A Text Clustering Based DBSCAN Algorithm

Caner Kaya, Guowei Wang

Keywords: text mining, dbscan, document clustering, vectorization

Contents

1. Introduction

Text mining is the process of extracting valuable knowledge that is effective, useful, understandable, and distributed in text files, and using this knowledge to better organize the information. It targets studies such as text categorization, text clustering, concept/entity extraction, production of granular taxonomy, sentimental analysis, document summarization and entity relationship modelling[1].

Text mining technology is now broadly applied to a wide variety of government, research, and business needs. All these groups may use text mining for records management and searching documents relevant to their daily activities. Governments and military groups use text mining for national security and intelligence purposes. Scientific researchers incorporate text mining approaches into efforts to organize large sets of text data (i.e., addressing the problem of unstructured data), to determine ideas communicated through text (e.g., sentiment analysis in social media) and to support scientific discovery i3n fields such as the life sciences and bioinformatics. In business, applications are used to support competitive intelligence and automated ad placement, among numerous other activities.

The application of cluster analysis to text-based documents is known as text clustering. Clustering algorithms are of great importance in the field of data mining. However, the choice of clustering algorithms depends on both the current data type and the purpose and application[2]. There are various types of clustering algorithms; the top 5 ones are partition-based algorithms, hierarchy-based algorithms, fuzzy theory-based algorithms, distribution-based algorithms, and density-based algorithms. Partition-based algorithms divide data into non-hierarchical clusters and consider the centers in the dataset as the centers of the clusters. The most popular example of this type of cluster is the K-means algorithm, and another examples are CLARA, PAM, and CLARANS [3]. Density-based algorithms are based on the density of the data region at any given location. In this algorithm, clusters can take any irregular shape unlike

K-Means and work well for noisy datasets[4]. However, these algorithms have trouble dealing with data of varying densities or high dimensions[5]. DBSCAN algorithm is the most popular of density based clusters and was also used in this project.

2. General Overview

Important component of the data mining system is a data source, data mining engine, server, data warehouse model evaluation module, graphical user interface and the knowledge base. Figure 1 shows a general overview of the system.

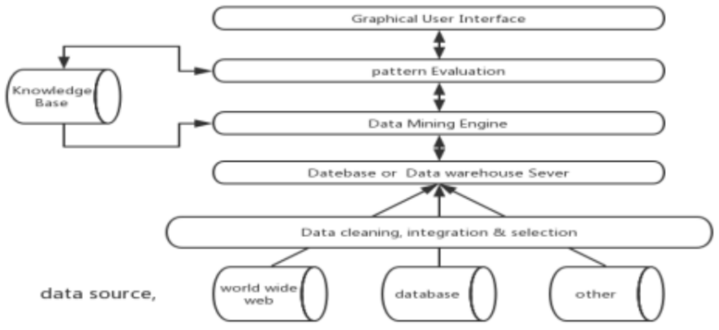


Figure 1: General Overview

2.1. Data Sources

The actual sources of data are databases, data warehouses, the World Wide Web, text files, and other documents. Organizations typically store data in databases or data warehouses. A data warehouse can include one or more databases, text files, spreadsheets, or other data repositories. Sometimes, even plain text files or spreadsheets may contain information. Another major source of data is the World Wide Web or the Internet.

2.2. Cleaning Data

Before data is passed to the database or data warehouse server, it must be cleared, data integration and selection. Due to the information from various sources and different formats, therefore cannot be directly applied to the data mining process, because the data may not be complete and not accurate. Therefore, the data needs to be cleared and unified. From all kinds of data will be collected more than the required information, and only need to select the data which we are most interested in and pass it to the server. Database or data warehouse server: The database or data warehouse server consists of ready to deal with the data.

2.3. Data mining engine

Data mining engine is the main component of any data mining system. It contains several for manipulating data mining task modules, including connection, characteristic, classification, clustering, forecasting, time series analysis, etc. Data mining is the foundation of our data mining system structure. It includes from various data sources and data stored in the data warehouse in acquiring knowledge and knowledge of tools and software.

2.4. Pattern evaluation module

Pattern evaluation module is mainly responsible for the use of thresholds to investigate patterns. It uses data mining engine collaboration to find the most interesting parts.

2.5. The graphical user interface

A graphical user interface (GUI) module in the communication between system and user data mining. The module can easily and efficiently help users using the system, without having to understand the complexity of the process. When the user specifies queries or tasks and displays the results, the module and the data mining system.

2.6. Knowledge base

Knowledge base is very helpful to the whole process of data mining. Knowledge base and even may contain user view from the user experience and data, this may help in the data mining process. Data mining engines can receive input from the knowledge base, in order to make the result more accurate and reliable. Interaction pattern evaluation module on a regular basis and knowledge base to get input, and carries on the updates.

3. Methodology

This section describes the workflow of the proposed system. As seen in figure 2, it basically consists of data splitting, data preprocessing and clustering steps.

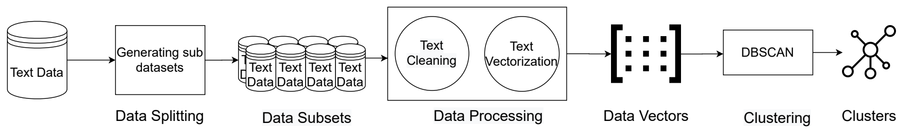


Figure 2: The workflow of the proposed system

3.1. Dataset

With the increase of data on the Internet, the subject of clustering documents from each other is a popular issue. Document clustering can be examined in 2 categories as online and offline. While online applications have limited solutions due to efficiency problems, offline applications may contain more flexible solutions. Therefore, the publicly available BBC news dataset[6], which was previously used in publication[7], is used for clustering .

