# Implementation of a KNN Consensus Clustering Algorithm Based On K Means

Riley de Domenico
December, 2021
Supervisors: Paula Branco, Guy-Vincent Jourdan

**Abstract**

There are a number of pre-existing clustering algorithms that perform the task of clustering well. However, combining these algorithms in a manner where they work together theoretically results in a more optimal solution. This project ca
rries out a study of four pre-existing algorithms, and one newly implemented KNN Consensus Clustering algorithm. In this study, both bigram and trigram features are used, and all algorithms' clustering solutions are compared against the ground truth labels with the use of various metrics. The study finds that the newly implemented KNN Consensus Clustering algorithm, that is based on the K Means algorithm, is more effective in its solution than any of the other tested algorithms. As such, this implementation is a better clustering solution than the other tested algorithms, both for bigram and trigram features.

## 1. Introduction

Oftentimes complex computing systems will incur errors, resulting in downtime for maintenance (Eriksson, 2016). A viable source that is used for diagnosis, auditing, and maintenance of these errors is log files (Ogbuagu, 2021). As multiple devices may be utilized across the same system, particularly in complex systems, log files will need to be examined frequently for maintenance purposes (Ogbuagu, 2021).

All major operating systems generate log files, which are produced for various reasons (Gavin, 2018). As such, log files have different meanings and uses depending on their purpose, and why they were generated.

| | | |
|---|---|---|
| 0 | Linux Apache | 023.012.8.10 - - [07/Oct/0010:09:29:10 -0400] "POST /obtain-your-degree/hwa-start/da %252Gzmgng_bulk_operations_fran/0ng4ko 217gr57l51s01214jvx5z0002f HTTP/1.0" 200 - "\Rosa/1.02 (linux-gnu)" |
| 1 | WinEventLog System | 00/06/1010 09:26:18 PM LogName=System LwkigrSgig=Alexandra-Windows- UhcvwOvfuiy XeiiwIwod=0201 EventType=4 Type=Information ComputerName=josephina. jeanene.ad.jeanene.yv User=NOT_TRANSLATED Tam=S-1-5-06 NmxIesz=0 RidfVdhmrijn=None OpCode=Start IekeclMaumwo=233441 Keywords=None Message=The Group Policy settings for the computer were processed successfully. New settings from 13 Group Policy objects were detected and applied |
| 2 | CISCO IOS | Oct 14 16:20:27 10.40.8.30 Oct 14 0010 20:20:26.160 BEE: %LINK-3-AUDRIE: Interface SqksfclXlibqohk1/0/12, changed state to up |

Figure 1. Example of log events generated from Linux Apache, Windows Event Log System, and CISCO IOS

Log files are typically generated in large amounts and have a semi-structured format, as such, a processing dilemma arises (Ogbuagu, 2021). To deal with this problem, log management solutions are often used to parse, store and index advanced log analysis. These solutions utilize machine learning techniques, particularly clustering, to group log entries with similar features (Ogbuagu, 2021). These clustering solutions allow for easier exploration through log files for anomaly detection.

To cluster log files, tokenization must occur. First, the features of each log event must be extracted, and then are grouped together based on a variable n. This would be clustering with n-gram features. In this project, bigram and trigram features are used, such that tokenization is performed on two and three features respectively.

Clustering solutions pose many advantages. For instance, clustering solutions do not require much explicit domain knowledge, as well correctly labeled training data is not usually necessary (Ogbuagu, 2021). Another benefit to clustering is that it provides a robust framework which can be efficient even if the type of log data is altered.

### 1.1 Problem Description

When we are faced with a large amount of data, it is more feasible to resort to a coded solution than it is to manually sort the data. In this report, 27,093 logs coming from 10 different sources will be processed; however, log files could come from hundreds of sources. Thus, this data needs to be sorted such that incoming logs are clustered with their correct source. To accurately perform this mapping, an effective clustering algorithm needs to be implemented.

To ensure that an effective clustering algorithm is implemented, there are a few things to consider:

- By what means are we measuring the effectiveness of an algorithm?
- What algorithms are being measured?
- Can this algorithm be further optimized or made more effective?

Based on these considerations, which of the measured clustering algorithms will be deemed the most effective, and how can it be further optimized or made more effective.

The aim of this project is to study various clustering algorithms in hopes of finding a solution to the question posed above. Thus, a comparative study of various algorithms will be undertaken and different clustering algorithms will be implemented based upon those that have been studied.

### 1.2 Objective

The objective of this project is not only to find the best possible clustering algorithm, but to improve upon pre-existing clustering algorithms. To do so, a nearest neighbour consensus clustering algorithm will be developed. It will take into account multiple algorithms and provide a better clustering solution than the best algorithm of those that have been explored.

## 2. Literature Review

Device logs are unstructured forms of data which are a necessity for monitoring a device (Ogbuagu, 2021). Analysis can be carried out on device logs, but this analysis requires converting the unstructured data into structured data (Ogbuagu, 2021). Multiple different solutions have been proposed to structure the data efficiently, but this can be difficult when log files come from various devices simultaneously. The first step, for detecting devices through device logs is by clustering. Clustering is an

approach that attempts to group data based on similarities (Ogbuagu, 2021). Clustering is useful for partitioning large data sets, or for detecting outliers.

Clustering is an effective approach to dealing with log files as it addresses various log message properties: the same log message gets repeated multiple times, some log messages feature high amounts of correlation in their appearance, and log messages include fixed tokens as well as variable tokens (Ogbuagu, 2021). Fixed tokens are static and do not change, while variable tokens differ based on the events properties. These log message properties are effectively addressed through means of grouping by similarity, which is what clustering is based upon.

Five different clustering methods were compared both with bigram and trigram features. This study found that the clustering algorithms performed best with trigram features as a result of the TF-IDF transformation better accentuating trigram features as opposed to bigram features (Ogbuagu, 2021). Further, it was also deemed that when the number of clusters is known, the K Means clustering algorithm performed better than the other tested algorithms (Ogbuagu, 2021).

As K Means was found to be the most effective algorithm, it is noteworthy to explore Bregman's clustering algorithm. This algorithm maintains the benefits of the K Means algorithm, while expanding the clustering method to a large class of clustering loss functions (Banerjee et al., 2004). Bregman's clustering is based on the idea of centroid parametric clustering, but instead through minimizing the Bregman's divergence (Banerjee et al., 2004).

To further build upon clustering algorithms, cluster ensembles have proven to be a better alternative (Vega-Pons & Ruiz-Shulcloper, 2011). Cluster ensembles consist of utilizing different sets of clustering partitions of the same data to converge on one clustering solution (Vega-Pons & Ruiz-Shulcloper, 2011). The aim of utilizing multiple clustering partitions is to improve the individual data. Thus, where an individual algorithm may falter, other algorithms can be relied upon. As such, ensemble clustering employs a variety of algorithms to achieve a better partition than that of one individual algorithm.

