BLG 527E Machine Learning Project Report

İrem Beyza Onur [1], Elif Özcan [1]
[1] Istanbul Technical University, Department of Computer Engineering
İstanbul, Turkey
Email: {onur16, ozcane22}@itu.edu.tr

***Abstract*—In this study main goal is clustering the academic documents into clusters with different clustering algorithms, optimizing the performance of algorithm and comparing the results. For clustering we used K-means, Spectral Clustering, DBSCAN, and Gaussian Mixture models and represented the best performing model. While working with documents we also compared the embedding models and used our model for different area. The results show that between this 4 clustering algorithms Gaussian Mixture algorithm performed the best for our text data.**

***Index Terms*—clustering, document, model**

I. Introduction

Clustering is one of the most popular topics in data science which can be used for many tasks such as document clustering or customer clustering. Clustering can be considered as the grouping of data into smaller groups based on the pattern of the data. The clustering algorithm aims to place similar samples in the same cluster, whereas different samples into different clusters. Clustering is an unsupervised machine learning algorithm meaning that the labels are not known, therefore the model aims to learn data patterns, in other words, the data structure to cluster the data. There are common steps that have a significant impact on the clustering performance such as data transforming, data cleaning, and scaling, feature selection, clustering method selection, and evaluation. Since the machine learning algorithms can not process raw string data, therefore the string data must be transformed into numerical space. For this projection, the models that vectorize the data can be used. Before this projection to numerical space, the data must be cleaned, meaning that the invalid values duplications must be removed. One of the most crucial steps is feature selection since using non-informative features may make it difficult for the algorithm to make decisions and has a poor impact on the clustering performance. When the data is ready to be clustered, the model must be selected because different clustering algorithms achieve high performance on different tasks, for instance, spectral clustering algorithms stand out on the data that has a convex structure. The final step is the selection of the evaluation method that makes us able to make inferences about the model or data selection based on the selected performance metrics.

In this project, we studied paper clustering by their subject. We created our own dataset using the data provided by taking two different approaches. The provided data contains several features of the papers such as paper abstracts, authors, published conferences, and citations. For the first approach, we only took the paper abstract into account but we reached extremely bad results forcing us to follow another approach for dataset creation. As our second approach, we took the paper title, paper abstract, a paper published conference, cited paper title, cited paper published conference, cited paper abstract, and the conferences to cluster the papers by subject. We performed data cleaning by removing non-informative words and punctuations from the data, and transformed the text data into numerical space using pre-trained text-to-vec models. Principal Component Analysis (PCA) is used for informative feature selection and four well-known clustering algorithms (KMeans, Spectral Clustering, DBSCAN, and Gaussian Mixture) are used to cluster the papers by subject. In the final step, we evaluate the clustering performance by the Silhouette score and Calinski-Harabasz index. We reached the highest clustering performance (Silhouette score of 0.3340, 2546.707 of Calinski-Harabasz index) with the Gaussian Mixture algorithm using all-MiniLM-L6-v2 tex-to-vec model.

II. Literature Review

1) **DBSCAN Algorithm for Document Clustering** [1]
In this paper the main purpose was clustering text documents with Density-based spatial clustering of applications with noise (DBSCAN) algorithm and representing the results of parameter tuning of the algorithm. For dataset Reuters dataset, a collection of news published by Reuters agency is used. The dataset has 542 samples and each document represented in a vector space model as frequency of word occurrences in document with 1000 attributes after selection. For feature selection information gain is used. The data split into train and test set with ratio of 2:1 (%66 in train set, rest in test set). For parameter tuning no prior method is used, for epsilon values between 0.1 and 0.9 is used for both training and test set while MinPts is constant and results shown as seen in Figure-1. For the MinPts parameter, epsilon is chosen as 0.9 as a result of Figure-1 and 10 different MinPts values are used and results shown as seen in Figure-2. DBSCAN algorithm evaluated as number of samples that are left outside of cluster formed by algorithm, also known as noise. As for conclusion of the paper they

declared that “From the ‘Noise’ point of view we can observe that, when the number of MinPts increases, for the same value for ϵ, the number of considered noise samples increases also. This means that in the file the samples are not uniformly distributed, and this becomes a problem to group such type of samples. Also, because we can have a relatively a small number of samples, we can’t increase the MinPts value. The conclusion is that for text documents we need to use a large value for ϵ and a small number for MinPts.”

The only similarities between this study and ours is that we both used DBSCAN and text data has too much dimensions. As for differences for dimensionality reduction we used PCA and for feature selection we used domain knowledge while in this study information gain is used. For the sake of hyparameter choosing we used ”NearestNeighbors” algorithm for epsilon value.

2) **Deep k -Means: Jointly clustering with k -Means and learning representations** [2]

This study mainly focuses on k-means related deep clustering problem, for high dimension data that standard clustering algorithms such as k-means and Gaussian Mixture Models have a hard time clustering. For evaluating the clustering results compared with state-of-art standard and k-means related deep clustering models. The suggested methods tested on different dataset of both images and text collections, image datasets being MNIST and USPS and text collections 20NEWS and RCV1-v2 dataset.

As goal of this work is to study the k -Means clustering algorithm in embedding spaces, they mainly focus on the k -Means-related models and compare approaches against state-of-the-art models from this family using both standard and deep clustering models. The pretraining we performed here simply consists in initializing the weights by training the auto-encoder on the data to minimize the reconstruction loss in an end-to-end fashion–greedy layer-wise pretraining did not lead toimproved clustering in our preliminary experiments. The proposed Deep k -Means (DKM) is, as DCN, a “true”k -Means approach in the embedding space; it jointly learns AE-based rep- resentations and relaxes the k -Means problem by introducing a parameterized softmax as a differentiable surrogate to k -Means argmin. For all deep clustering approaches, the training is based on the Adam optimizer with standard learning rate = 0 . 001 and momentum rates ß1 = 0 . 9 and ß2= 0 . 999 . The minibatch size is set to 256 on all datasets following. Each model is trained on the whole data and only the validation set labels are leveraged in the line search to identify the optimal hyperparameters. As for evaluating Normalized Mutual Information (NMI) and the clustering accuracy (ACC) are used. As for conclusion of the paper they declared that “The AE-KM(Auto Encoder K means) method, that separately performs dimension reduction and k -Means clustering, overall obtains the worst results. While the quality of the clustering results and that of the representations learned by the models are likely to be correlated, it is relevant to study to what extent learned representations are distorted to facilitate clustering. The experiments conducted on several datasets confirm the good behavior of Deep k -Means that outperforms DCN, he current best approach for k -Means clustering in embedding spaces, on all the collections considered.”

This study is proposed a k-means related deep clustering and compared their method with other related work and they to indicated in the paper that “several studies have proposed solutions to this problem with different clustering losses, to the best of our knowledge, this is the first approach that truly jointly optimizes, through simple stochastic gradient descent updates, representation and k -Means clustering losses.”. The difference is obviously that they used deep learning methods for escaping the curse of dimensionality.

3) **The Determination of Cluster Number at k-mean using Elbow Method and Purity Evaluation on Headline News** [3]

In this study, the eagerness to reach the information fastly is focused on due to the dramatic growth in technological developments. Rather than avoid spending too much time learning what’s going on outside, people want to stay up-to-date on new developments in many fields. For these reasons, categorizing the information into groups by their subjects can be helpful to facilitate the flow of information. For the categorizing, the authors stated that a thousand news titles were retrieved from various news sites to be used in the experiments since they are very informative, clear, short, and descriptive. TFIDF is used for the text feature calculation method, whereas KMeans is chosen to be the clustering method with the Elbow Method for the cluster optimization. For the evaluation, the Purity Method which gives information about the similarity between real and predicted clusters is used. This paper mainly focused on choosing the optimum number of clusters in which the KMeans algorithm is used, and concludes that the ELbow Method can be an efficient choice to decide the optimum number of clusters. Figure 5 shows the flow chart of the clustering system proposed in the paper.

As seen in Figure 5, the proposed system has a pre-processing block before the clustering algorithm which contains four steps. Tokenization cleans the data by removing the periods and commas which are the most common instance in headlines, whereas ”stop word removal” also cleans the data by removing prepositions and pronouns which give no information about the content of the title. After the data cleaning process, text weighting is applied using TF-IDF which represents each document in the dataset as N-dimensional vectors. After term weighting, PCA is applied to feature vectors to represent the data in a lower dimensional space. PCA briefly can be considered as an algorithm that eliminates the features

ε	MinPts	Nr. clusters	C1	C2	C3	C4	C5	C6	C7	Noise	Learning Rate
0,9	6	3	102	61	19					176	50,84%
0,8	6	3	100	56	19					183	48,88%
0,7	6	3	94	49	19					196	45,25%
0,6	6	3	94	45	19					200	44,13%
0,5	6	6	41	7	27	7	11	8		257	28,21%
0,4	6	6	41	7	27	7	11	8		257	28,21%
0,3	6	6	40	7	25	7	11	8		260	27,37%
0,2	6	7	25	6	16	6	10	6	8	281	21,51%
0,1	6	6	10	6	15	6	13	7		301	15,92%

Fig. 1: Choosing epsilon value [1]

ε	MinPts	Nr. clusters	C1	C2	C3	C4	C5	C6	C7	...	Noise	Learning Rate
0,9	2	11	2	102	61	19	2	2	2	...	158	54,60%
0,9	3	3	104	62	19						173	51,68%
0,9	4	4	102	61	19	4					172	51,96%
0,9	5	3	102	61	19						176	50,84%
0,9	6	3	102	61	19						176	50,84%
0,9	7	3	102	59	19						178	50,28%
0,9	8	3	103	61	19						175	51,12%
0,9	10	3	101	59	19						179	50,00%
0,9	15	3	101	59	19						179	50,00%
0,9	20	2	101	59							198	44,69%

Fig. 2: Choosing MinPts value [1]

Table 2
Clustering results of the DKM and IDEC methods. Performance is measured in terms of NMI and ACC (%); higher is better. Each cell contains the average and standard deviation computed over 10 runs. Bold (resp. underlined) values correspond to results with no significant difference ($p > 0.05$) to the best approach with (resp. without) pretraining for each dataset/metric pair.

Model	MNIST		USPS		20NEWS		RCV1	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
Deep clustering approaches without pretraining								
IDEC$^{\text{-p}}$	61.8 ± 3.0	62.4 ± 1.6	53.9 ± 5.1	50.0 ± 3.8	22.3 ± 1.5	22.3 ± 1.5	**56.7 ± 5.3**	**31.4 ± 2.8**
DKM$^{\text{a}}$	82.3 ± 3.2	78.0 ± 1.9	**75.5 ± 6.8**	73.0 ± 2.3	44.8 ± 2.4	42.8 ± 1.1	53.8 ± 5.5	28.0 ± 5.8
Deep clustering approaches with pretraining								
IDEC$^{\text{p}}$	**85.7 ± 2.4**	**86.4 ± 1.0**	**75.2 ± 0.5**	74.9 ± 0.6	40.5 ± 1.3	38.2 ± 1.0	**59.5 ± 5.7**	**34.7 ± 5.0**
DKM$^{\text{p}}$	**84.0 ± 2.2**	79.6 ± 0.9	**75.7 ± 1.3**	**77.6 ± 1.1**	**51.2 ± 2.8**	**46.7 ± 1.2**	**58.3 ± 3.8**	**33.1 ± 4.9**

Fig. 3: Clustering Reslut of DKM and IDEC methods [2]

that will not cause loss of information.