The BBC dataset consists of 2225 news published in 5 different categories between 2004-2005. The distribution of the categories of news in the dataset is shown in figure 3. As can be seen in the figure, the distribution of the categories is not equal, there is approximately 5% difference between the numbers of the most found sports category and the least present entertainment category in the data set. Although unlabeled data are used while clustering, labeled news is not a problem. Labels can be discarded in the preprocessing phase.

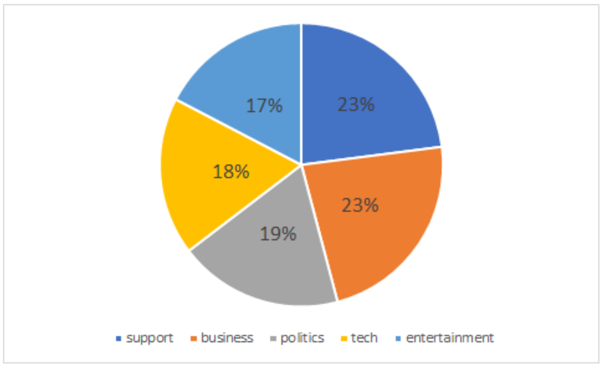


Figure 3: The distribuiton of the number of categories

3.2. Data Splitting

The data were shuffled and divided into subsets of 300 documents. In total,
100 7 subsets were obtained. 425 documents remained for the last subset. You can see the category distribution of subsets in figure 3.

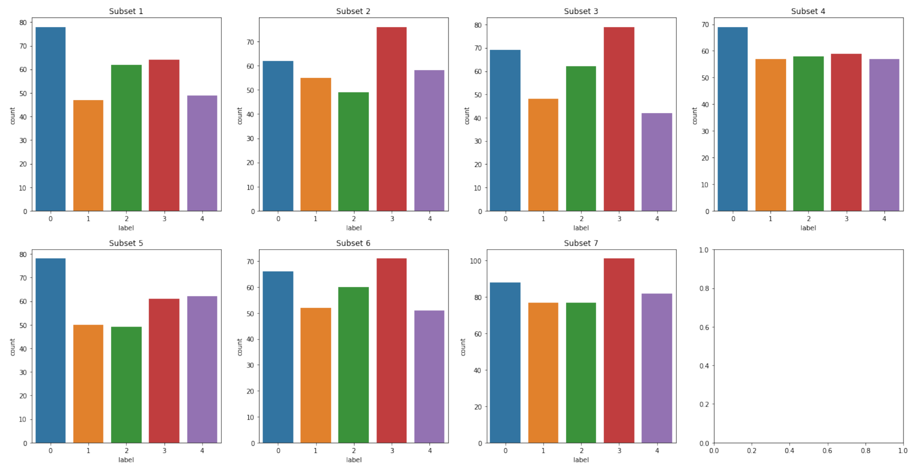


Figure 4: The category distribution of the subsets

3.3. Data Processing

The data processing step is where data is brought into the proper form before it is given to the algorithm. The data goes through the text cleaning and text vectorization stages. As a result of these operations, each document is expressed with a vector and all documents with a matrix.

3.3.1. Text Cleaning

As can be understood from the name of the text cleaning stage, it is the removal of unwanted characters, addings, punctuations and words in the text that may mislead the algorithm. If the text cleaning step is applied correctly, it has a very positive effect on the success of the algorithm and time consumption [8].

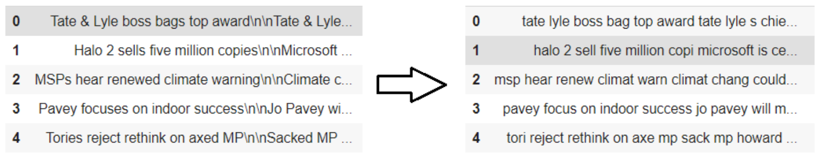
Stop words: Stop words are common terms that convey little information or, in some situations, introduce unneeded noise in particular computer processes, and hence must be deleted.

Stemming: The practice of reducing words to their fundamental form is known as stemming. For example, the words ”snow,” ”snowy,” and ”snowing” all have pretty similar meaning. These will be reduced to the root form of ”snow” by the stemming process. This is yet another approach to minimize noise and data complexity.

Lowercase: In text classification and clustering problems, whether the letters are uppercase or lowercase does not carry any important information. Therefore, making all letters lowercase will simplify the data.

Punctuations: Punctuation adds noise to a sentence that brings ambiguity when clustering text. These also need to be cleaned.

The text sample obtained after applying the above data cleaning methods can be seen in Figure 5.



0	Tate & Lyle boss bags top award\n\nTate & Lyle...
1	Halo 2 sells five million copies\n\nMicrosoft ...
2	MSPs hear renewed climate warning\n\nClimate c...
3	Pavey focuses on indoor success\n\nJo Pavey wi...
4	Tories reject rethink on axed MP\n\nSacked MP ...

0	tate lyle boss bag top award tate lyle s chie...
1	halo 2 sell five million copi microsoft is ce...
2	msp hear renew climat warn climat chang could...
3	pavey focus on indoor success jo pavey will m...
4	tori reject rethink on axe mp sack mp howard ...