This literature review will explore different log clustering approaches, as well as new implementations that can be built upon to further find a more effective solution. As such, aiming to provide a solution to the question posed within the problem description. The related works section will

explore the different clustering techniques that were examined throughout this study with the purpose of addressing the problem previously posed. The objective section will detail the goal, and the methods explored to achieve this goal. As such, both sections will further expand upon Bregman's clustering as well as ensemble clustering.

### 2.1 Related Works

This section discusses the two clustering methods that were explored, as well as the analysis of different clustering techniques with the underlying goal of the project. Further, related works to Bregman's Hard Clustering, as well as Ensemble Clustering are examined as the focus of the project.

#### 2.1.1 Analysis of Different Clustering Techniques

In total four algorithms were examined and measured with bigram and trigram features. All of which produced good results. The algorithms examined were: K Means, BIRCH, HDBScan and DBScan.

The K Means algorithm is a partitioning algorithm based on a distance metric. K Means assumes a spherical partition shape. As such, the algorithm attempts to create exclusive partitions of the data through maximization of its distance function (Ogbuagu, 2021).

The BIRCH algorithm utilizes the implementation of hierarchical structures to create subclusters through means of a CF tree. This algorithm, unlike that of the K Means, does not assume a spherical shape to the partitions. However, the BIRCH algorithm is susceptible to noise (Ogbuagu, 2021).

The DBScan algorithm seeks to identify dense regions as clusters and sparse regions as noise, thus further separating the sparse regions from the denser ones (Ogbuagu, 2021). Unlike K Means and BIRCH, the number of clusters is not a necessary parameter for this algorithm.

The HDBScan algorithm is based on the DBScan algorithm as well as single linkage clustering. HDBScan takes a similar approach to DBScan in the identification of dense regions. However, it allows for a variance in cluster density, thus presenting a hybrid implementation of DBScan (Ogbuagu, 2021).

Upon the inspection of four different clustering algorithms, it was deemed that all clustering algorithms perform best with trigram features as opposed to bigram features (Ogbuagu, 2021). With these

same algorithms, and when the cluster count is available, the best clustering method for both bigram and trigram features is the K Means clustering algorithm (Ogbuagu, 2021).

### 2.1.2 Bregman's Hard Clustering

The Bregman's clustering algorithm is a clustering algorithm based on the Bregman divergence distortion function (Banerjee et al., 2004). With this algorithm, nodes are placed into a predefined number of clusters based upon distance to a centroid function, where the centroid is calculated by using the previously mentioned distortion function.

In simple terms, with convex function f: Rn → R, which is typically derivable, the Bregman divergence $B_f(\cdot, y)$ is the gap that exists between f and its linearization at y (Banerjee et al., 2004). We utilize the divergence to measure the distortion, and as such, to find the Bregman information of set X, with the simple encoding scheme S (Banerjee et al., 2004). Thus, we calculate the Bregman information with $E_v[d\varphi(X,S)]$, where dφ represents the Bregman divergence function (Banerjee et al., 2004). The optimal vector S, which achieves the minimal distortion is the Bregman representative of X. Where $I\varphi(X) = \min_{s \in ri}(S)\ E_v[d\varphi(X,S)]$ (Banerjee et al., 2004). Finally, with the set X, we split the set into k partitions, where each has its own Bregman representative. The k partitions are the number of clusters, and from this we calculate individual centroidal points using the Bregman's representative over different partitions of the set X (Banerjee et al., 2004).

As a whole, the Bregman's clustering algorithm is fairly similar to the K Means algorithm in that they both compute node distance to a centroid point. The main difference being that the Bregman's algorithm employs distortion functions for its centroid points. Bregman's clustering works in 3 phases: the initialize, assignment, and re-estimation phases (Banerjee et al., 2004). This means that the centroidal points are initialized and the distance to such points are calculated for each node. From here, we assign the nodes to a cluster based on its distance to the centroid. The assignment phase continues from the nearest node to the furthest node, until the defined limit is reached. Finally, the centroidal points are re-estimated using the distortion function, and these steps are repeated until convergence is reached (Banerjee et al., 2004).

Bregman's algorithm limits the number of nodes to add to a cluster through its assignment phase (Banerjee et al., 2004). As such, this leaves the algorithm susceptible to outliers. However, the algorithm

as a whole maintains the benefits of the K Means algorithm, while also expanding the clustering function (Banerjee et al., 2004).

### 2.1.3 Ensemble Clustering

No single clustering algorithm can correctly organize all sets of data (Vega-Pons & Ruiz-Shulcloper, 2011). As there is never one individual algorithm that is best to use in any given situation, it is reasonable to assume that employing an ensemble of algorithms, of which there are a wide variety, could yield better results (Vega-Pons & Ruiz-Shulcloper, 2011).

The idea of utilizing a combination of clustering results emerged as an approach geared towards improving the quality of those results. The idea behind ensemble clustering follows two main steps: the generation step, and the consensus function step (Vega-Pons & Ruiz-Shulcloper, 2011).

The generation step requires the creation of a set of partitions on the same data. As algorithms generate labels differently, no cluster will be labeled the same way. This poses a dilemma where a set of partitions exist, but there is no means to equate the partitions amongst the set (Vega-Pons & Ruiz-Shulcloper, 2011). As a result, a correspondence problem emerges.

The consensus function phase is the main step of ensemble clustering. The generation of an effective consensus function is the problem with this step. A few approaches exist to address this problem. These approaches are:

- Objects Co-occurrence: Here, the idea is to determine the correct cluster label that is associated to each node in the consensus partition. To do so, the partition set is analyzed to determine how many times the node belongs to one cluster, or how often two nodes belong to the same cluster (Vega-Pons & Ruiz-Shulcloper, 2011). Then, a consensus is obtained by means of voting among the nodes within the partitioning set. Voting is performed by each partition to assign the node to its respective cluster. An example of this approach is the relabeling and voting based methodology.
- Median Partition: Here, the consensus partition is obtained by an optimization solution. The aim is to find the median partition of the cluster ensemble (Vega-Pons & Ruiz-Shulcloper, 2011). This is done through mathematical means. An example of this approach is the Mirkin distance-based methods.