The Elbow method is based on the idea that there is an optimum clustering number where adding more clusters does not make a significant change in clustering performance.

To find the optimum number of clusters, cluster number is increased one by one, and the SSE (Sum Square Error) is calculated for each of them as in Equation (1).

$$SSE = \sum_{K-1}^{K} \sum_{x_i} |X_i - C_k| \tag{1}$$

where X_i and C_k respectively represent the i. point and center of the k. cluster, meaning that SSE describes the distance between the point and cluster center it is assigned to. To evaluate the number of cluster found by the Elbow method, the Purity method which is based on labeled data is used. The purity method represents the percent of of the total samples that were correctly classified according to ground truth. The Purity method is calculated as in Equation (2).

$$Purity = \frac{1}{N} \times \sum_{i=1}^{k} max_j \times |c_i \cap t_j| \tag{2}$$

where N, k respectively represent the numbers of samples and clusters, whereas c_i and t_j indicate the cluster in C and the classification which has the max count for cluster c_i.

The purity that equals 1 means that all the samples are classified correctly, otherwise, all the samples are misclassified.

The authors performed all the steps mentioned earlier and reached the optimum number of clusters of 8 using the Elbow Method in the range of 2 and 8. Figure 5 shows the SSE values obtained by different cluster numbers.

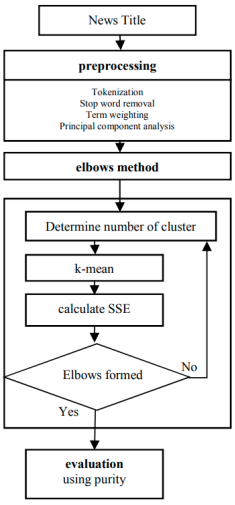


Fig. 4: The Flow Chart of the News Clustering System. [3]

As seen in Figure 5, the SSE difference between clusters

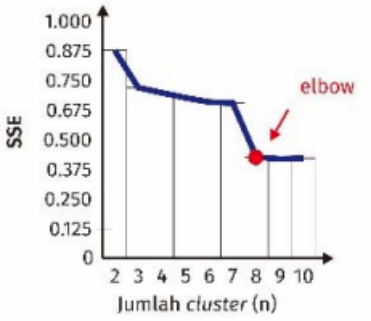


Fig. 5: SSE with different cluster numbers. [3]

7 and 8 is dramatically large compared to other ones, and after that point, SSE values do not change so much. Then, the number 8 is decided to be the ”elbow” point.
The purity is calculated for each cluster as in Figure 6.
As seen in Figure 6, the highest purity is obtained with

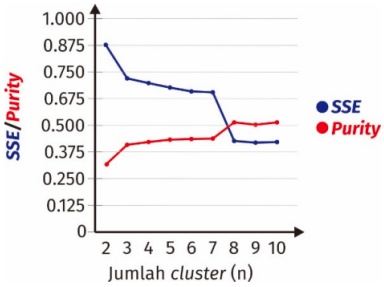


Fig. 6: The purity with different cluster numbers. [3]

8 clusters and started to decrease after that point. The authors concluded that the elbow method is an efficient way to calculate the optimum number of clusters in the KMeans algorithm as evidenced by the fact that the highest purity score was obtained at the ”elbow” point.

4) **Improving Spectral Clustering with Deep Embedding and Cluster Estimation** [4] Spectral clustering is one of the most popular clustering algorithms that stands out when the data has a convex structure. It is a promising clustering method since it uses the eigenspace of the Laplacian matrix obtained using similarities between the data itself for the KMeans clustering method, and takes advantage of these similarities between data points compared to the KMeans algorithm. But the algorithm achieves poor results when the data has a complex structure and deciding the number of clusters is always an issue, especially in these cases. In the case of complex data structures or very high-dimensional data, using deep learning models is the first solution that comes to mind. For this reason, authors proposed [4] to use a deep autoencoder to obtain better feature representations and then perform KMeans clustering on these representations. They also proposed using a softmax autoencoder to estimate the number of clusters. They evaluated their approach on four images and two text datasets. They used metrics of Normalized Mutual Information and the Adjusted Rand Index that are used for the supervised clustering algorithms.
They shared the results as in Figure 7.
As seen in Figure 7, the proposed algorithm outperforms

TABLE II
ESTIMATION OF THE NUMBER OF CLUSTERS AND RUNNING TIME (SEC.) OF EACH METHOD ON ALL DATASETS.

Method	MNIST		Fashion-MNIST		USPS		STL-10		Reuters-8		20Newsgroups	
	k	Time	k	Time	k	Time	k	Time	k	Time	k	Time
Ground Truth	10	-	10	-	10	-	10	-	8	-	20	-
X-means	20	357	20	321	20	19	20	235	16	56	40	470
G-means	15	732	16	647	20	41	20	605	16	121	40	1892
DB	20	912	3	785	18.8	37	6.6	588	2	114	34.8	1840
SS	2	2972	2	2935	5	88	3	664	15.4	150	2	2232
SA (ours)	11.2	**85**	12	**88**	6.6	**10**	2.8	**56**	12.6	**24**	31.6	**59**
DE+X-means	20	1304	20	1328	20	169	20	474	16	164	40	508
DE+G-means	9	1301	19.4	1352	20	172	16	475	16	165	40	550
DE+DB	10.8	1313	6.2	1340	17.8	169	8.8	473	14.4	162	25	536
DE+SS	9	2475	5.2	2413	12.2	222	7.8	512	14.8	196	2	701
DE+SA (ours)	**10.2**	1301	**9.2**	1327	**9.6**	167	**9.8**	469	**8**	162	**20.8**	494

The best estimation and running time are highlighted in boldface. The k is a floating number since it is the average of found ks.

Fig. 7: The experiment results of the proposed algorithm. [4]

several well-known algorithms with different numbers of clusters. On the other hand, the authors stated that the selected number of clusters are quite close to the number of ground truth cluster, meaning their proposed cluster number estimation method has a good impact on the clustering performance. In conclusion, feature representations are one of the most crucial factors that has an impact on the clustering performance, and by using deep features produced by autoencoders, spectral clustering performance can be increased.

III. METHODOLOGY

A. Pre-trained Text-to-Vec Models

In this project, we used three different pre-trained text-to-vec models [5], based on their clustering performances

and speeds. The texts must be converted to numbers because learning algorithms can not process raw strings, the text-to-vec models are involved at this point. The models are used to project the text data into numerical space.

A summary of all Pre-trained Text-to-Vec Models is as in Figure 8. As seen in 8, all-mpnet-base-v2 has the highest performance, whereas the paraphrase-MiniLM-L3-v2 model is the fastest model, therefore we chose these two models besides all-MiniLM-L6-v2 model.

1) all-MiniLM-L6-v2
   The model is trained on the large dataset which contains approximately 20 different datasets, aiming to map sentences paragraphs to a 384-dimensional dense vector space.
2) all-mpnet-base-v2
   Unlike the first model, this model maps sentences paragraphs to a 768-dimensional dense vector space.
3) paraphrase-MiniLM-L3-v2
   Like the first model, this model maps sentences paragraphs to a 384-dimensional dense vector space.

B. Clustering Methods

1) K-means: K-Means clustering aims to learn the model of the data, as an output of this, it divides the data into subgroups such that similar samples are in the same cluster and different samples are in different clusters based on a similarity metric. This similarity metric is application-specific and can be chosen based on euclidean or correlation. This clustering can be performed on the data itself or its features, we studied feature-based clustering in this project. The clustering can be considered an unsupervised learning algorithm since it is aimed to learn the data structure based on the data itself or its features, not its true labels. It is commonly used in market research, image segmentation, or document clustering as in this project. KMeans algorithm aims to assign each data sample to a different cluster, such that making the samples in the same cluster as similar as possible, whereas making the samples in the different clusters as different as possible. The most crucial point in this algorithm is the selection of the optimum number of clusters. The optimum number of clusters is chosen by minimizing the sum of the squared distance between the data points and the center of the corresponding cluster.

The steps of the algorithm are given in Figure 9.

As seen in Figure 9, the first step of the KMeans algorithm is chosen to K random points as the centroids of each cluster. Then each data sample is assigned to the cluster that is closer than other clusters based on the Euclidean distance between them. After the assignment, the centroid of the clusters is updated by averaging all the data points belonging to the cluster. The data-cluster assignment is re-performed until the assignments do not vary. The KMeans algorithm is an Expectation-Maximization problem, the assigning each sample to a cluster is performed in the Expectation part, whereas the center of the clusters is re-calculated in the Maximization part. Equation (3) represents the objective function to be minimized in the E part.

$$J = \sum_{1}^{m} \sum_{k=1}^{K} w_i \times |x_m - c_k|^2 \tag{3}$$

where m and k indicate the numbers of samples and clusters, whereas x_m and c_k represent the sample and center of the cluster. On the other hand, w_i gives the weight such that equals 1 if the data sample x_m is a member of c_k, otherwise 0. The minimization (E part) is performed by differentiating the objective function wrt. w_i as in Equation (4).

$$\frac{\partial J}{\partial w_k} = \sum_{1}^{m} \sum_{k=1}^{K} |x_m - c_k|^2 \tag{4}$$

The updating of the center of clusters (M part) is calculated by differentiating Equation (5) wrt. c_k as in Equations (5) and (6).

$$\frac{\partial J}{\partial c_k} = 2 \times \sum_{1}^{m} w_i \times |x_m - c_k| \tag{5}$$

$$c_k = \frac{\sum_1^m w_{ik} \times x_m}{\sum_1^m w_{ik}} \tag{6}$$

In the Python implementation, sklearn libraries are used, and the built-in KMeans class takes the number of clusters, the method that initializes the center of the initial clusters, the number of times the algorithm runs with different seeds, and the maximum number of iterations for a single run as an input. As stated before, the number of clusters is the most crucial thing for the KMeans algorithm. In this project, the Elbow method described in the previous section is used.

2) Spectral Clustering: Spectral clustering is one of the commonly used unsupervised machine learning algorithms for clustering tasks. The main difference from the KMeans algorithm is that Spectral Clustering clusters the data points based on their affinity instead of absolute locations as in KMeans. For this reason, spectral clustering is much more efficient than KMeans on data that have complex shapes.

