example, the words played, plays, player are all reduced to play.

Similarly, lemmatization also reduces the word to its root but instead of just cutting of the end characters it has knowledge on the actual meaning of the word in the language it belongs to. For example, using lemmatization the word better, best, and good will be converted to good as it knows that better and best were derived from the word good, this would not have been the case with stemming. These techniques normalize the text by removing repetition and transforming words to their base and aid in building a robust model.

Post applying these cleaning techniques, the words are joined back together as a sentence.

Data Preprocessing

The data even though is cleaned must be preprocessed and converted into a form which can be given as input to the clustering algorithms. The clustering algorithms in sklearn's module can take different types of matrices as input. Word embedding methods are available to vectorize text data. Bag of Words methods takes into consideration the frequency of the words in a sentence. But it does not take into consideration the importance of the word in a document how each word is related. There might be many zeros resulting in a sparse matrix.

Term Frequency-Inverse Document Frequency method reflects on how important a word is in a document. Since this information might be useful for the models this project uses this method to vectorize the data. Tf-idf is the product of term frequency and inverse document frequency. Term-frequency is the number of times a word appears in the document whereas inverse document frequency is a measure of how important the word is, i.e., if the word is common or rare across the document.

$$TF = \frac{No. of\ times\ a\ term\ appears\ in\ doc}{Total\ number\ of\ terms\ in\ a\ doc}$$

$$IDF = \ln\left(\frac{No. of\ docs}{No. of\ docs\ the\ term\ appears\ in}\right)$$

This method is used widely to rank content. The basic idea behind this method is to weigh down the frequent terms while scaling up the rare terms.

The cleaned data in the previous step is passed through the TfIdfVectorizer function provided by the sklearn.feature_extraction.text module. The stop_words parameter in this function also acts as a second filter to remove the stop words.

Clustering Algorithms

The four clustering algorithms used to build models for this project are as follows,

Agglomerative Clustering

It is the most used type of hierarchical clustering to group data points into clusters, works on the principle of bottom-up approach. This algorithm works in such a manner that each data point or object is considered a single-element cluster. Upon each iteration, any two clusters having the most similarity measure between them are combined to form a new cluster, that is, the algorithm clusters points that have least distance using the linkage function specified. This process is repeated until all points become a part of a single root cluster. The most preferred linkage methods are ward (works on minimizing variance of clusters being merged) and complete (makes use of the maximum distances between all points of the two clusters being merged).

This project uses the ‘ward’ method to link clusters and hence, Euclidean distance to compute similarity.

$$EuclideanDist = \sqrt{[((x2-x1)^2) + ((y2-y1)^2)\ ]}$$

where (x1,y1) and (x2,y2) are coordinates of points

This algorithm performed better when compared to all the other algorithms that were built for analysis.

K-Means Clustering

This algorithm works on the principle of finding the optimal centroid to identify the K number of groups in the dataset. Since the number of clusters is already known it is also called the flat clustering algorithm. The algorithm works by choosing K data points at random and assigning each to a cluster and categorizing them based on the number of data points. Now, points are clustered based on the centroids. During each iteration the following steps are performed, the distance between data points and centroids are calculated, each data point is allocated to a cluster closest to centroids and finally compute the centroids for the clusters by averaging all the data points of the clusters. There are primarily two important steps in this algorithm wherein data points are assigned to the nearest cluster during Expectation-step and computation of the centroids of each of the clusters called the Maximization-step. It is important to standardize the data such that the data has a mean of 0 and standard deviation of 1. Also, during the iterative process there is a possibility that the algorithm does not converge to a global optimum and hence, it was observed that random initializations and low squared-sum distance yield better results.

This project uses k-means++ for initialization that purely selects and initializes cluster centroids based on empirical distribution of data during sampling and thereby, results in faster convergence.