The relabeling and voting methodology first seeks to solve the label correspondence problem which emerges within the first step of the ensemble algorithm (Vega-Pons & Ruiz-Shulcloper, 2011). This issue is the primary difficulty with this methodology. To address the correspondence problem, a maximum likelihood solution can be used, wherein the Hungarian method is employed (Vega-Pons & Ruiz-Shulcloper, 2011). The Hungarian method is a combinatorial optimization solution that operates in four steps (HungarianAlgorithm.com, 2013). With a matrix, one must first subtract the row minima from each row. Then, subtract the column minima from each column. Next, determine how many rows or columns are required to cover all zero values. Then, find the smallest uncovered number, and subtract it from all uncovered elements. Finally, repeat the third step, where all zero values are covered. The end result is an optimal assignment amongst the zeros in the matrix (HungarianAlgorithm.com, 2013). This same method can be employed as a maximum likelihood problem to determine which cluster label has the highest likelihood of correspondence, thus forming a translation. The voting step takes on many different voting methodologies, however, as long as there is a vote between partitions, the algorithm is deemed to be following the relabeling and voting method (Vega-Pons & Ruiz-Shulcloper, 2011).

The Mirkin distance-based methodology takes into account two partitions $P_a$ and $P_b$ of the set X. From these partitions one has:

- $n_{00}$ which is the number of pairs of nodes that have been grouped into separate clusters in the two partitions.
- $n_{01}$ which is the number of pairs of nodes that have been grouped into different clusters in $P_z$ but in the same clusters in $P_b$.
- $n_{10}$ which is the number of pairs of nodes that have been grouped into different clusters in $P_b$ but in the same clusters in $P_a$.
- $n_{11}$ which is the number of pairs of nodes that have been grouped into the same clusters in both partitions.

The Mirkin distance is then represented as M and is calculated as:

$$M = n01 + n10 \quad (1)$$

This is a representation of the number of disagreements that exist between two partitions (Vega-Pons & Ruiz-Shulcloper, 2011). As such, the nodes that are disagreed upon are re-clustered, and various approaches exist to do so. Some approaches include:

- Best-of-k: This approach consists of selecting the partition P from the set of partitions that is closest to the main partition (Vega-Pons & Ruiz-Shulcloper, 2011). The partitioning selection which is made minimizes distance from the node to other partitions within the ensemble.
- Simulated Annealing One Element Move: This approach guesses at an initial partition, and repeatedly updates the partition by moving one node to another cluster (Vega-Pons & Ruiz-Shulcloper, 2011).
- Best One Element Move: This approach is based on the simulated annealing one element move approach, but instead commences with an initial partition (Vega-Pons & Ruiz-Shulcloper, 2011). As a result, a greedy process is followed whereby nodes are repeatedly moved from one cluster to another while simultaneously updating the partition.

In total, ensemble clustering is a leading technique for clustering analysis as these techniques are capable of correcting individual algorithm mistakes by means of incorporating other algorithms (Vega-Pons & Ruiz-Shulcloper, 2011). Thereby allowing for the compensation of errors made by other algorithms.

## 3. Methodology

Typically, clustering algorithms are utilized when existing information on the data is limited; however, in this case the classification of the log files is known. With this, metrics can be selected to determine the effectiveness of the algorithms as if a supervised clustering solution is being performed.

Further, the details of the KNN Consensus clustering algorithm's implementation will be included as mentioned in the objective section of this paper. This section will determine the metrics to be used to analyze an algorithm's effectiveness and the algorithms that will be implemented including the Consensus algorithm that improves upon pre-existing algorithms.

### 3.1 Evaluation Criteria

The ground truth labels will be utilized as the basis for these evaluations. The following six different metrics will be used: Homogeneity, Completeness, V-Measure, Adjusted Rand Index, Normalized Mutual Information, and Fowlkes Mallows Index.

### 3.1.1 Homogeneity

The homogeneity metric is satisfied when all generated clusters contain only data points that are members of a single class (Ogbuagu, 2021). It follows that the objects of a cluster all belong to a similar class. This metric is not prone to change if the cluster labels or ground truth labels are rearranged in any way. The closer this metric is to the value 1, the more homogenous the clustering solution becomes.

### 3.1.2 Completeness

The completeness metric is satisfied when all data points of a specific class are placed within the same cluster (Ogbuagu, 2021). Therefore, the objects of a class belong to the same cluster. This metric is not prone to change if the cluster labels or ground truth labels are rearranged in any way. The closer this metric is to the value 1, the more complete the clustering solution.

### 3.1.3 V-Measure

The V-Measure metric represents the harmonic mean of the homogeneity and completeness metrics (Ogbuagu, 2021). This metric is varied using the beta parameter to assign a weight to the homogenous value (Ogbuagu, 2021). V-Measure is calculated as:

$$v = \frac{(1+\beta)*homogeneity*completeness}{\beta*homogeneity*completeness} \quad (2)$$

This metric is not prone to change if the cluster labels or ground truth labels are rearranged in any way.

### 3.1.4 Adjusted Rand Index (ARI)

The Rand Index (RI) metric considers both the true and predicted solutions. The pairs that belong to similar or different clusters in the comparison of the two solutions are counted (Ogbuagu, 2021). The ARI metric is a version of the RI metric that is adjusted for chance (Ogbuagu, 2021). Thus, the ARI metric scores better in cases where clustering solutions do not feature extra class labels. ARI is calculated as:

$$ARI = \frac{RI - Expected\ RI}{(maxRI) - Expected\ RI} \quad (3)$$

### 3.1.5 Normalized Mutual Information (NMI)

The Mutual Information (MI) metric is a measure of similarity between two labeling solutions of the same data (Ogbuagu, 2021). One solution being the true solution, and the other being the predicted solution. Given $|S_i|$ as the number of samples in cluster $S_i$ and $|T_i|$ as the number of samples in cluster $T_i$, the MI is calculated as:

$$MI\ (S,T)\ =\ \sum_{i=1}^{|S|}\sum_{j=1}^{|T|}\frac{|Si\cap Tj|}{N}log(\frac{N|Si\cap Tj|}{|Si||Tj|}) \tag{4}$$

The NMI is a normalization of the MI. To do so, the MI score is normalized to be between the values of 0 and 1 through the usage of a generalized mean that is defined by the average method parameter (Ogbuagu, 2021). This metric is not prone to change if the cluster labels or ground truth labels are rearranged in any way.

### 3.1.6 Fowlkes - Mallows Index (FMI)

The FMI metric is the geometric mean of precision and recall (Ogbuagu, 2021). FMI takes into consideration: the number of true positives (TP), which is the total number of pairs of nodes that are found in the same clusters for both true and predicted labeling solutions. The number of false positives (FP), similar to TP, takes a total of pairs of points that are found in the true solution, but not in the predicted solution. The number of false negatives (FN), which is the number of pairs of points that are found in the predicted solution, but not in the true solution. As such, FMI is calculated as:

$$FMI\ =\ \frac{TP}{\sqrt{(TP+FP)*(TP+FN)}} \tag{5}$$

## 3.2 Algorithms

This section will detail the five algorithms that will be explored throughout this project. The algorithms include: K Means, Bregmans, Agglomerative, HDBScan, and the implementation of Consensus Clustering.