The only data form to which spectral clustering can be applied is the graph of connected nodes. For this reason, the first step of the algorithm is building a similarity graph from the data using the euclidean distance or k-nearest neighbor algorithm. Then, the adjacency matrix can be formed from the similarity matrix by thresholding the values. The threshold value is set to a specific number and the elements larger than that point are set to 1, otherwise to 0. The value of 1 here indicates that elements in corresponding rows and columns are connected to each other, so formed data can be considered as a graph of connected nodes. Then, the degree matrix which is the diagonal matrix containing the degrees of each node. The degrees of each node are computed by summing all the elements of the corresponding rows.

Finally, the Laplacian matrix is calculated as in Equation (7)

$$L = D - W \tag{7}$$

All models

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-distilbert-cos-v1	65.98	52.83	59.41	4000	250 MB
multi-qa-MiniLM-L6-cos-v1	64.33	51.83	58.08	14200	80 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
all-MiniLM-L6-v2	68.06	49.54	58.80	14200	80 MB
paraphrase-multilingual-mpnet-base-v2	65.83	41.68	53.75	2500	970 MB
paraphrase-albert-small-v2	64.46	40.04	52.25	5000	43 MB
paraphrase-MiniLM-L3-v2	62.29	39.19	50.74	19000	61 MB
paraphrase-multilingual-MiniLM-L12-v2	64.25	39.19	51.72	7500	420 MB
distiluse-base-multilingual-cased-v1	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilinqual-cased-v2	60.18	27.35	43.77	4000	480 MB

Fig. 8: Pre-trained text-to-vec Models

Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

Fig. 9: Pseudo Code of KMeans Algorithm

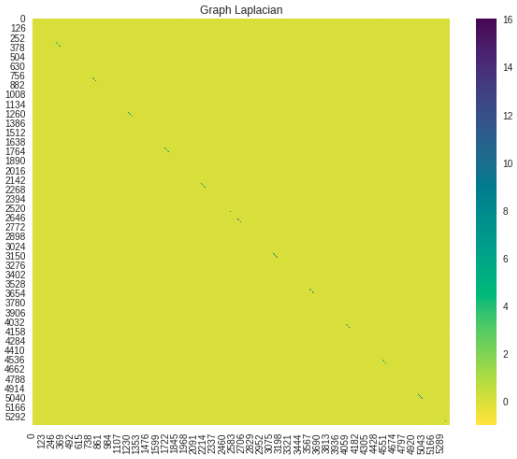


Fig. 10: The Laplacian Graph

where D and W denote the degree and adjacency matrices, respectively.
The Laplacian matrix is symmetric and positive semi-definite so that its all eigenvalues are real and non-negative. The eigenvalues of the Laplacian becomes larger as the number of connection of the nodes increases. In fact, the number of 0 valued eigenvalues gives the number of the connected components in the graph. The first non-zero eigenvalue is called the spectral gap which gives the density of the graph, if it equals zero, then the graph is densely connected which means all nodes are connected to each other. The second non-zero eigenvalue corresponds to the Fiedler vector. The Fiedler value gives the minimum number of graph cuts to separate the graph into subgraphs. If the graph is already divided into two graphs, then the Fiedler value becomes zero. To summarize, the eigenvalues that are equal to zero give the number of connected components, whereas the ones that are near zero indicate that there is almost a separation between two components. In light of this information, we use the eigenvectors corresponding to eigenvalues that are near zero to project the data to a lower dimension and feed it to KMeans Algorithm.
Figure 10 represents the Laplacian graph.
The diagonal elements of the Laplacian graph represent the degrees of the nodes, and off-diagonals indicate the negative edge weights.

3) Density-Based Spatial Clustering of Applications with Noise (DBSCAN): [6][7] DBSCAN, that was proposed by Martin Ester et al. in 1996, is a density based clustering algorithm which is groups data samples together based on their distances and minimum number of points to core. Algorithm works on the belief of that clusters are dense regions and also marks outliers which are too far away from their neighbors(Figure-11). On contrary to basic clustering algorithms such as k-means DBSCAN algorithm is able to find non-linear shaped clusters as it is based on density as seen at Figure-12. Also DBSCAN algorithm does not require the number of clusters beforehand, which is one of the biggest dilemma of clustering algorithms.
Algorithm mainly needs two parameters epsilon (ϵ) and

minimum number of points, where ϵ is the maximum distance between 2 neighbor points and $min\_samples$ is the minimum number of points to form a dense region.

For the parameter estimation there exists generally used methods such as for ϵ using NefarestNeighbors where the maximum curvature is the optimal ϵ value and it is said that for the $min\_samples$ using domain knowledge is the best but as a general rule, a minimum number off points can be derived from a number of dimensions (D) in the data set, as $min\_samples \geq$ D + 1.
The disadvantage of the algorithm is that in high-dimensional data, the curse of dimensionality says that all distances become similar so DBSCAN does not really work well on high dimensional data.

4) Gaussian Mixture: Gaussian mixture models assume that there are certain number of Gaussian distributions in data and each of this distributions are a cluster, and attempts to find mixture of multi-dimensional Gaussian probability distributions in the data, means that GMMs try to group data of a distribution into a cluster.
GMM clustering can accommodate clusters that have different sizes and correlation structures within them. Therefore, in certain applications, GMM clustering can be more appropriate than methods such as k-means clustering(Figure-13). Also GMMs can do both soft clustering and hard clustering since GMMs are actually probabilistic models. Algorithm can sometimes miss the globally optimal solution, and thus in practice multiple random initializations are used.
As for parameters of Gaussian mixture model, number of clusters are mandatory just like many clustering algorithms. For number of clusters the value that minimizes the Akaike information criterion (AIC) or the Bayesian information criterion (BIC) generally used or number of clusters that come from k-means elbow method. The $init\_params$ parameter which represents the how to choose initial center points of cluster is also import, this parameter can get values 'kmeans, kmenas++, random, $random\_from\_data$'. The parameter $covariance\_type$ must be one of the followings: 'full' which means each component has its own general covariance matrix, 'tied' which means all components share the same general covariance matrix, 'diag' which means each component has its own diagonal covariance matrix or 'spherical' which means each component has its own single variance.

$$P(x \mid \lambda) = \sum_{k=1}^{M} w_k \mathcal{N}(x \mid \mu_k, \sigma_k)$$

IV. Data Preparation & Experimental Results

While creating the dataset we used 6 features as follows: "ppaer_title, paper_abstract, paper_published_conference, cited_paper_title, cited_paper_published_conference, cited_paper_abstract". For text features we combined them together and embedded them 3 different pre-trained models, the reason of combining the text data is to escape from the dimensionality and for the other 2 features, paper_published_conference and cited_paper_published_conference, we categorized them with integer numbers. After embedding the text feature and categorized the other 2 columns we split data for tarin-test with 0.7 to 0.3 ratio and used PCA to reduce the dimensions of embedded data and add other 2 categorized data to both train and test data. While reducing the dimensions we trained PCA only with the train data. Before combining the text features we tried with not combining them but because of the curse of dimensionality clustering algoritms performed poorly.

The dataset has 7719 rows in total, 2316 in test 5403 in train set. The data embedded with "all-MiniLM-L6-v2" pre-trained model has 197 featuers, "all-mpnet-base-v2" has 262 features and "paraphrase-MiniLM-L3-v2" has 173 features after PCA (Figure-16).

A. Clustering Performance Evaluation Metrics

- Silhouette Score [11]: The Silhouette Score is used to measure the separation distance between clusters. It displays a measure of how close each point in a cluster is to points in the neighboring clusters 14. This measure has a range of [-1, 1] and inspect the similarities within clusters and differences across clusters. 1: Means clusters are well apart from each other and clearly distinguished. 0: Means clusters are indifferent, or we can say that the distance between clusters is not significant.
  -1: Means clusters are assigned in the wrong way.

  $$SilhouetteScore = (b - a)/max(a, b)$$

  ...where a is average intra-cluster distance i.e the average distance between each point within a cluster and b is average inter-cluster distance i.e the average distance between all clusters.
- Calinski-Harabasz Index [11]: The score is defined as the ratio between the within-cluster dispersion and the between-cluster dispersion. A high CH means better clustering since observations in each cluster are closer together (more dense), while clusters themselves are further away from each other (well separated). For calculating the CH index first step is to calculate inter-cluster dispersion or the between group sum of squares (BGSS) which is calculated as seen in Equation-8

  $$BGSS = \sum_{k=1}^{K} n_k * \parallel C_k - C \parallel^2 \tag{8}$$

  where n_k is the number of observations in cluster k, C_k is the centroid of cluster k, C is the centroid of the dataset (barycenter) and K is the number of clusters.
  The second step is is to calculate the intra-cluster dispersion or the within group sum of squares (WGSS) as seen at Equation-9.

  $$WGSS_k = \sum_{i=1}^{n_k} \parallel X_{ik} - C_k \parallel^2 \tag{9}$$

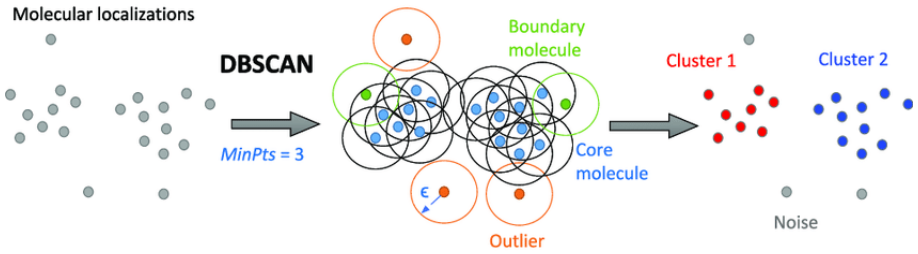


Fig. 11: An Example Illustrating the Density-Based DBSCAN Clustering Method Applied to SMLM Data [8]

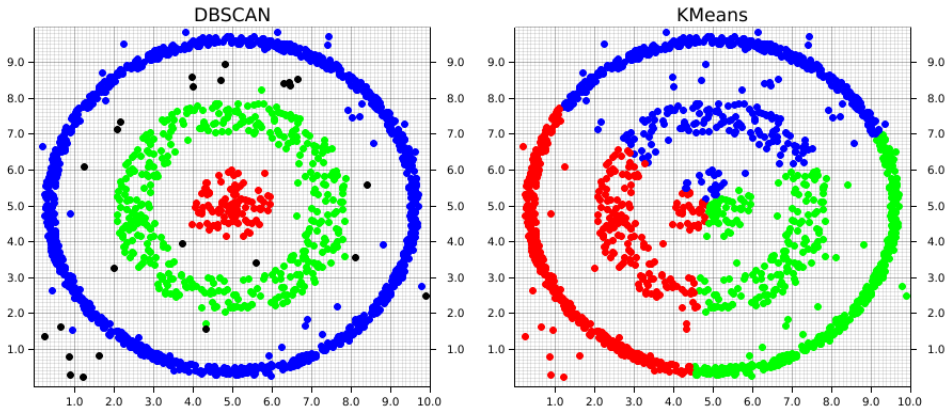


Fig. 12: DBSCAN vs K-Means Clustering [9]

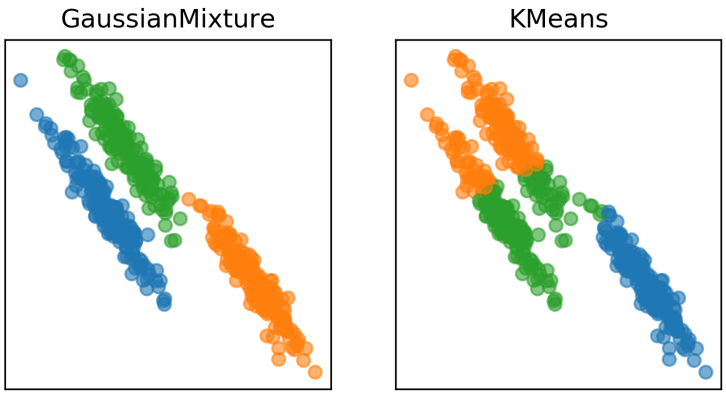


Fig. 13: GM vs K-Means Clustering [10]

where n_k is the number of observations in cluster k, X_{ik} is the i-th observation of cluster k and C_k is the centroid of cluster k. And the last step is to calculate CH index.