Figure 5: The Example Data After Cleaning Step

3.3.2. Text Vectorization

Since clustering algorithms operate on a numeric feature space, we should first convert our documents into vector representations before performing clustering on them. There are several methods of text vectorization such as TF-IDF, WordtoVec, Glove embedding. In this project, TF-IDF, which is a simple method, was used first.

TF-IDF: Term frequency–inverse document frequency is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus**??**. Formula is as follows:

$$tfidf_{t,d} = tf_{t,d} \times idf_{t,d}$$

Figure 6: TF-IDF

The tf–idf is the product of two statistics, term frequency (tft,d) and inverse document frequency (idft,d). There are various ways for determining the exact values of both statistics. Words are represented by t, documents by d and corpus by D. The higher this value, the more important the word is to the text. Term frequency, tf(t,d), is the frequency of term t, formula is as follows:

$$tf_{t,d} = \frac{f_{t,d}}{\sum_{k \in d} f_{k,d}}$$

Figure 7: Term frequency

Where, the numerator is the number of Term t appearing in the text , and the denominator is the number of all Term in the text.The result of calculation is the word frequency of a particular Term. The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$idf_{t,D} = \log\frac{N}{1 + |\{d \in D: t \in d\}|}$$

Figure 8: Idftd

After applying the tf-idf vectorization, the dimensions of the subset are shown in table 1.

subset no	**dimension**
1	300 x 8324
2	300 x 11533
3	300 x 11213
4	300 x 12085
5	300 x 11304
6	300 x 11539
7	425 x 13398

Table 1: subset's dimensions

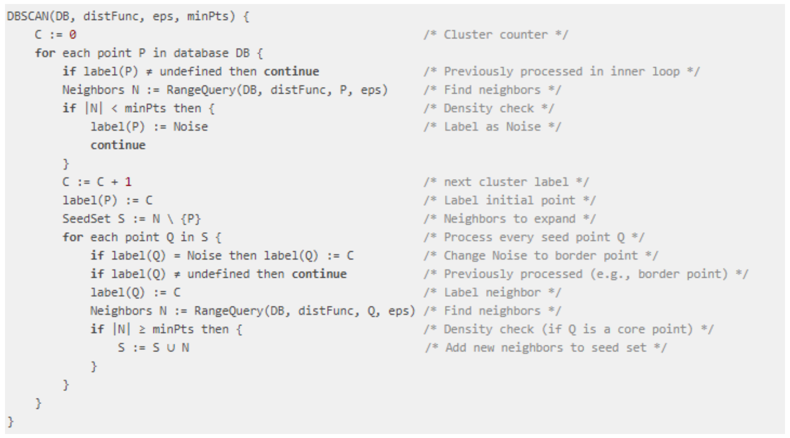
3.4. DBSCAN Algorithm

The characteristics of text mining objects are massive and heterogeneous. There are two basic types of clustering algorithms partitioning and hierarchical algorithms. The partitioning text clustering method can only recognize spherical clusters, and hierarchical algorithms need to compare the similarity between all the clusters globally, so the running speed is slow. None of them are suitable for mining large amounts of text.

DBSCAN (Density-based spatial clustering of applications with noise) relying on a density-based notion of clusters[9]. It defines a density-reachable(Definition 3) equivalence relation on the data set(it likes clustering in partitioning algorithms) and density-connected(Definition 4). Because of the above definition, DBSCAN can quickly find arbitrarily-shaped clusters. There is huge information resources on the Internet. Text mining has a wide range of application scenarios. DBSCAN algorithm is suitable for processing huge amounts of heterogeneous text, the clustering speed and the effect is better than the speed of the traditional clustering methods and effects.

Noise: A data point that is not a core point or a border point is considered noise or an outlier.

Core Point: A data point is considered to be a core point if it has minimum number of neighboring data points (min_samples) at an epsilon distance from it. This minimum number of data points includes the original data point.



```
DBSCAN(DB, distFunc, eps, minPts) {
    C := 0                                                /* Cluster counter */
    for each point P in database DB {
        if label(P) ≠ undefined then continue             /* Previously processed in inner loop */
        Neighbors N := RangeQuery(DB, distFunc, P, eps)   /* Find neighbors */
        if |N| < minPts then {                            /* Density check */
            label(P) := Noise                             /* Label as Noise */
            continue
        }
        C := C + 1                                        /* next cluster label */
        label(P) := C                                     /* Label initial point */
        SeedSet S := N \ {P}                              /* Neighbors to expand */
        for each point Q in S {                           /* Process every seed point Q */
            if label(Q) = Noise then label(Q) := C        /* Change Noise to border point */
            if label(Q) ≠ undefined then continue         /* Previously processed (e.g., border point) */
            label(Q) := C                                 /* Label neighbor */
            Neighbors N := RangeQuery(DB, distFunc, Q, eps) /* Find neighbors */
            if |N| ≥ minPts then {                        /* Density check (if Q is a core point) */
                S := S ∪ N                                /* Add new neighbors to seed set */
            }
        }
    }
}
```

Figure 9: DBSCAN algorithm pseudo code

Border Point: A data point that has less than minimum number of data points needed but has at least one core point in the neighborhood. Algorithm Steps: 1. Decide the value of eps and minPts. 2. For each point: Calculate its distance from all other points. If the distance is less than or equal to eps then mark that point as a neighbor of x. If the point gets a neighboring count greater than or equal to minPts, then mark it as a core point or visited. 3. For each core point, if it not already assigned to a cluster than create a new cluster. Recursively find all its neighboring points and assign them the same cluster as the core point. 4. Continue these steps until all the unvisited points are covered [10].