### 3.2.1 K Means Clustering

K Means Clustering is a partitioning clustering algorithm in which the data is segmented into a set number of clusters (Dabbura, 2020). The number of clusters must be specified for K Means

Clustering, this being one of its drawbacks (Dabbura, 2020). From this, the algorithm generates n centroids where n is the number of clusters. Further the n centroids are calculated through the mean of the objects within the cluster, and periodically the centroid gets recalculated until the optimal centroid is found (Ogbuagu, 2021). As this is a partitioning algorithm, this is a distance-based algorithm.

We can measure the quality of each cluster with the sum of the squared differences between each point in a cluster and that cluster's centroid. The goal then is to improve our measurement of the cluster, the within-cluster variation, as best as possible, such that the solution is optimized (Ogbuagu, 2021).

The within-cluster variation can be defined as:

$$E = \sum_{i=1}^{k} \sum_{p \epsilon C_i} dist(p, c_i)^2 \quad (6)$$

As this is a partitioning algorithm, it has its drawbacks. One downfall to this partitioning algorithm is that it can be computationally inefficient to determine the cluster quality (Ogbuagu, 2021). As such, the K Means algorithm aims to overcome this through the means of a greedy approach. The cluster quality gets calculated by iteratively performing the mean of a cluster using its assigned nodes from the previous iteration.

### 3.2.2 Bregman Hard Clustering

Bregman Hard Clustering is a partitioning clustering algorithm which shares similarities with K Means Clustering. A partitioning algorithm attempts to organize data in a predefined number of clusters based on a measurement of distance (Ogbuagu, 2021). Bregman Hard Clustering then differs from K Means Clustering, as it is based on ideas which come from Shannon's rate distortion theory (Banerjee et al., 2004).

Shannon's rate distortion theory relates to the compression of data, stemming from some information source, to a specified encoding rate which is less than the source's entropy (Yamamoto, 1997). This encourages distortion between the original signal, and the encoded version. As such, in relation to the Bregman Hard Clustering, an encoding scheme must be found with a given rate such that the source variable and decoded variable are minimized (Banerjee et al., 2004). From here, a Bregman

divergence is used to measure the distortion. As such, one can formulate a distortion rate function which incorporates the Bregman divergence measurement that can be optimized. The optimal distortion rate function is referred to as Bregman information (Banerjee et al., 2004).

The Bregman Hard Clustering problem then becomes a quantization problem where there is an attempt to reduce the loss of Bregman information (Banerjee et al., 2004). This is done to find the Bregman representative which is the optimal vector for the formulated distortion rate function (Banerjee et al., 2004). The Bregman representative provides the minimal distortion to the formula.

The Bregman information, $I_{\varphi}$,for the variable $X$, of divergence $d_{\varphi}$ is denoted as:

$$I\varphi = min_{s \epsilon ri(S)} E_v[d\varphi(X, s)] = min_{s \epsilon ri(S)} \Sigma_{i=1}^{n} v_i d\varphi(x_i, s)] \quad (7)$$

Where an encoding scheme represents the variable by a constant vector s, S is the convex, X is representative of the random variables taken from a finite set $x_i$, and v is the discrete probability measure (Banerjee et al., 2004).

Natural iterative relocation algorithms can be considered as an option to solve this problem. Consequently, Bregman Hard Clustering is similar to K Means Clustering; however, the iterative relocation of centroidal points of clusters aims to minimize the loss of Bregman information (Banerjee et al., 2004). This is performed in three stages: initialization, assignment, and re-estimation. First, initialize the centroids as well as nodes. Second, periodically assign nodes to each centroid based on the nodes proximity to the centroid. This is done until a cutoff point is reached. The purpose for the cutoff point is to reach a lower quantization error. A smaller set of Bregman information is maintained by partitioning the set of nodes accordingly. The Bregman information is re-estimated and so are the centroidal points in an attempt to minimize the loss of Bregman information. This process is repeated until the expected Bregman divergences of points from the original set are minimized in correspondence to their representatives.

Pseudo Code of Bregman Hard Clustering (Banerjee et al., 2004):
Input: *X, v, dφ, k* (Number of clusters)

Output: $M^t$ , local minimizer of $L\varphi(M) = \sum^{k}_{h=1} \sum_{xi \in Xh} v_i d\varphi(x_i, \mu_h)$ where $M = \{\mu_h\}^{k}_{h=1}$ and the hard partitioning of $X$.

Method:

First we must initialize $\{\mu_h\}^{k}_{h=1}$ *with* $\mu_h \in ri(S)$

//Assignment

Set $X_h \leftarrow 0,\ 1 \le h \le k$

For $i = 1$ to m do

$X_h \leftarrow X_h \cup \{x_i\}$

where $h = h^t(x_i) = argmin^{h'} d\varphi(x_i, \mu_{h'})$

End for loop

//Re-estimate

For $h = 1$ to k do

$\pi_h \leftarrow \sum_{xi \in Xh} v_i$

$\mu_h \leftarrow 1/\pi_h \sum_{xi \in Xh} v_i x_i$

End for

Repeat until convergence is achieved

//Convergence is when the points are minimized in correspondence to their representatives. Further this convergence is proven to exist as long as equality holds, such that if the loss function is equivalent multiple times in proceeding iterations, the algorithm will terminate.

return $M^t \leftarrow \{\mu_h\}^{k}_{h=1}$

**3.2.3 Agglomerative Clustering**

Agglomerative Clustering is a form of hierarchical clustering which performs its grouping based on a node's similarity with another node. The agglomerative algorithm utilizes a tree-based structure to merge the nodes in pairs into clusters, where the overarching result is an all-encompassing clustering containing the clusters within it (Kassambara, 2018).

In simplest terms, the algorithm iterates over all nodes, where at each iteration the most similar nodes or clusters are merged together to form a cluster (Kassambara, 2018). This iteration occurs until all nodes are within a cluster. By default, the similarity is calculated through the node's Euclidean distance (Kassambara, 2018). The agglomerative algorithm performs the following steps throughout its process:

1. Calculating the Euclidean distances of the nodes and clusters.

2. Computing similarity.
3. Using a linkage function to join the most similar nodes or clusters in the hierarchical tree structure.
4. Repeating steps 1 through 3 until all nodes are clustered, then deciding where to cut the tree structure into separate clusters.