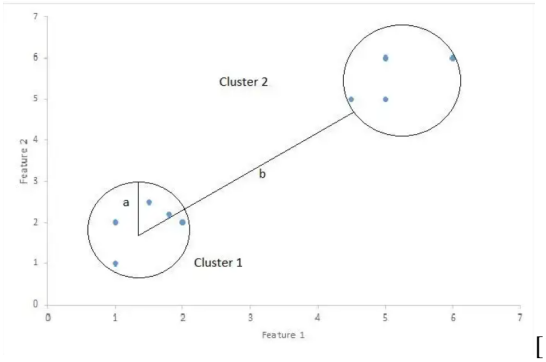
$$CH = \frac{BGSS}{WGSS} * \frac{N-K}{K-1} \tag{10}$$

B. Experiments & Results

1) ***K-means****:* For KMeans clustering, the three pre-trained models are used to project the data into the embedding space. After the vectorization, the PCA algorithm is used as in mentioned in Section 2. PCA algorithm aims to project the data into a lower dimension keeping the features that carry most of the information. Before the dimension reduction with PCA, the features are standardized to ensure that each feature contributes equally. The standardization is performed using built-in sklearn library in Python, with the math behind as in Equation (11).

$$z = \frac{x-m}{std} \tag{11}$$

where x, m, and std denote the data sample, mean value, and standard deviation, respectively.



[12]

Fig. 14: Silhouette Score

The scaled data is fed to the PCA algorithm which is also implemented by the built-in sklearn library. As mentioned before PCA aims to transform the high dimensional data into lower dimensional space while preserving its "important" features. The crucial part here is to find "important" features which the PCA accomplishes by using the variance of the data. The variance represents the information in the data and PCA aims to reduce the dimension of the data by keeping features that have a high variance. In this project, we performed PCA algorithm to select the meaningful features to cluster papers by their subject.
The pseudocode of the PCA algorithm is given in Figure 15.

Algorithm 1 Principal Component Analysis

procedure PCA
Compute dot product matrix: $\mathbf{X}^T\mathbf{X} = \sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})^T(\mathbf{x}_i - \boldsymbol{\mu})$
Eigenanalysis: $\mathbf{X}^T\mathbf{X} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$
Compute eigenvectors: $\mathbf{U} = \mathbf{X}\mathbf{V}\boldsymbol{\Lambda}^{-\frac{1}{2}}$
Keep specific number of first components: $\mathbf{U_d} = [\mathbf{u}_1, \ldots, \mathbf{u}_d]$
Compute d features: $\mathbf{Y} = \mathbf{U_d}^T\mathbf{X}$

Fig. 15: The Pseudocode of PCA Algorithm.

As seen in Figure 15, the correlation matrix of the data is computed, then eigenvalues are computed by eigenvalue decomposition. A specific number of eigenvalues are selected and the data is projected using eigenvectors that correspond to selected eigenvalues. We decided on the specific number of principal components by the explained variance of components. Explained variance is a statistical measure of how much of the total variance can be represented by each principal component. For three pre-trained models, the PCA algorithm is performed and a selected number of components are decided with the help of explained variance.
Explained variances of each component are computed and plotted with cumulative variance as in Figure 16.

As given in Figure 8, the highest performance has been reached by all-mpnet-base-v2, whereas paraphrase-MiniLM-L3-v2 model has the lowest performance. On the other hand, it can be observed that embedding produced by all-mpnet-base-v2 can be represented the maximum number of components, while the one obtained by paraphrase-MiniLM-L3-v2 can be represented the minimum number of components. In light of this information, we can conclude that the model with the highest performance (paraphrase-MiniLM-L3-v2) produces the embedding that carries the most information so that the number of components that be able to represent this data becomes the maximum one.
After the feature selection with PCA, KMeans algorithm is performed as described in the previous section, the best pre-trained model is selected based on the test results and a hyperparameter search is performed.
The initial parameters used in the KMeans algorithm are given in Table I.

TABLE I: Initial Parameters of KMeans Algorithm.

	n_cluster	init	n_init	max_iter
Kmeans	501	k-means++	10	300

The train and test performance were reported with the Silhouette and Calinski Harabasz Scores as in Table II.

As seen in Table II, the train performances belonging to all-MiniLM-L6-v2 and paraphrase-MiniLM-L3-v2 are quite close to each other, while the gap starts to increase on the test set. Model all-MiniLM-L6-v2 has reached the highest performance on the test set based on both Silhouette Score and Calinski Harabasz Score. For the training phase, 501 clusters were selected using the elbow method described in the previous section.
Figure 17 represents the elbow graphs of the three models. As seen in Figure 17, the optimum number of clusters chosen is 501 which is the elbow point. On the other hand, the lowest distortion score at this point is belonging to the Model all-MiniLM-L6-v2 model as parallel to training performance results as given in II. In addition, the visualization results of silhouette scores received in the training phase are shown in Figure 18.

As seen in Figure 18, the red line gives the mean value of silhouette scores. The lines that lay in the left side of the zero can be considered misclassified data samples.
The clustering results on the training and test data are visualized in Figure 19 and Figure 20, respectively.

In Figures 19 and 19, the red points indicate the center of the clusters.
Hyperparameter Space As seen in Table II, the best train and test performances are observed when the all-MiniLM-L6-v2 model is used. For this reason, we continue with this model for the hyperparameter search. The selection of the number of cluster is the hyperparameter that has the most impact on the clustering performance, so we conducted experiments with three different numbers of clusters and analysed the result.
The performance metrics obtained by the hyperparameter search with KMeans algorithm is given in Table III. As seen in Table III, the highest performance is achieved with the highest

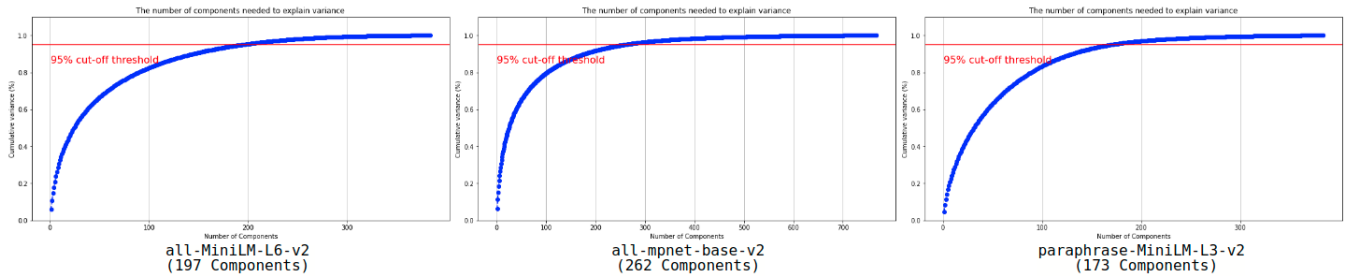


Fig. 16: Comparison of the selected number of components.

TABLE II: The performance metrics of the Kmeans algorithm.

	Train		Test	
(l)2-5	**Silhouette Score**	**Calinski_Harabasz Score**	**Silhouette Score**	**Calinski_Harabasz Score**
all-MiniLM-L6-v2	0.2328	3118.85	0.2013	1364.83
all-mpnet-base-v2	0.1865	1844.83	0.1534	818.93
paraphrase-MiniLM-L3-v2	0.2339	3060.83	0.1968	1333.13

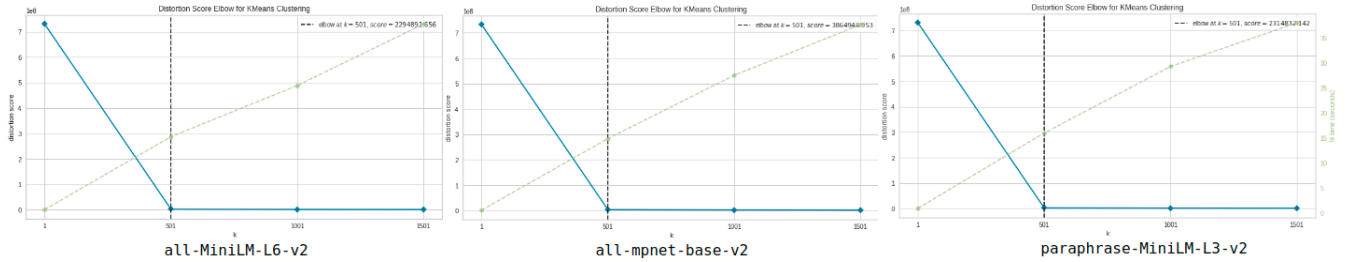


Fig. 17: The selection of the optimum number of clusters.

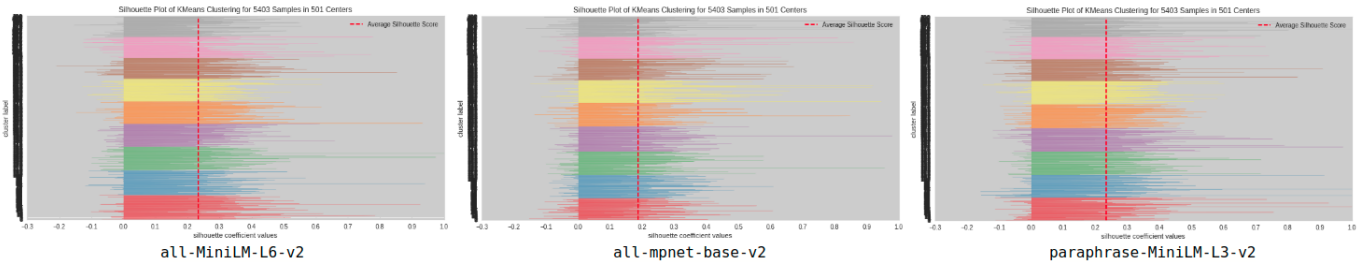


Fig. 18: The selection of the optimum number of clusters.