DBSCAN

This is a density-based clustering algorithm as the name suggests and is very useful when we need to detect data patterns from large input data of different shapes and sizes to form clusters or groups. In simple words, this algorithm treats points in high density areas to form clusters and in low density areas to be noise. Some advantages of this algorithm over hierarchical clustering methods are, it does not require users to specify the number of clusters needed and slight variations in the orientation of data points does not affect the end results of this as it does not depend on the mean value of data points present in the system. The two key parameters used in this algorithm are Epsilon Value (eps) - a distance measure to locate the points that lie in the neighbourhood and Minimum points (minPts) - represents the number of points threshold value for a cluster to be considered a high dense area. The algorithm works on the principle of Connectivity - that contains information about the points present in a particular cluster and Reachability - that talks about by how much each of the points are reachable from one another (within a particular eps value). In each of the clusters thus formed during the process we get three types of points, they are, core, border, and noise points. The points that have at least minPts within a particular range of eps are core points and the points that lie at eps distance from a core point are called border points. All other points are classified as noise points. The algorithm initially visits all the points in the dataset in an arbitrary fashion, clusters all the minPts that lie within the mentioned eps value into a cluster. This process then happens recursively to expand the clusters formed. Therefore, it becomes quite evident that parameters such as minPts and eps play a pivotal role in determining how clusters are formed. As a part of parameter estimation, it is found that minPts should be set to at least 3 and any value below yields results same as hierarchical clustering. In general, for large data, minPts is set equal to twice the dimension of the dataset. The ideal value of eps can be chosen from a k-distance graph that is a plot of distance ranked from the largest to the smallest value of k = minPts-1 nearest neighbors. The eps values are often chosen from the elbow of this graph. If the chosen eps is very small, a large chunk of the data will remain unclustered; whereas for too high eps, merging of clusters results in most of the points belonging to the same cluster. It is of paramount importance to choose appropriate distance functions to obtain expected clustering results.

This project tried to embody this algorithm using multiple combinations of eps and minPts. However, since the data was sparse the algorithm could not cluster the data points precisely as expected. It was observed that DBSCAN is better when data is of disproportional densities and can be separated in a non-linear manner.

Spectral Clustering

It is one of the top growing algorithms today and works better than most of the traditional methods for clustering and solves unlabeled data problems by graph-partitioning method wherein it treats every data point in the dataset as a graph-node. This algorithm based on graph theory focuses on identifying groups of nodes based on the edges connecting them. Three core steps of this algorithm are building a similarity matrix/graph, projecting this data to a rarer dimensional space, and clustering the reduced data.

The concepts of Eigenvectors and Eigenvalues play a critical role in this algorithm as they provide information about the dynamics of the matrix representation of data. It is known that if there exists a vector x (not containing all zeros) and a scalar factor $\boldsymbol{\lambda}$ such that there exists a relation,

$$A\ x = \boldsymbol{\lambda}\ x$$

then, x is the eigenvector of A for a corresponding eigenvalue $\boldsymbol{\lambda}$. Graphs contain nodes and edges that connect them, these can either be undirected or directed and can sometimes have associated weights. As a naive approach, it is easy to cluster all nodes associated with these connected components to form a cluster. But there is a possibility that all nodes can be connected, or these components may be too large to form a single cluster. To start with the algorithm, a graph, often represented as an Adjacency matrix A_{ij}, is created where the nodes are represented by rows and columns, and the entries denote existence or absence of an edge between the nodes. This similarity graph can be generated using a few approaches. The Epsilon-neighborhood graph, that is, points that lie within the vicinity of a eps are connected and this graph thus formed is often undirected and unweighted. K-Nearest Neighbors graph forms a connection between two nodes if they lie within a specified k parameter. This results in the formation of directed and weighted graphs. The graph can be converted to undirected by directing two nodes to each other if any one of them lies and/or in the k-nearest neighbors of the other. For fully connected graphs each node is connected by a weighted edge, that is the distance between the two nodes and is undirected. In this way all nodes are connected, and a Gaussian similarity measure is used for this. The next step is to create a Graph Laplacian Matrix, which is used to reduce the data to a much lower dimensional space. In this way, the points now lie closer in the new dimensional space and can then be clustered using traditional techniques. To create this matrix, the degrees of all the nodes (d_{ij}) are defined and a subsequently a degree matrix D_{ij} is formed.

$$L = D - A$$

The difference between Degree matrix and Adjacency matrix results in the Laplacian matrix (L). This resultant matrix is then normalized. The final step is to cluster the data obtained using traditional methods (mostly K-Means Clustering). In this way data points spread across a large dimensional space can also be clustered as expected much faster. However, this technique is not scalable due to the computations involved.