The final step in this process is performed with the pre-existing knowledge of how many clusters exist; therefore, the number of clusters is a required parameter for this algorithm (Kassambara, 2018).

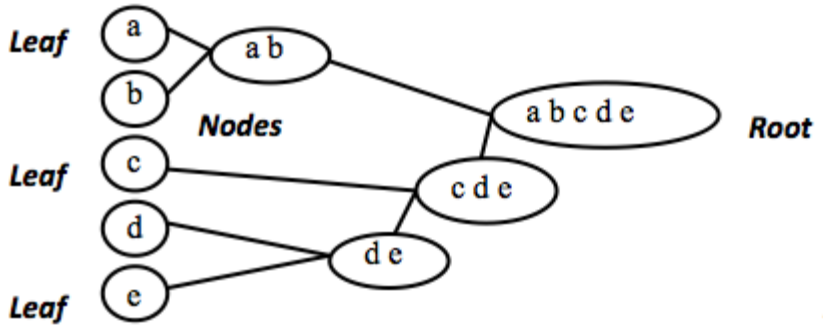


Figure 2. Agglomerative Clustering Algorithm Hierarchical Tree Structure

### 3.2.4 HDBScan Clustering

The HDBScan algorithm is a hybrid algorithm which makes use of both hierarchical and density based clustering methods (Leland et al., 2016). This algorithm builds upon the DBScan algorithm.

This algorithm, at its base, uses a single linkage clustering technique similar to that of the agglomerative clustering (Ogbuagu, 2021). The density based clustering aspect is then used where the density within each cluster is estimated through a node's distance $k^{th}$ nearest neighbour (Ogbuagu, 2021). To disperse points that exist within low density regions, a mutual reachability distance metric, d, is defined between two points, a and b:

$$d(a,b) = max(core(a), core(b), distance(a,b)) \quad (7)$$

This metric aims to separate nodes within sparse regions such that they exist a core distance away from each other, thus allowing for less dense regions to be clustered in an optimal manner. While also

utilizing the single linkage clustering. Therefore, nodes are clustered in a single linkage manner based upon their density regions.

This algorithm produces a minimal spanning tree, which is converted into a hierarchical tree containing the connected components (Leland et al., 2016). As the number of clusters is not a parameter for this algorithm, the clusters' labeling is gathered by selecting a horizontal region of the tree, and utilizing the clusters within this region.

### 3.2.5 Consensus Clustering

The idea behind Consensus Clustering is to gather the opinion of multiple clustering algorithms, and compare the clustering of a node x to its nearest neighbours (Vega-Pons & Ruiz-Shulcloper, 2011). This can either form an agreement or a disagreement. Agreements are handled simply by maintaining the current labeling of the node. Disagreements however need to be resolved.

The Consensus Clustering algorithm takes into account 3 algorithms. The one in which it is based upon is the K Means algorithm. The other 2 algorithms are HDBScan, and Agglomerative Clustering. When the most common nearest neighbours of the HDBScan algorithm or the Agglomerative Clustering Algorithm agree with the clustering of x from the KMeans algorithm, there is an agreement. If this is not the case, the result is a disagreement.

To resolve a disagreement, the decision is based on the HDBScan algorithm and the Agglomerative algorithm. In the case that these two algorithms incur similar labeling of x’s nearest neighbours, such that the majority of nodes are placed in the same cluster, a change to the labeling of x to match the clustering of these two algorithms is made.

In some cases, no agreements are encountered, and as such this situation needs to be addressed as well. When this happens, there is a need to utilize the silhouette coefficient to find which cluster node x belongs to best.

The silhouette coefficient is a metric used to determine how well a node belongs within a cluster (Scikit Learn, 2017). As such, there is a need to determine if x belongs in its already determined cluster, or if it would best be suited in the cluster labeling of the most common nearest neighbour from the

HDBScan algorithm and Agglomerative algorithm. To do this, the Silhouette Coefficient must be calculated and one must determine which labeling would provide the largest Silhouette Coefficient.

The Silhouette Coefficient, *S*, for the mean intra cluster distance, *a*, and the mean inter cluster distance, b, is denoted as:

$$S = (b - a)/max(a, b) \quad (8)$$

Using the silhouette coefficient determines which cluster x belongs to best. As an end result, the aim is to correct any shortcomings of the K Means Clustering Algorithm with the aid of non-partitioning algorithms. Thus, being able to correct the labeling of nodes.

Pseudo Code of Consensus Clustering:
Input: *X* (The array of data to be clustered), n (number of clusters)
Output: X_Labels (The partitioning of *X*).
Method:
//Partition *X* with K Means, agglomerative, HDBScan Clustering
KMeans_Labels = Kmeans(X,n)
HDBScan_Labels = HDBScan(X)
Agglomerative_Labels = Agglomerative(X,n)
//Translation of labeling with a contingency matrix using hungarian method
Labels = translateLabels(KMeans_Labels,HDBScan_Labels,Agglomerative_Labels)
//Labels is a 2D Array of the labels translated in terms of KMeans
i=0
For *x* in *X*
//Get nearest neighbors, where tempLab is the 5 nearest neighbours labels
tempLab = nearest(X,x,5,Labels) //nearest calculates the euclidean distance, returning nearest
//Consensus voting
acLabels = tempLab[1]
hdbLabels = tempLab[2]
kLabel = KMeans_Labels[i]
ac = mostCommon(acLabels) //find which cluster label is most common

```
hdb = mostCommon(hdbLabels)
if (ac is kLabel or hdb is kLabel)  //majority of other algorithms labels are in the same cluster
        then X_Labels[i] = kLabel
else if (ac is hdb) //When other two algorithms agree on a cluster, but KMeans does not agree
        then X_Labels[i] = ac
else //The case where no algorithm agrees
        then tempLabels = KMeans_Labels
        tempLabels[i] = ac
        acSilhouette = silhouette(X,tempLabels)[i]
        tempLabels[i] = hdb
        hdbSilhouette = silhouette(X,tempLabels)[i]
        kSilhouette = silhouette(X,KMeans_Labels)[i]
        if acSilhouette is greater than hdbSilhouette and kSilhouette
                then X_Labels[i] = ac
        else if hdbSilhouette is greater than acSilhouette and kSilhouette
                then X_Labels[i] = hdb
        else
                then X_Labels[i] = kLabel
i+=1
end for loop
return X_Labels
```

**3.3 Experiment Process**

For this project, the algorithms previously detailed have been implemented in the code, which can be found in the experiment code section below. With these implementations, the metrics previously depicted will be computed to determine the effectiveness of the algorithms. Further, there will be a comparison of this data and a conclusion produced from the findings.