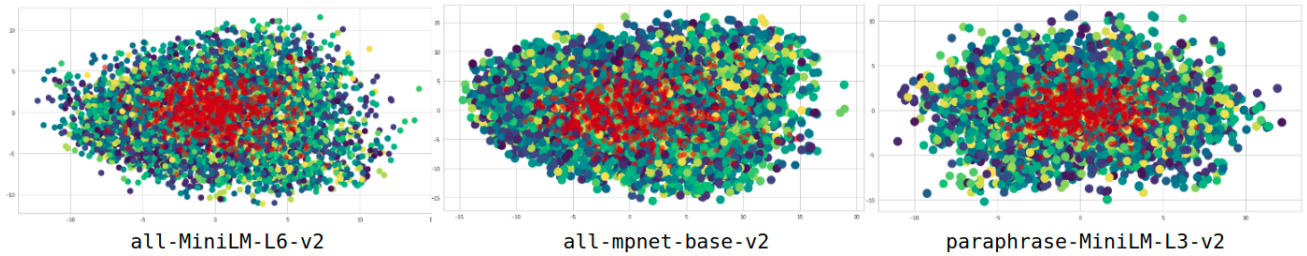


Fig. 19: The KMeans clustering results on the training data.

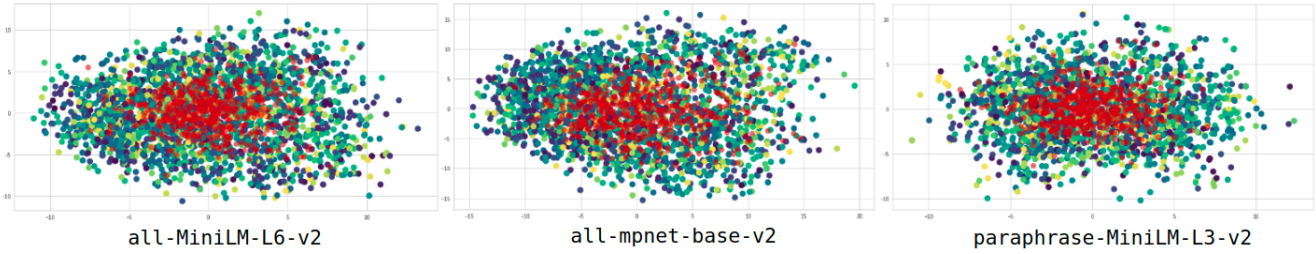


Fig. 20: The Kmeans clustering results on the test data.

number of clusters in the training phase. On the contrary, with the smallest number of clusters, the highest performance is achieved. With these results, it can be concluded that the higher number of clusters has a good impact on the clustering performance but the model starts to learn the data pattern by heart after a point and overfit on the training data. This can explain the poor performance result of the best model on the training phase. On the other hand, it can be observed that the model that has the lowest number of clusters does not learn the training data as other ones, whereas it reaches the highest performance on the test set, meaning that it does not overfit on the training data and has more capability of generalization compared to other ones.

Figure 21 visualizes the difference clustering results between the models with the lowest and highest number of clusters.

As seen in Figure 21, the 1500 clusters almost cover all the data compared to 250 clusters that formes a much more compact form in the training phase. On the other hand, the 1500 clusters distribute a wider area than the test data as it tries to cover the training data again in the test phase.

2) Spectral Clustering: For the spectral clustering, the Laplacian graphs are obtained for three pre-trained models as described in Section 2 and the eigendecomposition is performed.

Figure 22 shows the sorted eigenvalues for the features produced by each model.

The smaller eigenvalues (45, 45, and 42, respectively) than a specific threshold which is set to 0.1 are selected to project data which has produced by each model to a lower space. As explained in Section 3, the eigenvalues that are near zero is considered as the number of connected components in the graph, in other words, the number of clusters on the data. For this reason, the eigenvectors corresponding to these eigenvalues are used to transform the data into a lower dimensional space, and then the Kmeans algorithm is performed on the transformed data.

The initial parameters used in the KMeans algorithm are given in Table I. The number of clusters are chosen with the elbow method explained in the previous section.

The performance results of the three models are given in Table IV. The number of clusters is decided using the elbow method as seen in Figure 23 for the spectral clustering. As seen in Table IV, the paraphrase-MiniLM-L3-v2 model reached the highest test performance based on both evaluation metrics, so the hyperparameter search will be performed with this model in the next section.

A seen in Figures 17 and 23, the selected number of clusters are the same for KMeans and Spectral Clustering Algorithms but the distortion score reached by Spectral Clustering is dramatically lower than the one obtained by KMeans algorithm. As a result of this, the silhouette scores reached in the training phase are much more promising compared to ones that obtained by KMeans algorithm as seen in Figure 24.

The spectral clustering results on the training and test sets are visualized as in Figure 25

Hyperparameter Space. For the hyperpatameter search, paraphrase-MiniLM-L3-v2 model is used since the highest performance results achived by this model. The experiments were conducted considering different eigenvalue threshold used on the selection of components to project data into lower dimensional space using the Laplacian matrix. The number of components are changed with the threshold values of 0.1, 0.2 and 0.05. When 42 and 77 eigenvectors are selected, the performance metrics reaches similar values, whereas performance decreases a little with 21 eigenvectors. It can be concluded that 21 eigenvectors are unefficient to represent the data. On the other hand, it can not be said that increasing the number of selected eigenvectors always has a good impact on the performance since 77 eigenvectors shows lower performance that 42 eigenvectors.

3) Density-Based Spatial Clustering of Applications with Noise (DBSCAN): For DBSCAN we fine tuned parameters epsilon, min_samples and distance metric. For epsilon we used Nearest Neighbors and used the distance value at maximum curved point as seen at Figure-26 and for the distance metric ”euclidean” and ”manhattan” distances.

Hyperparameter Space Parameter we tired for GM as seen at Table-VI.

4) Gaussian Mixture: For Gaussian Mixture Model parameters we fine tuned for performance are n_components, max_iter, covariance_type, init_params. For n_components or number of cluster usually the value that minimizes the Akaike information criterion (AIC) or the Bayesian information criterion (BIC) generally used so we used that all and we also used elbow method to determine number of clusters, although we used this methods we also tried other values too,

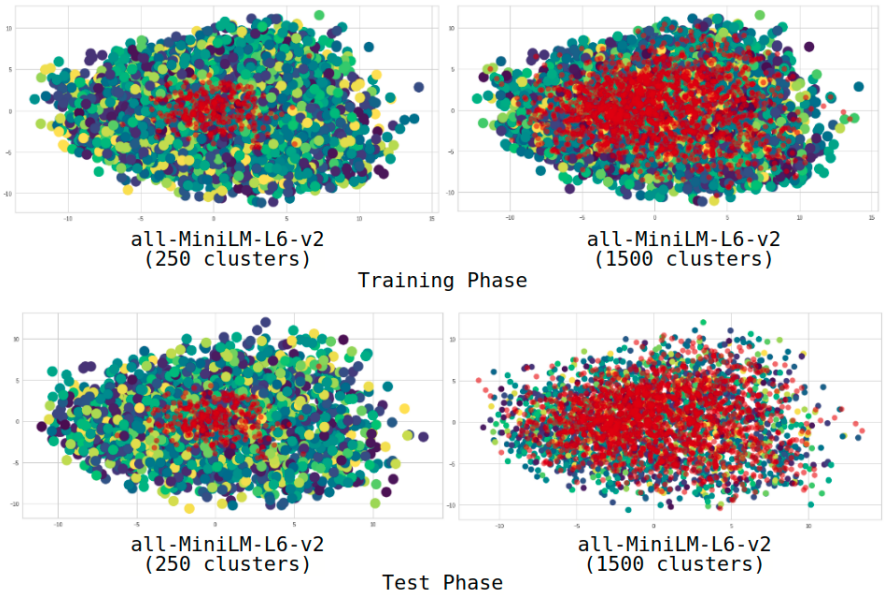


Fig. 21: The KMeans clustering results on the test data.

TABLE III: Hyperparameters search results of KMeans algorithm.

all-MiniLM-L6-v2	Train		Test	
(l)2-5 (r)1-1	**Silhouette Score**	**Calinski_Harabasz Score**	**Silhouette Score**	**Calinski_Harabasz Score**
n_cluster=501	0.2328	3118.85	0.2013	1364.83
n_cluster=250	0.2782	4347.19	0.2603	1868.86
n_cluster=1500	0.2926	1772.78	0.1814	930.00

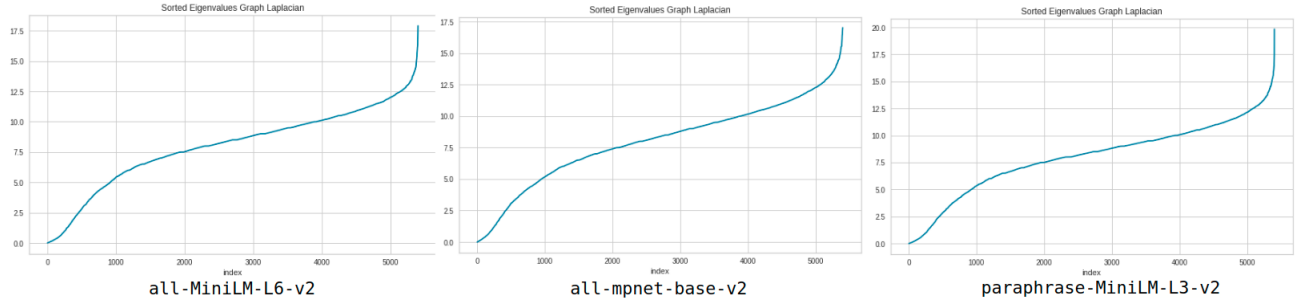


Fig. 22: The sorted eigenvalues for each model.

TABLE IV: The Spectral Clustering performance results of the models.