4. Evaluation

In this part, the bbc data set was clustered using the dbscan algorithm, and the data were visualized and success scores were obtained for the interpretation of the results.

4.1. Parameter Determination

As explained in section 3.4, DBSCAN algorithm requires 2 parameters as input, eps (the radius of the cluster) and minPts (the minimum data objects required inside the cluster). Deciding these parameters is one of the challenges of the dbscan algorithm. There are various studies for parameter determination,in these researches [11], [12] k-dist lists were used. Another method used to determine parameters is grid based technique [13]. In this study, the parameters were determined by using the k-nearest neighbourhoud distance and gridseach together. Optimization target was determined according to criteria such as minimum - maximum number of clusters and outlier numbers. In figur 10 you can see the k-distance graph of the cluster number 1. The points of the curve before the horizontal course are ideal epsilon parameter candidates. Next, with the help of grid search, set the parameters k and epsilon parameters.

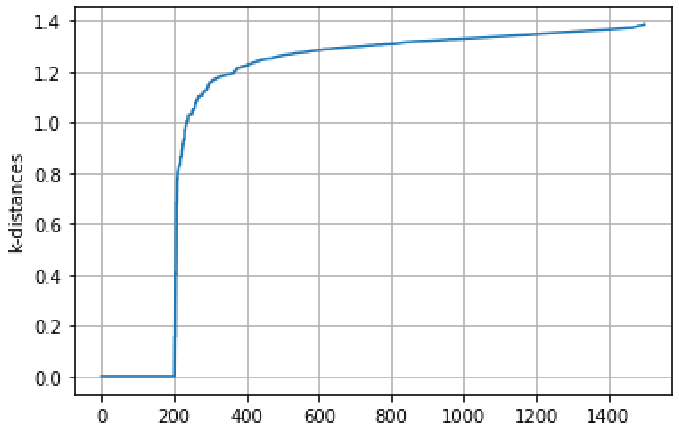


Figure 10: K distance example graph

Table 2 shows the determined parameters.

subset no	**Eps**	**MinPts**
1	1.33	10
2	1.33	9
3	1.31	7
4	1.31	5
5	1.31	5
6	1.31	6
7	1.31	6

Table 2: Determined Parameters

4.2. Data Visualization

Visualizing high dimensional data is a difficult task, but there are various techniques that reduce the data dimension. PCA(Principal Component Analysis) is a useful statistical technique used in the fields of recognition, classification, image compression. It is a technique whose main purpose is to keep the data set with the highest variance in high dimensional data, but to provide dimension reduction while doing this. By finding the general features in the multi-dimensional data, it enables the reduction of the number of dimensions and the compression of the data. Certain features will be lost with size reduction; but the intention is that these lost traits contain little information about the population. This method combines highly correlated variables to create a smaller set of artificial variables called “principal components” that make up the most variation in the data.

After the PCA is applied, the clustering plot of the subset is shown in figure 11.

4.3. Metrics

Unlike supervised learning algorithms, evaluating the performance of a clustering algorithm is not as simple as counting the number of errors, precision, and recall.Clusters are evaluated according to some measure of similarity or

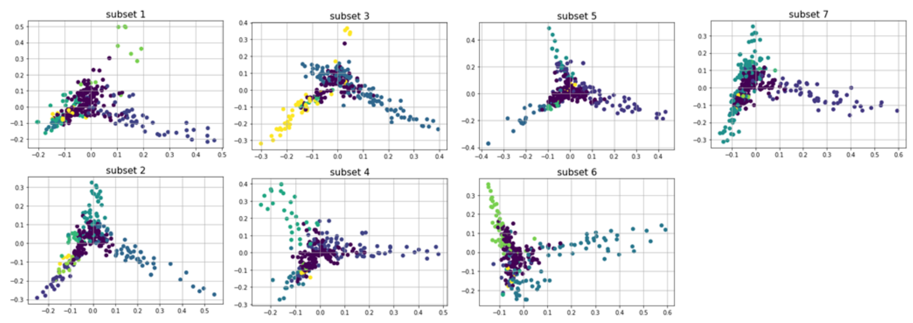


Figure 11: Cluster Graphs of the subsets

dissimilarity, such as the distance between cluster points. If the clustering algorithm successfully separates dissimilar observations from similar observations, it has done its task. The following metrics were applied during the evaluation, the sklearn.metrics[14] package was used in the calculations.

4.3.1. Rand index

Rand index need to be given category information C, assume that K is clustering results, A is the number of the same element in both C and K, B is the number of the different elements in both C and K, the denominator is the total number of pairs, Rand index as:

$$R = \frac{A+B}{C_2^n}$$

R values range for [0,1], the larger the value means clustering results and the real situation the anastomosis. For random results, R is no guarantee that the score is close to zero. In order to achieve ”in the case of a randomly generated clustering results, the indicators should be close to zero”, adjust the rand is taught (Adjusted rand index) have been proposed, it has a higher degree of differentiation

Given a set S of n elements, and two groupings or partitions (e.g. clusterings) of these elements, namely X=X1,X2,...,Xr and Y=Y1,Y2....,Ys, the overlap between X and Y can be summarized in a contingency table [nij] where each entry [nij] denotes the number of objects in common between Xi and Yj: nij= —XiYj—,The original Adjusted Rand Index using the Permutation Model is,

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}$$

where nij ,ai,bj are values from the contingency table .ARI values range for [-1,1], the larger the value means clustering results and the real situation the anastomosis.