**3.4 Experiment Code**

The code for this experiment can be found at: *https://github.com/rdedo099/HonoursProject2021*

## 4. Results

This section aims to visualize the results of the algorithms through the usage of U Maps and display the metrics computed with the ground truths as well as the algorithms output for both bigram and trigram features. There will be a discussion of the findings for each algorithm along with the meaning of the results. In total, the effectiveness of each algorithm will be depicted through the metrics of Homogeneity, Completeness, V-Measure, ARI, NMI, and FMI.

As well, the clustered data is visualized with the use of Uniform Manifold Approximate & Projection (UMAP) dimensional reduction process. The UMAP representation gives visual insight into the clustered solution, but is not exhaustive.

### 4.1 K Means Clustering

As seen from table 1 below, the results produced by the K Means Clustering algorithm are very good. As noted, the K Means algorithm is a partitioning algorithm which is based on distance, as such the algorithm assumes the cluster shape to be spherical and struggles to differentiate within cluster variation (Dabbura, 2020). Figures 1 and 2 both show that the visualization presents the cluster's membership in a spherical manner. This further emphasizes the issue of nodes being falsely added to a cluster, particularly around the cluster's edges. K Means is susceptible to noise, which can be problematic. Between the K Means Bigram and Trigram results in table 1, there were no major differences or changes between the two results. Slight improvements in completeness in VMeasure and NMI were seen, while slight decreases in ARI and FMI were also seen when comparing the Trigram data with that of the Bigram data. Similar to what was mentioned within the related works section, the K Means algorithm is highly effective, so it was utilized as the basis for the Consensus Algorithm in hopes of improving upon its effectiveness.

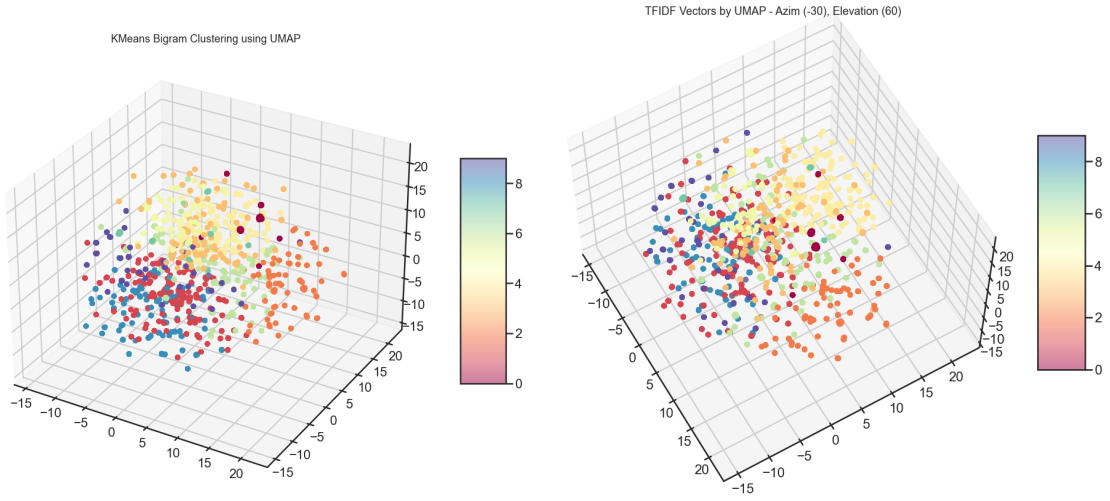


Figure 3. 3D plot of K Means Clustering with Bigram Features.

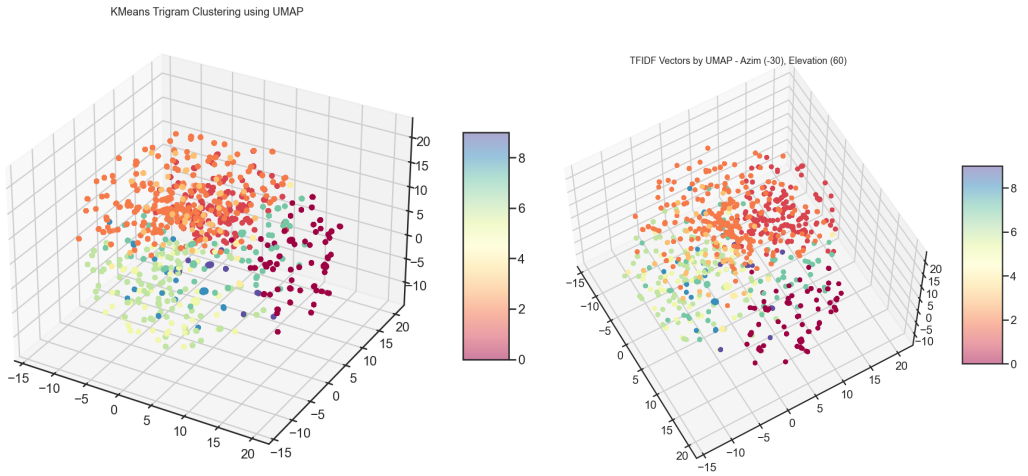


Figure 4. 3D plot of K Means Clustering with Trigram Features.

Table 1. Evaluation of K Means Clustering on Both Bigram and Trigram Features

| | **Homogeneity** | **Completeness** | **V Measure** | **ARI** | **NMI** | **FMI** |
|---|---|---|---|---|---|---|
| **Bigram** | 0.83 | 0.86 | 0.85 | 0.70 | 0.85 | 0.73 |
| **Trigram** | 0.83 | 0.89 | 0.86 | 0.67 | 0.86 | 0.71 |

### 4.2 Bregman Hard Clustering

By evaluation of the bigram and trigram clustering results of the Bregman Hard Clustering algorithm, it was determined that the results seen were below average for effectiveness. As this clustering algorithm is similar to K Means, comparisons can be drawn. However, it did not have results on par with the K Means algorithm, meaning it is less effective. The main difference noticed between the bigram and trigram results was that the results from trigram clustering overall had slight improvements in some metrics, while also suffering larger losses in other metrics when compared to the bigram clustering results as seen in table 2. An example of this would be a decrease by 0.06 in completeness from bigram clustering to trigram clustering, but also having an increase of 0.01 in homogeneity. Since Bregman Hard Clustering has many similarities to K Means clustering it shares in many of its benefits, such as scalability. However, it still has the disadvantage of requiring the information regarding the number of clusters. Another downside to Bregman Hard Clustering, is the cutoff point. This cutoff point can limit valid members of a cluster from being included in that cluster, while also allowing for outliers to be included. Since Bregman Hard Clustering utilizes Bregman divergence to calculate a measurement between nodes, a spherical relation is observed in the data, as seen in the 3D plots of the Bregman Hard Clustering performance below. This can also be due to the ideas on which partition clustering is based.