	Train		Test	
	Silhouette Score	**Calinski_Harabasz Score**	**Silhouette Score**	**Calinski_Harabasz Score**
all-MiniLM-L6-v2	0.21	2932.25	0.19	1349.76
all-mpnet-base-v2	0.17	1790.31	0.15	818.96
paraphrase-MiniLM-L3-v2	0.21	2933.95	0.20	1355.82

TABLE V: The hyperparameter search on spectral clustering.

paraphrase-MiniLM-L3-v2	Train		Test	
	Silhouette Score	**Calinski_Harabasz Score**	**Silhouette Score**	**Calinski_Harabasz Score**
eigenval thresh. = 0.1 42 eigenvecs	0.2188	2933	0.1980	1335.82
eigenval thresh. = 0.2 77 eigenvecs	0.2185	2913	0.1951	1324.23
eigenval thresh. = 0.05 21 eigenvecs	0.2150	2869	0.1823	918.50

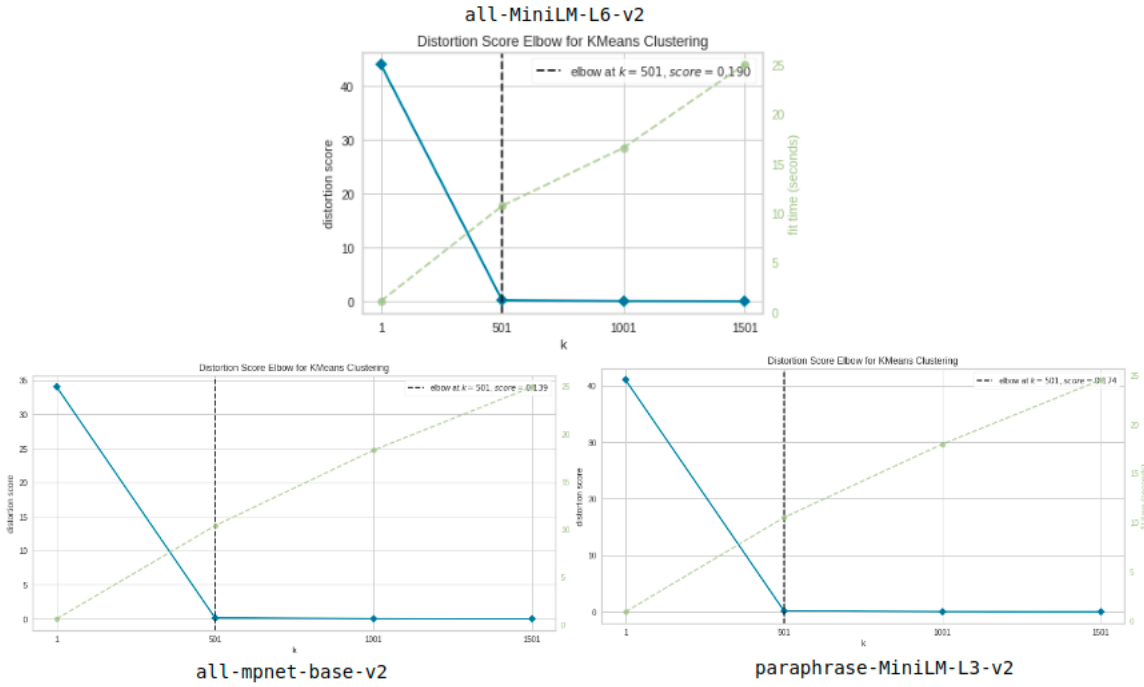


Fig. 23: The Elbow Method for spectral clustering.

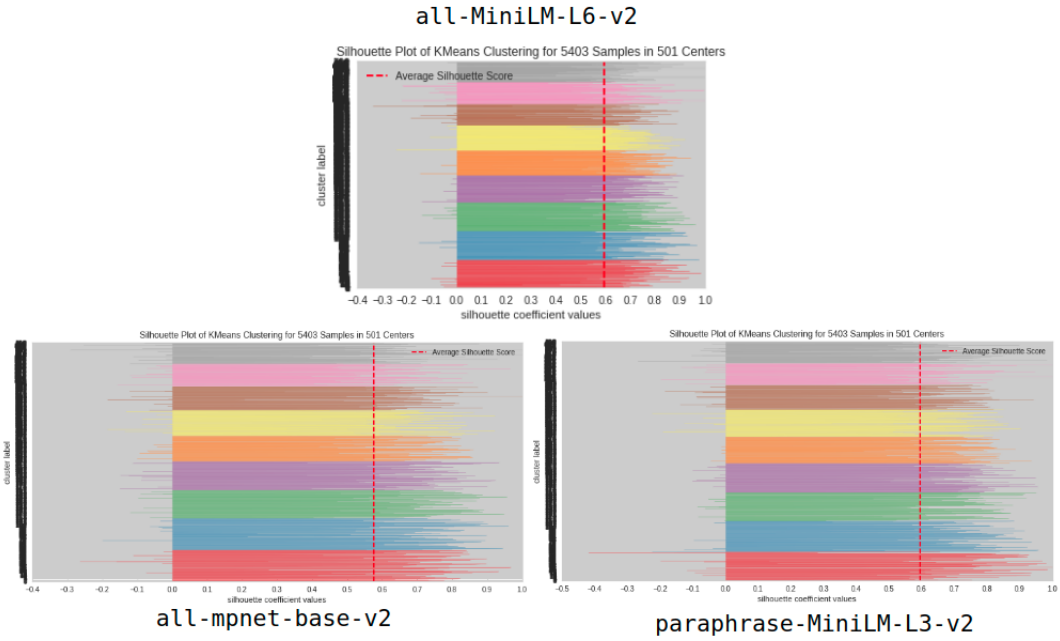


Fig. 24: The Silhouette scores reached by Spectral Clusering in the training phase.

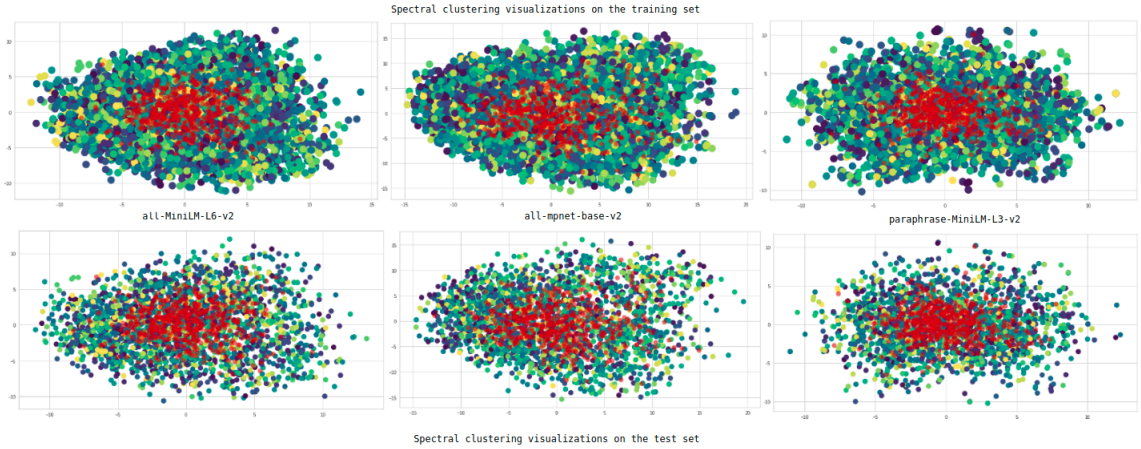


Fig. 25: The Spectral clustering visualizations in the training and test phase.

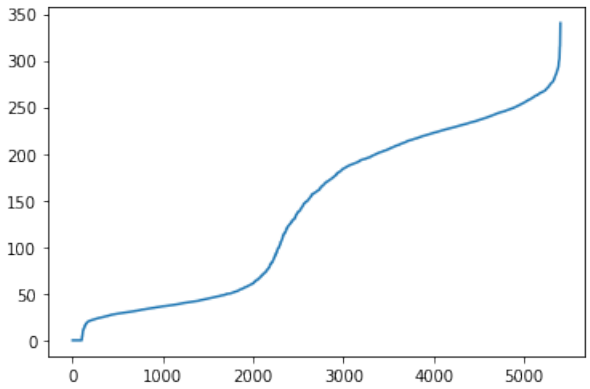


Fig. 26: Nearest Neighbor Graph for Data embedded with ”all_mini_l6v2” Model and Manhattan Distance as metric

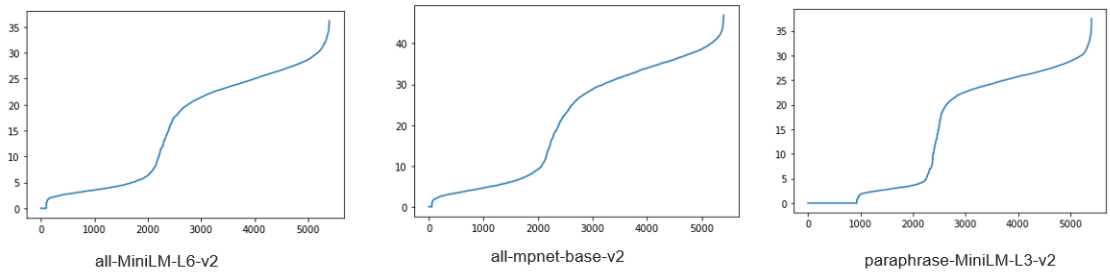


Fig. 27: Nearest Neighbor Graphs for to Determining The Epsilon Value for DBSCAN

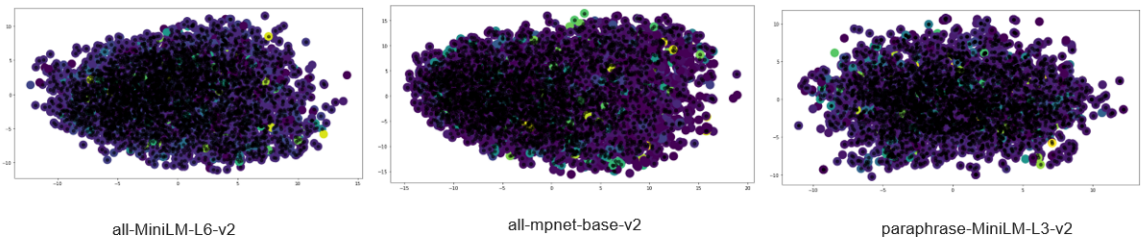


Fig. 28: The DBSCAN Clustering Result on Training Data

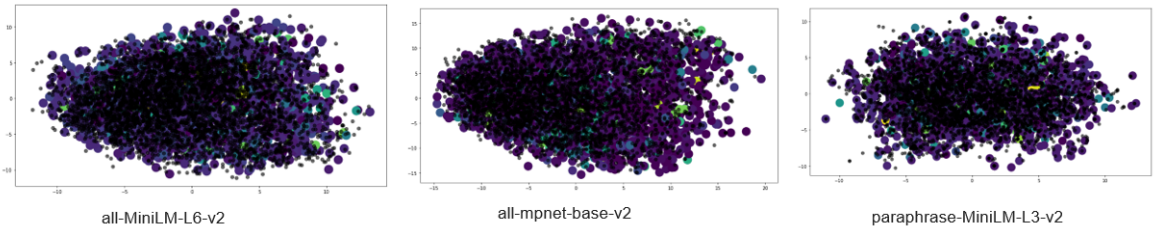


Fig. 29: The DBSCAN Clustering Result on Test Data

TABLE VI: Performance of DBSCAN with Different Parameters

Model	Train Silhouette Score	Train CH Index	Test Silhouette Score	Test CH Index	Number of Clusters
dbscan_30_5_all_mini_l6v2	-0.102	233.070	-0.143	71.811	131
dbscan_32_5_all_mini_l6v2	-0.387	275.378	-0.382	116.737	54
dbscan_28_5_all_mini_l6v2	-0.109	58.583	-0.221	18.229	229
dbscan_manhattan_280_5_all_mini_l6v2	-0.283	3.371	-0.048	0.4975	4
dbscan_38_5_all_mpnet_basev2	-0.219	103.030	-0.257	38.545	149
dbscan_40_5_all_mpnet_basev2	-0.476	127.097	-0.484	55.268	64
dbscan_42_5_all_mpnet_basev2	-0.519	17.226	-0.506	8.349	17
dbscan_37_5_all_mpnet_basev2	-0.191	63.119	-0.261	23.170	178
dbscan_manhattan_420_5_all_mpnet_basev2	-0.432	5.190	0.010	0.746	5
dbscan_30_5_all_minilm_l3v2	-0.131	196.845	-0.149	71.357	141
dbscan_32_5_all_minilm_l3v2	-0.405	302.476	-0.412	122.342	51
dbscan_28_5_all_minilm_l3v2	-0.110	44.765	-0.197	15.108	252
dbscan_manhattan_270_5_all_minilm_l3v2	-0.465	5.787	-0.036	0.037	8

for understanding and experimenting the algorithm. The best performed model is GM with all-MiniLM-L6-v2 as embedding model and we can see that around 50 is where they are both minimized at Figure-30 also we can clearly see at elbow method that 50 is the where elbow is at Figure-31, so we can say 50 is the best number of clusters for this model.