4.3.2. Mutual Information based scores

Mutual Information is used to measure both of the data distribution. Suppose U and V is the distribution of N samples label, is the distribution of the two kinds of entropy entropy represents uncertainty degree) are:

$$H(U) = \sum_{i=1}^{|U|} P(i)log(p(i))$$

$$H(V) = \sum_{j=1}^{|V|} P'(j)log(p'(j))$$

where P(i)=—Ui—/N is the probability that an object picked at random from U falls into class Ui. With P(j)=—Vj—/N. The mutual information (MI) between U and V is calculated by:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)log\left(\frac{P(i,j)}{p(i)P'(j)}\right)$$

adjusting information is defined as

$$AMI = \frac{MI-E|MI|}{max(H(U),H(V))-E|MI|}$$

Using the method based on the information to scale poly international bucket for effect of snow to word. MI and NMI value range of [0, 1]. AM1 value range of 1-1, 1] ,The higher their values are, the more consistent the clustering results are with the real situation.

4.3.3. Homogeneity score

This score can be used to see if the clustering algorithm meets a key requirement: a cluster should only contain samples from a single class[15]. It's defined as follows:

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

It has a range of 0 to 1, with lower values indicating lower homogeneity. In fact, when There's uncertainty is reduced by Y_pred knowledge, $H(True|Y\_pred)$ shrinks ($h \to 1$), and vice versa.

4.3.4. Silhouette score

The silhouette coefficient is a cluster validity measure. Sil(i) = (b(i) − a(i)) / max(b(i) , a(i)) where a(i) − average dissimilarity between i and the other objects of the cluster to which I belongs. D(i,C) − For all other clusters C, average dissimilarity of i to all observations of C. b(i) - minimum of D(i,C) − dissimilarity between I and the neighbor cluster C. If the Sil(i) is nearer to one the object is placed in the correct cluster. If the Sil(i) is negative the object is placed in the wrong cluster. If it is around 0 the object is between the clusters[16].

4.3.5. Calinski-Harabasz İndex

The Calinski-Harabasz index [17] also known as the Variance Ratio Criterion, is the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters, the higher the score , the better the performances. For a set of data E of size nE which has been clustered into K clusters, the Calinski-Harabasz score S is defined as the ratio of the between-clusters dispersion mean and the within-cluster dispersion:

$$S = \frac{tr(B_k)}{tr(W_k)} \times \frac{n_E - k}{k-1}$$

Where tr(Bk)is trace of the between group dispersion matrix and tr(Wk) is the trace of the within-cluster dispersion matrix defined by:

$$W_k = \sum_{q-1}^{k} \sum_{x \in c_q} (x - c_q)(x - c_q)^T$$

$$B_k = \sum_{q-1}^{k} n_q (c_q - c_k)(c_q - c_k)^T$$

with Cq the set of points in cluster q,Cq the center of cluster q,CE the center of E and nq the number of points in cluster q. Advantages: The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster. The score is fast to compute. Drawbacks: The Calinski-Harabasz index is generally higher for convex clusters than other concepts of clusters, such as density based clusters like those obtained through DBSCAN.

Table 3 shows the Rand index, Mutual Information based scores, Homogeneity scores, Silhouette Coefficient and Calinski Harabasz Index of the DBSCAN algorithm.

subset	**cluster size**	**Rnd Ind**	**AMI**	**Homogenity**	**silhouette**	**CH index**
1	8	0.1231	0.3159	0.3399	0.0092	2.2481
2	4	0.0984	0.2438	0.2158	0.0048	2.3123
3	6	0.0771	0.2196	0.215	0.0067	2.1018
4	6	0.1077	0.2434	0.2312	0.0046	2.0691
5	7	0.1161	0.3103	0.3084	0.0084	2.2852
6	9	0.167	0.3043	0.3286	0.0093	2.128
7	7	0.1665	0.301	0.2979	0.0069	2.4924
Average	6.7143	0.1223	0.2769	0.2767	0.0071	2.2338

Table 3: Evaluation scores of the DBSCAN algorithm

5. Discussion

In phase 1 we implemented DBSCAN algorithm and visualization. In reading data step, we split data into groups, 300 items per group. This is because the DBSCAN algorithm does not good at handle high dimensions data, as the data gets smaller, the dimension of the vector gets smaller, this will help improve the efficiency of the algorithm. Actually, we didn't split data at first, then the result is that all the points come together. In cleaning data step, we remove new line, punctuation and stemming and all letters to lowercase, in order to decrease the dimensions of the vector of the data. For visualization, we vectorize text using TF-IDF, that is also necessary for DBSCAN. Then reduce all vectors to 2 dimensions by PCA method, that is necessary to show data points in the plane.

6. HDBSCAN

Describe the new DBSCAN algorithm selected for Phase 2 HDBSCAN* – Hierarchical Density-Based Spatial Clustering of Applications with Noise[18]. Performs DBSCAN over varying epsilon values and integrates the result to find a clustering that gives the best stability over epsilon. This allows HDBSCAN* to find clusters of varying densities (unlike DBSCAN), and be more robust to parameter selection.