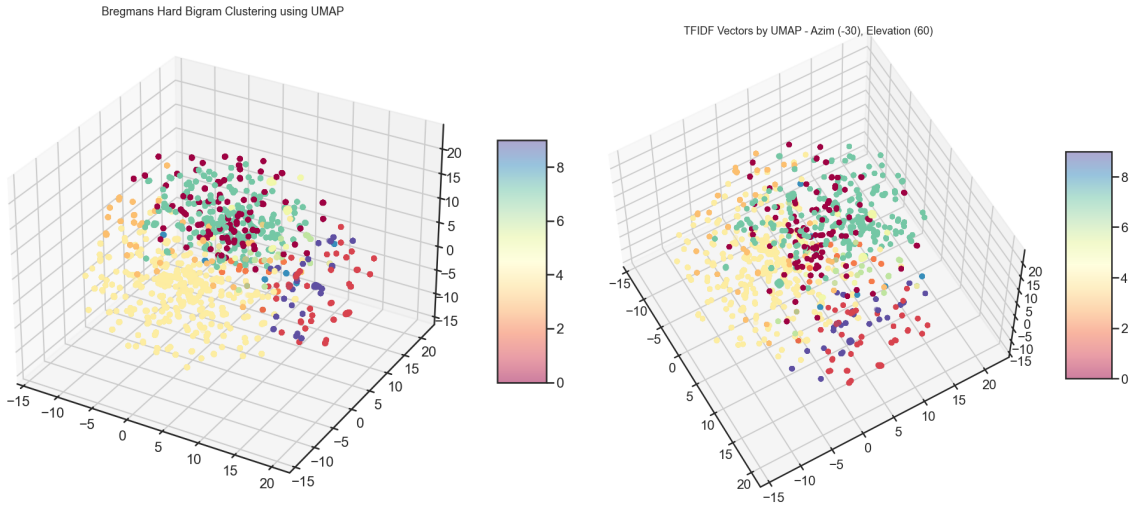


Figure 5. 3D plot of Bregman Hard Clustering with Bigram Features.

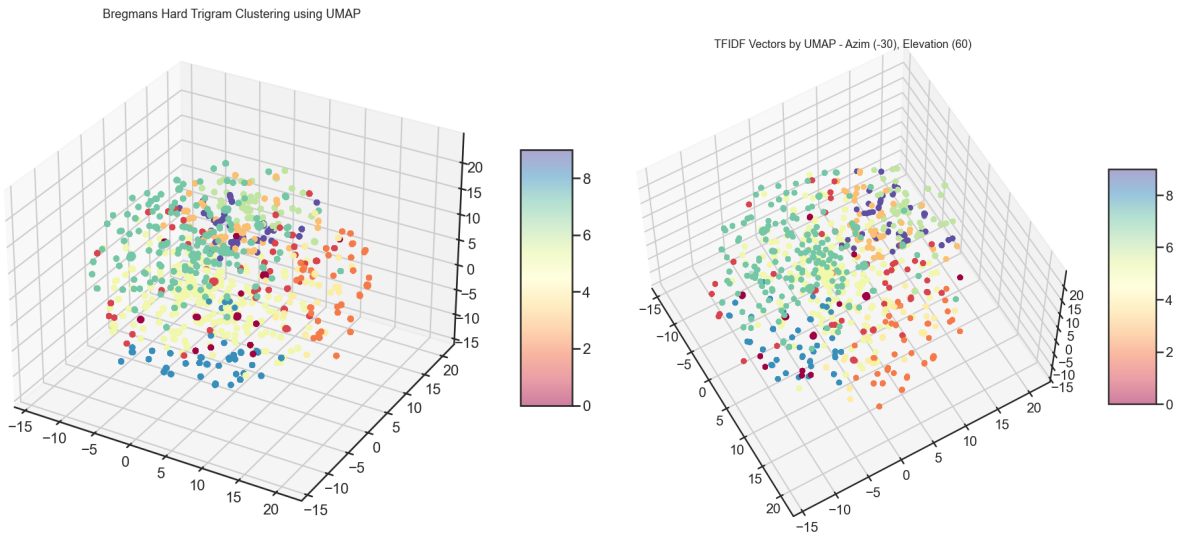


Figure 6. 3D plot of Bregman Hard Clustering with Trigram Features.

Table 2. Evaluation of Bregman Hard Clustering on Both Bigram and Trigram Features

| | **Homogeneity** | **Completeness** | **V Measure** | **ARI** | **NMI** | **FMI** |
|---|---|---|---|---|---|---|
| **Bigram** | 0.69 | 0.86 | 0.77 | 0.47 | 0.77 | 0.59 |
| **Trigram** | 0.70 | 0.80 | 0.75 | 0.48 | 0.75 | 0.56 |

### 4.3 Agglomerative Clustering

The agglomerative algorithm is fairly effective, albeit, not the most effective. The algorithm performs well in terms of completeness, V-Measure and NMI. Between the bigram and trigram features, there are no major differences other than the ARI and FMI scores which differ by 0.06 and 0.05 respectively. As well, in the figures below it can be seen that one cluster (the orange cluster) is far more distinct than any other cluster present; therefore, it is believed that this cluster is being over-represented. As such, some of these nodes possibly belong to other clusters. This could be the cause of some scores being less optimal than desired. In addition, this could be caused by the similarity factor being calculated by distance to the cluster, such that, as this cluster grew, it became more similar to nodes nearby. As such, nodes which possibly could have been more similar to other clusters were falsely labeled to belong to this cluster. This can be a flaw of a hierarchical method of clustering, as the clusters tend to bubble, where nearby objects are merged, even if they belong elsewhere.