For max_iter parameter we tried 100 which is default value and 1000. As for covariance_type we used "full" and "spherical" and for the init_param parameter we tried 4 possible values in sklearn module.

Hyperparameter Space Parameter we tred for GM as seen at Table-VII. Models named as gm_<n_components>_<covariance_type>_<embedding _model>_<init_param> .

V. Proposed Method in this Study

The Gaussian Mixture Model with number of clusters as 50, $covariance\_type$ as "spherical" and max_iter parameter as 1000 with "all-MiniLM-L6-v2" with Silhouette Score for training data as 0.3369, Calinski-Harabasz Index for train set as 5799.079, Silhouette Score for test data as 0.3340 and Calinski-Harabasz Index for test set as 2506.747.

GMM clustering can accommodate clusters that have different sizes and correlation structures within them and they are better than other clustering methods we used at clustering data with more dimensions and more complex data.

VI. Clustering Analysis

In this project, we studied on paper clustering by their subjects with four well-known clustering algorithms and three pre-trained models in the literature, and evaluate our experiments based on two unsupervised clustering evaluation metrics. Google COLAB is used for the working environment, and built-in Python libraries and functions are used to perform the experiments. We tried two approached to design the dataset, we first took just paper abstract into account which achieves significantly poor performance results, then we took paper title, paper abstract, paper published conference, cited paper title, cited paper published conference, cited paper abstract and the conferences to cluster the papers by subject. The conferences categorized with numbers, whereas other features are embedded with selected pre-trained models and dimension reduction is applied to these embeddings. All the experiments and hyperparameter searches will be conducted with the latter approach which reaches much more higher performance results. The best model we obtained is Gaussian Mixture Model with number of clusters as 50, $covariance\_type$ as "spherical" and max_iter parameter as 1000 with "all-MiniLM-L6-v2" as pre-trained sentence transformer model.

Because of GMM requests number of cluster as a parameter we already know the number of clusters the model created which is 50. We already talk about how did we choose the number of clusters in the Experiments and Results section.

The number of papers in each clusters can be seen in Figure-34. The unique number of authors in each clusters can be seen in Figure-35

The word cloud for top 3 most frequently used words (the words that describe the cluster) in each cluster can be seen in Figure-36

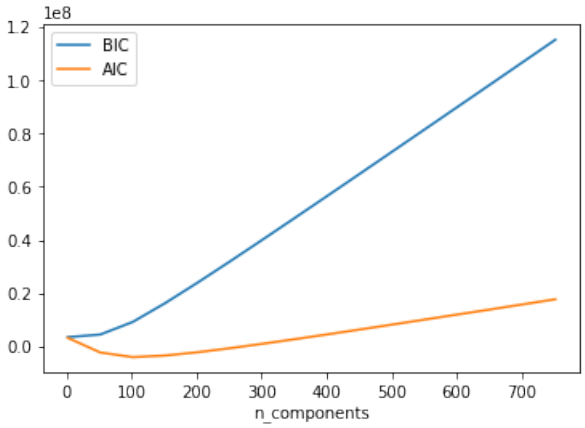


Fig. 30: AIC and BIC graph for GM with all-MiniLM-L6-v2 as Embedding Model

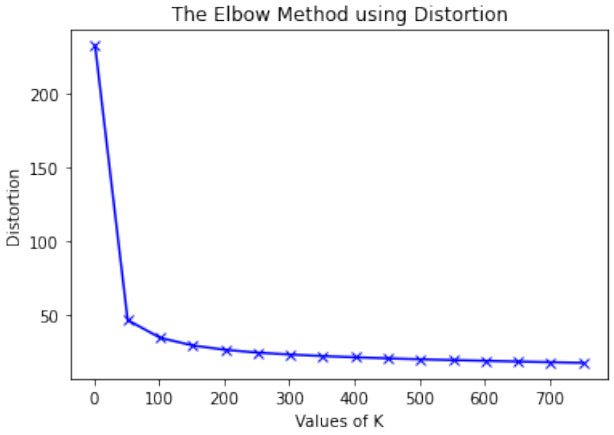


Fig. 31: Elbow Method for GM with all-MiniLM-L6-v2 as Embedding Model

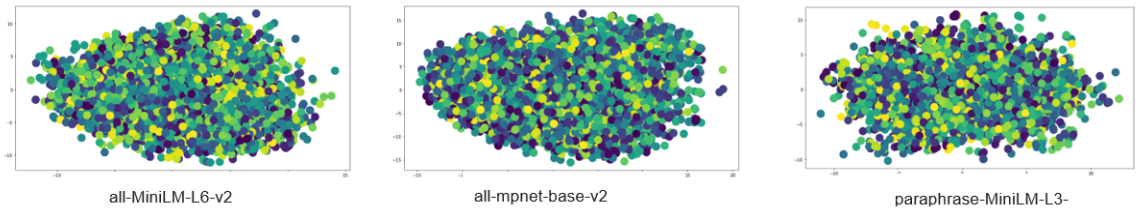


Fig. 32: The GM Clustering Results on The Train Data (n_components=50, covariance_type ='spherical', max_iter=1000)

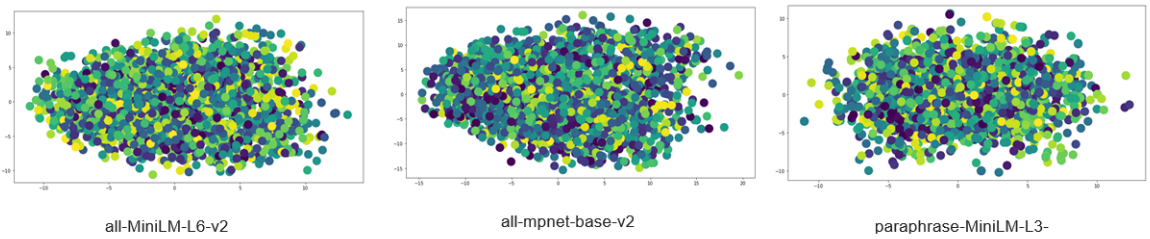


Fig. 33: The GM Clustering Results on The Test Data (n_components=50, covariance_type ='spherical', max_iter=1000)

TABLE VII: Performance of GM Model with Different Parameters

Model	Train Silhouette Score	Train CH Index	Test Silhouette Score	Test CH Index
gm_200_all_mini_l6v2	0.292	4663.259	-0.041	158.002
gm_200_all_mini_l6v2_random_init	-0.342	2.233	-0.689	1.926
gm_500_all_mini_l6v2_	0.226	3078.273	0.015	96.886
gm_350_all_mini_l6v2_	0.247	3702.734	0.0004	108.821
gm_400_all_mini_l6v2_	0.239	3463.310	0.0008	102.990
gm_150_all_mini_l6v2_	0.299	4962.012	-0.096	152.008
gm_100_all_mini_l6v2_	0.305	5310.301	-0.150	171.171
gm_50_all_mini_l6v2_	0.331	5109.234	-0.232	218.636
gm_50_spherical_all_mini_l6v2_	**0.3369**	**5799.079**	**0.3340**	**2506.747**
gm_150_spherical_all_mini_l6v2_	0.299	4982.179	0.288	2150.586
gm_50_spherical_all_mini_l6v2_kmeans++	0.324	5593.49	0.326	2454.34
gm_50_spherical_all_mini_l6v2_random_from_data	0.321	5266.629	0.318	2295.741
gm_50_spherical_all_mini_l6v2_random	0.300	4739.26	0.307	2122.95
gm_200_all_mpnet_basev2	0.230	3230.337	-0.014	169.193
gm_200_all_mpnet_basev2_random_init	-0.305	1.840	-0.666	1.822
gm_350_all_mpnet_basev2_	0.194	2325.130	-0.004	92.790
gm_150_all_mpnet_basev2	0.243	3644.811	-0.066	172.426
gm_50_spherical_all_mpnet_basev2	0.3036	4977.24	0.295	2147.131
gm_150_spherical_all_mpnet_basev2	0.243	3644.807	0.232	1576.922
gm_50_spherical_all_mpnet_basev2_kmeans++	0.300	4917.134	0.293	2116.78
gm_50_spherical_all_mpnet_basev2_random_from_data	0.292	4675.132	0.290	1436.570
gm_50_spherical_all_mpnet_basev2_random	0.259	4018.308	0.294	2038.740
gm_200_all_minilm_l3v2	0.239	1459.309	-0.415	58.062
gm_350_all_minilm_l3v2	0.231	2430.451	-0.367	39.985
gm_150_all_minilm_l3v2_	0.210	1593.498	-0.439	68.955
gm_50_spherical_all_minilm_l3v2	0.319	5448.83	0.317	2387.173
gm_150_spherical_all_minilm_l3v2	0.293	4895.325	0.278	2073.047
gm_50_all_minilm_l3v2_random_from_data	0.314	5229.071	0.311	2252.93
gm_50_spherical_all_minilm_l3v2_kmeans++	0.314	5321.14	0.309	2335.511
gm_50_spherical_all_minilm_l3v2_random	0.307	4412.901	0.317	1982.301