Among density based clustering techniques DBSCAN is attractive in that it is efficient and is robust to the presence of noise within data. Its primary difficulties include parameter selection and the handling of variable density clusters. proposing the HDBSCAN* algorithm which addresses both of these problems, HDBSCAN* resolves many of the difficulties in parameter selection by requiring only a small set of intuitive and fairly robust parameters, but its major difficulty is that it sacrifices performance to do so.

The specific process of HDBSCAN* algorithm is divided into the following steps:

1. Transform the space

Transform the space is that we express the distance between two sample points by mutual reachability distance. In this way, the sample distance in the dense area will not be affected, while the distance between the sample points in the sparse area and other sample points will be amplified. This increases the robustness of the clustering algorithm against scattered points. Core distance: The distance between the sample and the KTH nearest neighbor sample point is called core distance and expressed as:

$$core_k(x) = d(x, N^k(x))$$

Mutual reachability distance: The mutual distance between two sample points is defined as:

$$d_m reachk(a,b) = maxcore_k(a), core_k(b), d(a,b)$$

2.Build the minimum spanning tree The data can be regarded as a weighted graph, in which the data points are the vertices, and the weight of the edges between any two points is the mutual reachability distance between these points. The graph is split, and the final graph changes as follows: from the complete graph to the minimal connected subgraph

3.Build the cluster hierarchy

Step 1: Sort all edges of the minimum spanning tree by weight

Step 2: Then select each edge in turn and merge the two subgraphs linked by the edge.

4. The tree condensing

The first step of cluster extraction is to compress the large and complex cluster tree into a smaller tree. The essence of this is to get rid of the scatter. Minimal family size: The minimum number of samples in each family. We use this as a parameter to HDBSCAN.

Step 1: Determine the minimum family size

Step 2: After the minimum cluster size is determined, we can traverse the cluster tree from root, and when each node is split, we can see whether the sample number of the two sample subsets generated by splitting is greater than the minimum family size

5.Extract the clusters

After the compression operation of the cluster tree, there are no scattered points in the tree. Now task is just to merge the nodes that are relatively close to each other into a family, so that the cluster we finally choose can have better stability Explicitly we have an empirical density

$$f(x_j) = 1/\varepsilon X_j$$

where Xj is the value at which we recorded the point Xj leaving the tree.

HDBSCAN defined max,Ci (Xj) to be the value at which the point Xj falls out of the cluster Ci (either as an individual point, or as a cluster split in the condensed tree). Similarly it defined min,Ci (Xj) as the minimum lambda value for which Xj is present in Ci. Then the stability of the cluster Ci is defined as

$$\sigma(Ci) = \sum_{X_j \in C_i} \lambda_m ax, C_i(X_j) - \lambda_m in, C_i(X_j))$$

Step 1: Initialize the family to select each leaf node of the compressed cluster tree as a cluster.

Step 2: Now work up through the tree (the reverse topological sort order), and perform the following operations at each step:

If the sum of the stabilities of the child clusters is greater than the stability of the cluster, then we set the cluster stability to be the sum of the child stabilities. If, on the other hand, the cluster's stability is greater than the sum of its children then we declare the cluster to be a selected cluster and unselect all its descendants. Once we reach the root node we call the current set of selected clusters our flat clustering and return that.

6.1. Evaluation

The clustering plot of the subset is shown in Figure 12.

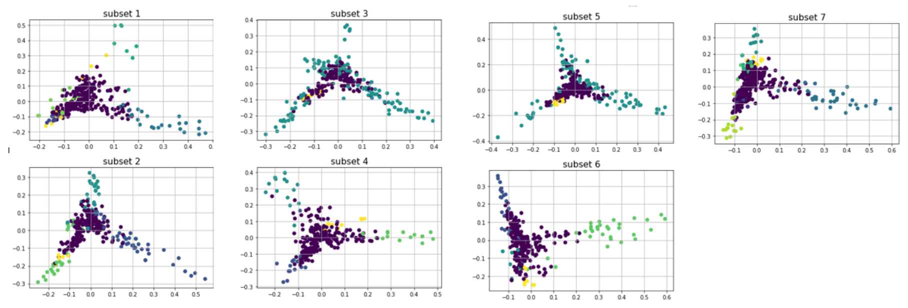


Figure 12: Cluster Graphs of the subsets

Evaluation scores of hdbscan can be seen in Table 4. When we compare it with DBSCAN, we can say that the results are slightly worse, but quite close results. We will see the importance of HDBSCAN when we perform performance analysis in the next section.

7. Benchmark

In this section, performance comparisons of the DBSCAN algorithm we implemented and the HDBSCAN algorithm were made.

7.1. Environment

Google Colab servers, whose features are shown below 5, were used to perform the tests.

subset	**cluster size**	**Rnd Ind**	**AMI**	**Homogenity**	**silhouette**	**CH index**
1	6	0.0293	0.2161	0.1871	0.0031	2.3665
2	5	0.1025	0.2962	0.262	0.0029	2.3276
3	3	0.021	0.05	0.0465	0.0022	1.8495
4	5	0.0307	0.2089	0.1665	0.0031	2.5727
5	3	0.0509	0.1322	0.1075	0.0018	2.1283
6	5	0.0406	0.2221	0.1843	-0.0003	2.5131
7	9	0.0407	0.2748	0.249	0.0045	2.7762
Average	5.1429	0.0451	0.2000	0.1718	0.0025	2.3620

Table 4: Evaluation scores of the HDBSCAN algorithm

Table 5: Device Specifications

Operating System	Linux-5.4.104+-x86_64-with-Ubuntu-18.04-bionic
Memory	13.3 GB
CPU Name	Intel(R) Xeon(R) CPU @ 2.20GHz
CPU Cores	1
CPU Cache	56320 KB
GPU Name	Tesla T4
GPU Memory	15.1 GB
CUDA Version	11.2

7.2. Tests

2D data was used for testing. This is because the DBSCAN algorithm works better on low-dimensional data and the sklearn library implemented versions include optimizations specific to low-dimensional data. Sklearn's makeblobs package was used to obtain the 2D data. This package Generates isotropic Gaussian blobs for clustering.The following results were obtained by generating data containing samples in the numbers [500,1000,5000,10000,50000,100000].