For this project, the affinity matrix was built on radial basis function rather than a graph of nearest neighbors as it achieved a better silhouette score.

Performance Metrics

The project looks at several performance metrics to analyze the clustering models in detail. Out of the metrics Silhouette score is used to compare different models. The metrics used and their brief description is as follows

Silhouette Score

This metric calculates the goodness of a clustering algorithm. It ranges from 1 to -1, with 1 being clusters are well separated, 0 meaning the distance between clusters is not significant and -1 being clusters are assigned incorrectly. It is usually more useful when the number of clusters are more than 3. This is also a method to calculate the optimal number of clusters.

Homogeneity Score

This score serves as a check to ensure that a cluster contains only samples belonging to a single class. It usually ranges between 0 to 1. When all samples in a cluster have the same label then the homogeneity score equals 1.

$$h = 1 - \frac{H(Y_{true}|Y_{pred})}{H(Y_{true})}$$

Completeness Score

This score on the other measure how similar samples is put together by the algorithm. It also ranges from 0 to 1. When all samples of a kind are assigned to a single cluster then the completeness score is 1.

$$c = 1 - \frac{H(Y_{pred}|Y_{true})}{H(Y_{pred})}$$

V-Measure Score

This score is nothing but the harmonic mean of Homogeneity and Completeness score. It also ranges from 0 to 1 and can be increased only if both completeness and homogeneity score both increases.

$$VMeasure = 2 \times \left(\frac{h \times c}{h + c}\right)$$

Adjusted Rand Index

Adjusted Rand Index is Rand Index score adjusted to chance. The Rand Index calculates the similarity between clusters. Like Rand Index (RI), Adjusted Rand Index (ARI) also ranges from 0 to 1.

$$ARI = \frac{RI - Expected\ RI}{Max(RI) - Expected\ RI}$$

Empirical Results

As mentioned earlier this project compares four different clustering models such as Agglomerative, DBSCAN, K-Means and Spectral. The results using the algorithms are discussed below.

Agglomerative Clustering:

The agglomerative clustering in sklearn.cluster module has a few parameters that can be tuned. This project mainly focuses on tuning the *linkage* and *affinity* parameters. The best set of parameters was chosen after running a gridsearchcv for the best silhouette score. The final set of chosen parameters where *n_clusters* = 8, *linkage* = 'ward' and *affinity* = 'euclidean'. Since these were the best parameters based on silhouette scores the other metrics were also checked to see how the model performed in depth.

Silhouette Score	0.1203
Homogeneity	0.673
V-Measure	0.695
Adjusted Rand Index	0.350

Completeness	0.719

The model is performing better in various aspects. From the completeness and homogeneity scores it can be inferred that the clusters and the samples are related and mostly similar.

DBSCAN:
Even though this is one of the best suited algorithms for text data, since the project uses fewer data which are farther apart this algorithm was not able to form clusters. By increasing the distance all the points appeared as outliers and by decreasing the distance each point formed a separate cluster. Thus, this approach was not able to explore more for this dataset.

Spectral Clustering:
For this algorithm using the sklearn.cluster module the *n_clusters*, *affinity*, and *eigen_solver* parameters were compared. Keeping *n_clusters* = 8, *affinity* = 'rbf' the best score was observed. Since change in *eigen_solver* did not affect the score, it was omitted. Following are other performance metrics for the best parameters chosen

Silhouette Score	0.1184
Homogeneity	0.607
V-Measure	0.634
Adjusted Rand Index	0.237
Completeness	0.664

These metrics also inform that the model performs well enough in separating the samples into meaningful clusters.