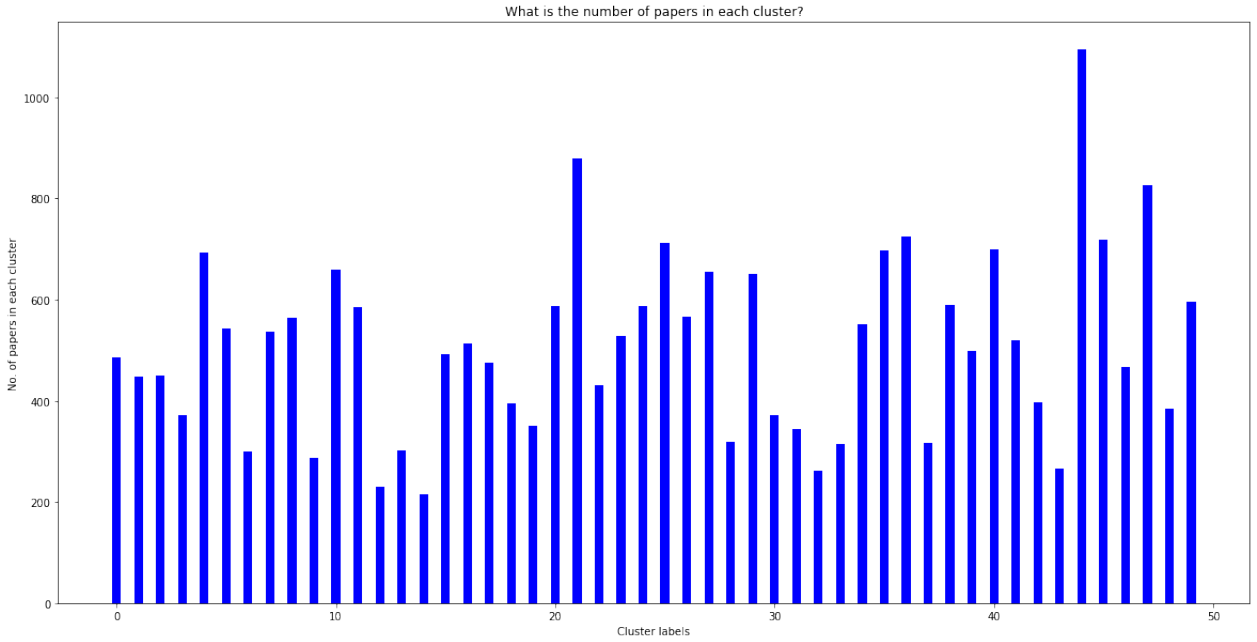


Fig. 34: No. of Papers in Each Cluster

VII. Clustering of Authors

For author clustering we used text data from paper clustering and combine the data after grouping by author names. After using PCA for dimensionality reduction we used Elbow method for number of clusters.

The shape of train data after PCA is (340, 139) and the shape of test data is (146, 139).

Google Colab Link of Author Clustering: https://drive.google.com/file/d/12Vg-4JzGyZ4AXavgJ3T38uUiuZABlYJi/view?usp=sharing

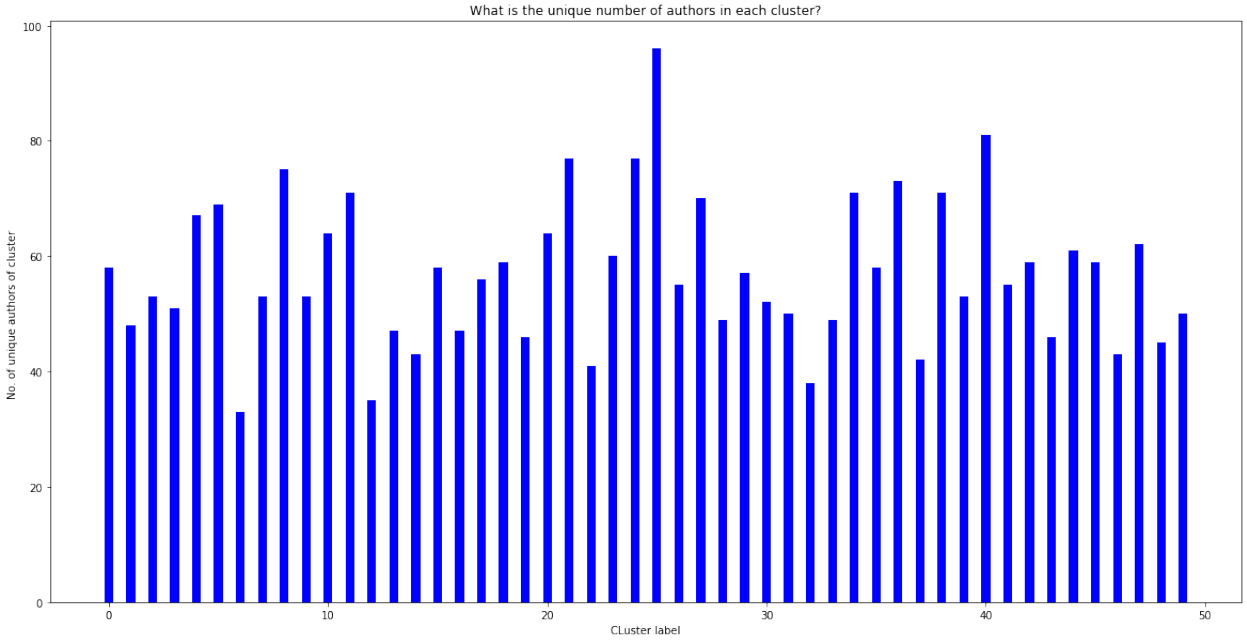


Fig. 35: Unique Number of Authors in Each Cluster

VIII. Future Works

For the standard clustering algorithms parameter optimization is really important, so for the future works we could work on parameter optimization such as different distance metrics for the DBSCAN algorithm as its performance depends on the quality of distance measure.
Also using deep learning as observed in the literature would probably increase the performance of the clustering as the data has many dimensions.

IX. Semantic Search

For sematic search we used ”msmarco-distilbert-base-v4” as pre-trained sentence transformer. The reason of using a large scale information retrieval corpus model rather than simple sentence embedding model is because with semantic search we are trying to find the closest paper to given text so we are trying to do asymmetric semantic search. For dataset we used ”paper_abstract” as corpus and ”paper_title” feature as text to find the closest paper and we used ”msmarco-distilbert-base-v4” for embedding. The step of the work we do as follows:

- Embedding the both corpus and queris for finding the closest paper
- Finding the most similar 5 paper for every query with cosine similarity
- Summing the paper and title matches for accuracy.

The embedding models performance was 0.698 and for improving the performance we tried scaling the data nad then PCA for dimensionality reduction. After reducing the dimeonsions the data shape changed form (2587, 768) to (2587, 296) and the performance of the semantic search is improved to 0.761.

Google Drive Link for Semantic Search Notebook: https://drive.google.com/file/d/1yelg5-QUDhQWAL4GsYELr3kTbRAdO6Et/view?usp=sharing

X. Conclusion

In conclusion, the highest clustering performance was achieved by GM with the Sihouette score of 0.3340 and 2506.747 of Calinski-Harabasz score. KMeans and Spectral Clustering Algorithms reach similar results. KMeans algorithm achieves better clustering performance than Spectral clustering since the spectral clustering algorithm stands out for the data that has a convex structure but the data used in this study has no clear convex shape but rather a complex one. Although DBSCAN is a more complex model and usually better than K-means at non-linearly separable clusters, because the quality of DBSCAN relies on the distance measure and the default distance metric is Euclidean metric or manhattan distance which is not so good with high dimensional data, DBSCAN performed worse than other clustering algorithms.
Since the GM algorithm calculates the probability that all points belong to all clusters, that is, considers clusters as a probability distribution, it can learn clusters with any elliptical shape, so its performance is higher than other clustering algorithms. And also as the central limit theorem states as the number of samples increases the data is more likely to resemble a Gaussian distribution so having over 7000 data is also increased the performance. The problem of GM is determining the number of clusters but we overcame this problem with both AIC-BIC and Elbow methods. The other problem is, like all other clustering algorithms, the number of dimensions is too much.

Fig. 36: Word Cloud for Top 3 Most Frequently Used Words

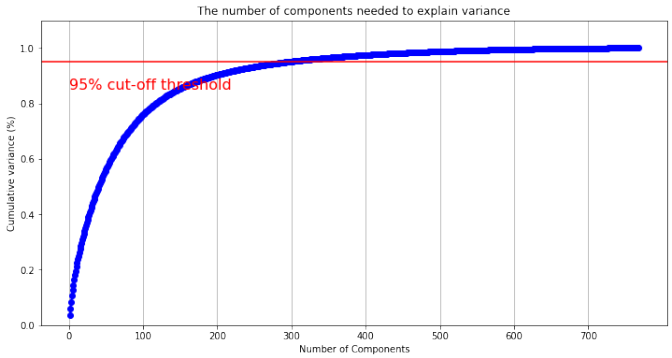


Fig. 37: PCA Grpah for Semantic Search Data

XI. Google Drive Link

Google Drive Link: https://drive.google.com/drive/folders/1UrktLs6KG4uG9j8hD04pVwDJEel_SOMR?usp=sharing
Google Colab Link: https://colab.research.google.com/drive/1yMtWRt8AH29S_k8S-DaskogoZYLgq0jq?usp=sharing

References

[1] R. G. Creţulescu, D. I. Morariu, M. Breazu, and D. Volovici, "Dbscan algorithm for document clustering," *International Journal of Advanced Statistics and ITC for Economics and Life Sciences*, vol. 9, no. 1, pp. 58–66, 2019. [Online]. Available: https://doi.org/10.2478/ijasitels-2019-0007

[2] M. Moradi Fard, T. Thonet, and E. Gaussier, "Deep k-means: Jointly clustering with k-means and learning representations," *Pattern Recognition Letters*, vol. 138, pp. 185–192, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865520302749

[3] D. Marutho, S. Hendra Handaka, E. Wijaya, and Muljono, "The determination of cluster number at k-mean using elbow method and purity evaluation on headline news," in *2018 International Seminar on Application for Technology of Information and Communication*, Sep. 2018, pp. 533–538.

[4] L. Duan, C. Aggarwal, S. Ma, and S. Sathe, "Improving spectral clustering with deep embedding and cluster estimation," in *2019 IEEE International Conference on Data Mining (ICDM)*, Nov 2019, pp. 170–179.

[5] "Pretrained models," https://www.sbert.net/docs/pretrained_models.html#pretrained-models, accessed: 2022-12-28.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[7] A. Moreira, M. Y. Santos, and S. Carneiro, "Dbscan snn density-based clustering algorithms – dbscan and snn," 0.

[8] I. Khater, I. Nabi, and G. Hamarneh, "A review of super-resolution single-molecule localization microscopy cluster analysis and quantification methods," *Patterns*, vol. 1, p. 100038, 06 2020.

[9] "Using dbscan with linfa-clustering," https://rust-ml.github.io/book/4_dbscan.html, accessed: 2022-12-25.

[10] "Clustering and mixture models," https://amueller.github.io/aml/03-unsupervised-learning/02-clustering-mixture-models.html, accessed: 2022-12-25.

[11] "Performance metrics in machine learning — part 3: Clustering," https://towardsdatascience.com/performance-metrics-in-machine-learning-part-3-clustering-d69550662dc6, accessed: 2022-12-28.

[12] "Silhouette coefficient," https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c#, accessed: 2022-12-28.