In tests, the execution time of our DBSCAN, sklearn's DBSCAN and HDB-

SCAN algorithms were compared.

7.2.1. 2D Data on CPU

First, the algorithms were run using only the CPU and Table 6 and Figure 13 were obtained. The numbers in the table indicate the seconds that pass during the execution of the algorithm. While calculating 100000 data size in the DBSCAN line, since the maximum execution time limit has been exceeded, +12 hours is written. If there was no limit, the estimated time for this test is expected to be more than 1 day. As can be seen from the results, there is a very serious time difference between our DBSCAN algorithm and the library algorithms. The reason for this is that library functions reduce the complexity of the DBSCAN algorithm from n2 to (nlogn) with the data structures they use in the distance calculation and use the improved versions of the DBSCAN algorithm. It is seen that it would not be correct to show all 3 algorithms on the same graph in figure 13, therefore the library versions were compared in the next graphs.

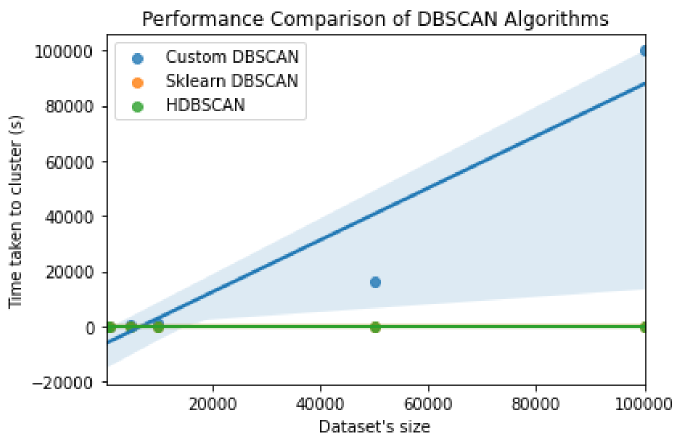


Figure 13: Execution time graph of algorithms on 2d data using CPU

Figure 14 shows the performance comparison between DBSCAN and HDBSCAN algorithms. While the time curve of the HDBSCAN algorithm is almost horizontal for this data size, the DBSCAN algorithm increases linearly.

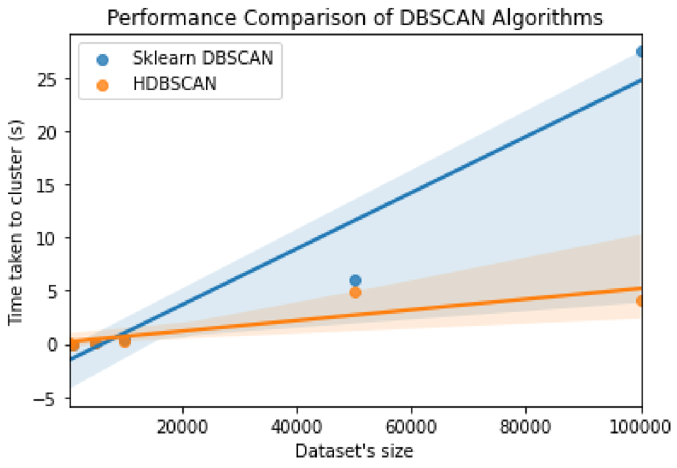


Figure 14: Execution time graph of algorithms on 2d data using CPU

	Dataset Size					
	500	1000	5000	10000	50000	100000
DBSCAN	3.4479	13.0225	334.8613	1347.7139	16 322.0418	+12 hours
DBSCAN (sklearn)	0.007735	0.017118	0.134665	0.38697	6.087746	27.518308
HDBSCAN	0.036746	0.021435	0.12711	0.258277	4.943284	4.166212

Table 6: Time elapsed(sec) during the execution of algorithms on CPU

7.2.2. 2D Data on GPU

In this section, the tests applied in the previous section were run on Tesla T4 GPU. Looking at the results in Table 7, it is seen that there is an improvement of 10-20 % in performance.

8. Conclusion

In this paper, we showed the process of data mining, described DBSCAN and HDBSCAN, and used these algorithms for text mining. We focused on the differences between the two algorithms.