K-Means Clustering:
For this algorithm using the sklearn.cluster module the *n_clusters*, *algorithm* and *init* parameters were varied and compared. The best parameters observed during various iterations were *n_clusters* = 8, *algorithm* = 'full' and *init* = 'k-means++'. The other metrics observed are as follows,

Silhouette Score	0.0812
Homogeneity	0.678
V-Measure	0.688
Adjusted Rand Index	0.365
Completeness	0.669

Comparing the performance of these models the project concludes that the Agglomerative algorithm is best suited for the data chosen. It has higher silhouette scores than the rest of the algorithms even though its other metrics are slightly lower than the K-Means. Since silhouette score gives information on formation of non-overlapping clusters it becomes one of the important metrics to consider in this use case. Below figure shows the dendogram for the agglomerative model chosen.

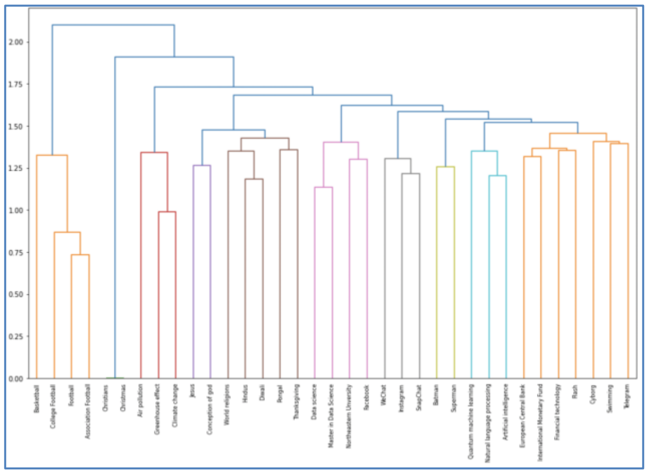

Fig 1: Clusters formed using agglomerative model

This project further used word cloud to visualize the clusters formed using the agglomerative algorithm. Below is a snapshot of the word cloud observed

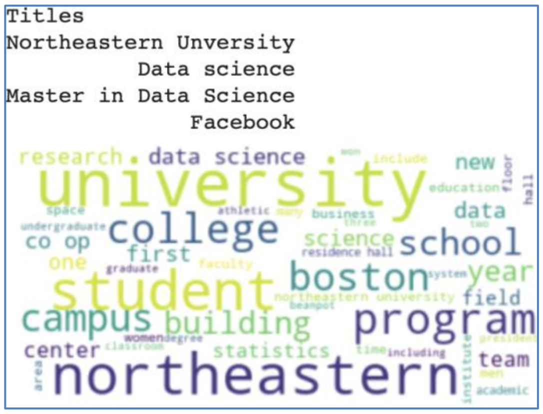

Fig 2: Data Science cluster

The above image represents one of the clusters formed. The most important words in this cluster are university, northeastern, Boston, student etc. Similarly, the titles clustered are Northeastern University, Data Science, Master in Data Science and Facebook. While the former three titles were originally placed in the Data Science cluster, Facebook was placed in the social media cluster. It might have been grouped with the Data Science clusters because of the words present in its content.

Summary

The primary focus of this project was to experiment with different clustering algorithms to find which suits better for the chosen text dataset. Algorithms like DBSCAN that are usually touted to perform better than hierarchical clustering algorithms for text data were unable to provide expected clustering results due to the varied sparseness of data. Best performing model was achieved using Agglomerative hierarchical clustering. Therefore, it was observed that traditional techniques can also achieve best model performance as opposed to latest and advanced clustering techniques depending on the data distribution and hyperparameters used. The next steps of this project could be, explore much larger datasets and compare the performance of the above 4 algorithms to find the best model. The best model thus obtained could have then been used to find out or organize files of folders using similarity of text.

References

[1] Github link to the code file: https://github.com/AkshayaMahesh/USMLProj_DS5230

[2] https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

[3] https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c

[4] https://towardsdatascience.com/v-measure-an-homogeneous-and-complete-clustering-ab5b1823d0ad

[5] https://scikit-learn.org/stable/modules/clustering.html

[6] https://www.geeksforgeeks.org/ml-spectral-clustering/

[7] https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html

[8] https://www.mygreatlearning.com/blog/introduction-to-spectral-clustering/