HDBSCAN algorithm can provide comparable performance to the popular

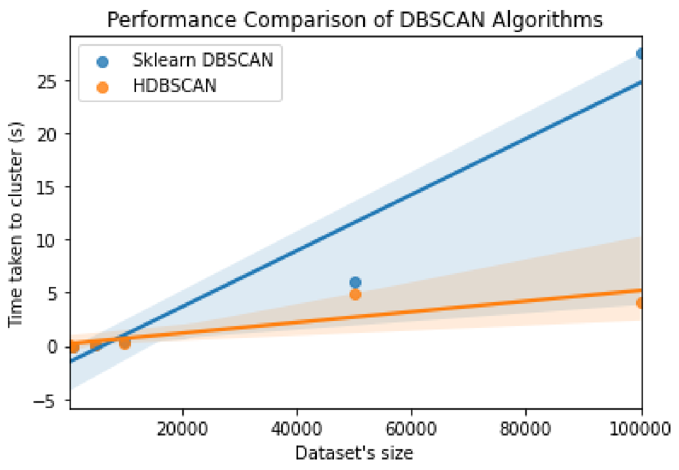


Figure 15: Execution time graph of algorithms on 2d data using GPU

	Dataset Size					
	500	1000	5000	10000	50000	100000
DBSCAN	2.9357	11.4814	283.2868	1133.0563	13796.2590	+12 hours
DBSCAN (sklearn)	0.008585	0.00988	0.113911	0.370377	5.411962	19.402351
HDBSCAN	0.021884	0.017642	0.101064	0.223361	4.075087	3.76294

Table 7: Time elapsed(sec) during the execution of algorithms on GPU

DBSCAN clustering algorithm. Since it has more intuitive parameters and can find variable density clusters, HDBSCAN is clearly superior to DBSCAN from a qualitative clustering perspective. As the improvements of our accelerated HDBSCAN make its computational scalability comparable in performance to DBSCAN, HDBSCAN should be the default choice for clustering.

A significant weakness of HDBSCAN algorithm as described is that it is inherently serial. The inability to parallel the algorithm is an obstacle for its use on large distributed data sets. Therefore, it is the major work in the further that changes HDBSCAN algorithm from serial algorithm to parallel algorithm. And we also think that the performance of HDBSCAN can be improve. Approximate nearest neighbor computations may offer significant performance improvements for a small trade-off in the accuracy of results.

References

[1] V. Kotu, B. Deshpande, Data science: concepts and practice, Morgan Kaufmann, 2018.

[2] J. Sander, M. Ester, H.-P. Kriegel, X. Xu, Density-based clustering in spatial databases: The algorithm gdbscan and its applications, Data mining and knowledge discovery 2 (2) (1998) 169–194.

[3] A. Dharmarajan, T. Velmurugan, Applications of partition based clustering algorithms: A survey, in: 2013 IEEE International Conference on Computational Intelligence and Computing Research, 2013, pp. 1–5. `doi:10.1109/ICCIC.2013.6724235`.

[4] A. Smiti, Z. Elouedi, Dbscan-gm: An improved clustering method based on gaussian means and dbscan techniques, in: 2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES), 2012, pp. 573–578. `doi:10.1109/INES.2012.6249802`.

[5] Y. Chen, S. Tang, N. Bouguila, C. Wang, J. Du, H. Li, A fast clustering algorithm based on pruning unnecessary distance computations in dbscan for high-dimensional data, Pattern Recognition 83 (2018) 375–387. `doi:https://doi.org/10.1016/j.patcog.2018.05.030`.
URL `https://www.sciencedirect.com/science/article/pii/S0031320318302103`

[6] I. Resources, Bbc dataset: Online.
URL `http://mlg.ucd.ie/datasets/bbc.html`

[7] D. Greene, P. Cunningham, Practical solutions to the problem of diagonal dominance in kernel document clustering, in: Proceedings of the

23rd International Conference on Machine Learning, ICML '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 377–384. doi:10.1145/1143844.1143892.
URL https://doi.org/10.1145/1143844.1143892

[8] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, K. Kochut, A brief survey of text mining: Classification, clustering and extraction techniques, arXiv preprint arXiv:1707.02919.

[9] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65. doi:https://doi.org/10.1016/0377-0427(87)90125-7.
URL https://www.sciencedirect.com/science/article/pii/0377042787901257

[10] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, X. Xu, Dbscan revisited, revisited: Why and how you should (still) use dbscan, ACM Trans. Database Syst. 42 (3). doi:10.1145/3068335.
URL https://doi.org/10.1145/3068335

[11] F. Ozkok, M. Celik, A new approach to determine eps parameter of dbscan algorithm, International Journal of Intelligent Systems and Applications in Engineering 5 (4) (2017) 247–251. doi:10.18201/ijisae.2017533899.
URL https://www.ijisae.org/IJISAE/article/view/649

[12] W.-T. Wang, Y.-L. Wu, C.-Y. Tang, M.-K. Hor, Adaptive density-based spatial clustering of applications with noise (dbscan) according to data, in: 2015 International Conference on Machine Learning and Cybernetics (ICMLC), Vol. 1, 2015, pp. 445–451. doi:10.1109/ICMLC.2015.7340962.

[13] H. Darong, W. Peng, Grid-based dbscan algorithm with referential parameters, Physics Procedia 24 (2012) 1166–1170, international Conference on Applied Physics and Industrial Engineering 2012.

doi:https://doi.org/10.1016/j.phpro.2012.02.174.
URL https://www.sciencedirect.com/science/article/pii/S1875389212002179

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[15] A. Rosenberg, J. Hirschberg, V-measure: A conditional entropy-based external cluster evaluation measure, in: Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL), 2007, pp. 410–420.

[16] A. P. Engelbrecht, Fundamentals of computational swarm intelligence, John Wiley & Sons, Inc., 2006.

[17] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Communications in Statistics 3 (1) (1974) 1–27. doi:10.1080/03610927408827101.

[18] L. McInnes, J. Healy, Accelerated hierarchical density based clustering, in: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE, 2017, pp. 33–42.