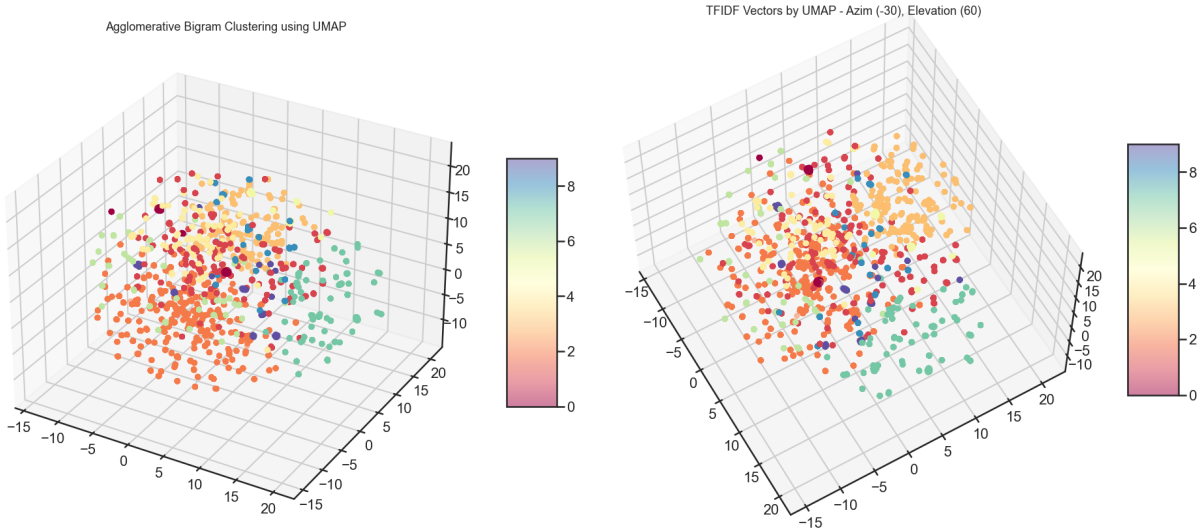


Figure 7. 3D plot of Agglomerative Clustering with Bigram Features.

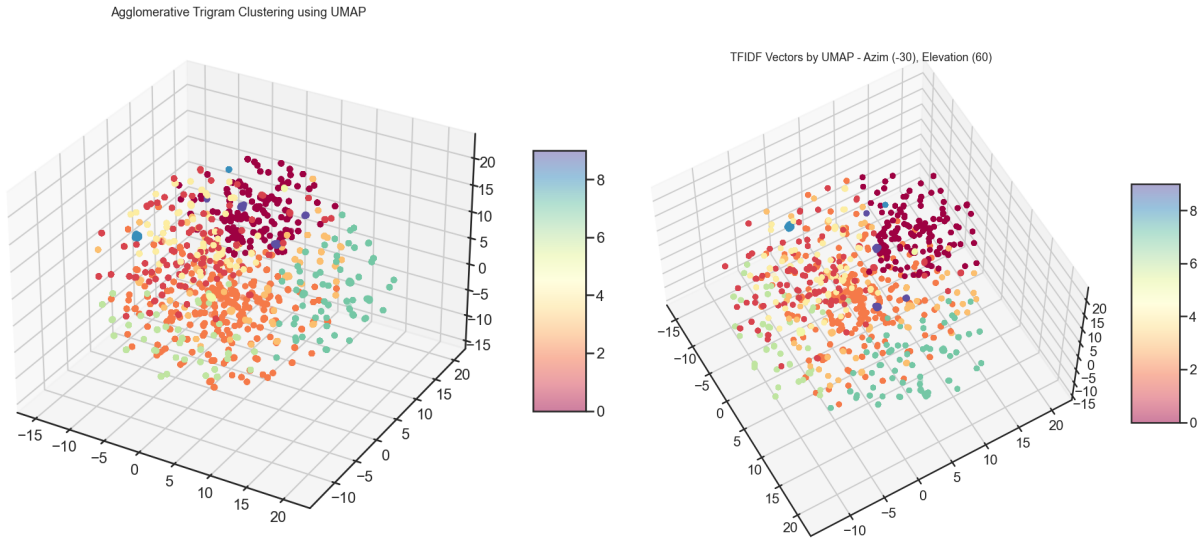


Figure 8. 3D plot of Agglomerative Clustering with Trigram Features.

Table 3. Evaluation of Agglomerative Clustering on Both Bigram and Trigram Features

| | Homogeneity | Completeness | V Measure | ARI | NMI | FMI |
|---|---|---|---|---|---|---|
| **Bigram** | 0.79 | 0.85 | 0.82 | 0.60 | 0.82 | 0.66 |
| **Trigram** | 0.80 | 0.88 | 0.84 | 0.66 | 0.84 | 0.71 |

### 4.4 HDBScan Clustering

It is noticeable that HDBScan performs better with trigram features than with bigram features; this is most noticeable with the difference in all metrics being greater than or equal to 0.04. Hence, this algorithm is not as effective as desired, or as consistent as hoped. Nevertheless, it is still fairly effective given that it operates under the circumstance that the number of clusters is an unknown factor whereas all the other algorithms are aware of the number of clusters. This is especially important to note as some of the metrics punish algorithms for including more algorithms than actually exist . As seen from the figures below, more than 10 clusters are displayed, and as such, it is safe to assume a penalty was incurred. In total, this algorithm is still effective as it provides good scores for most of the trigram's features, such as the homogeneity being 0.84. In comparison, this is the highest homogeneity score seen for trigram features.

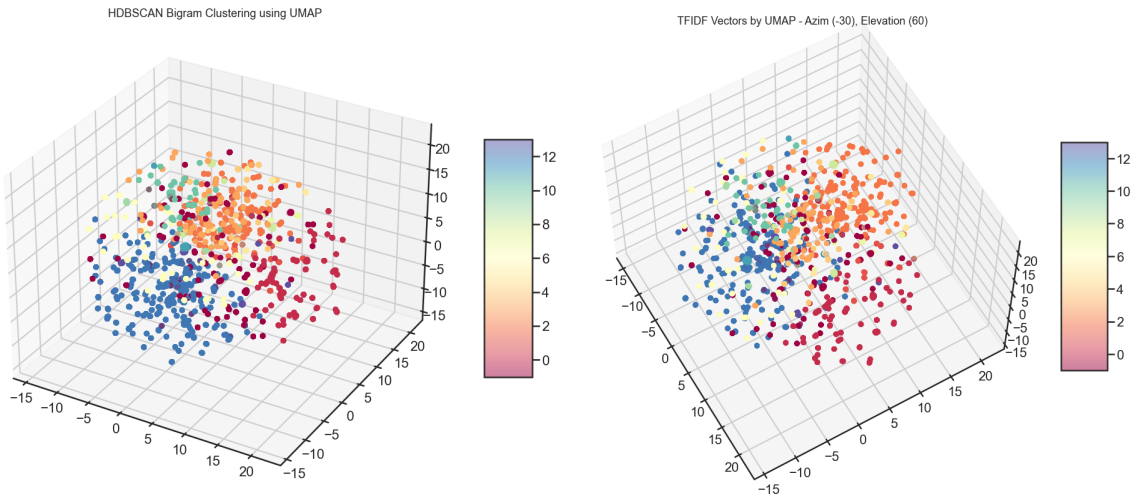


Figure 9. 3D plot of HDBScan Clustering with Bigram Features.

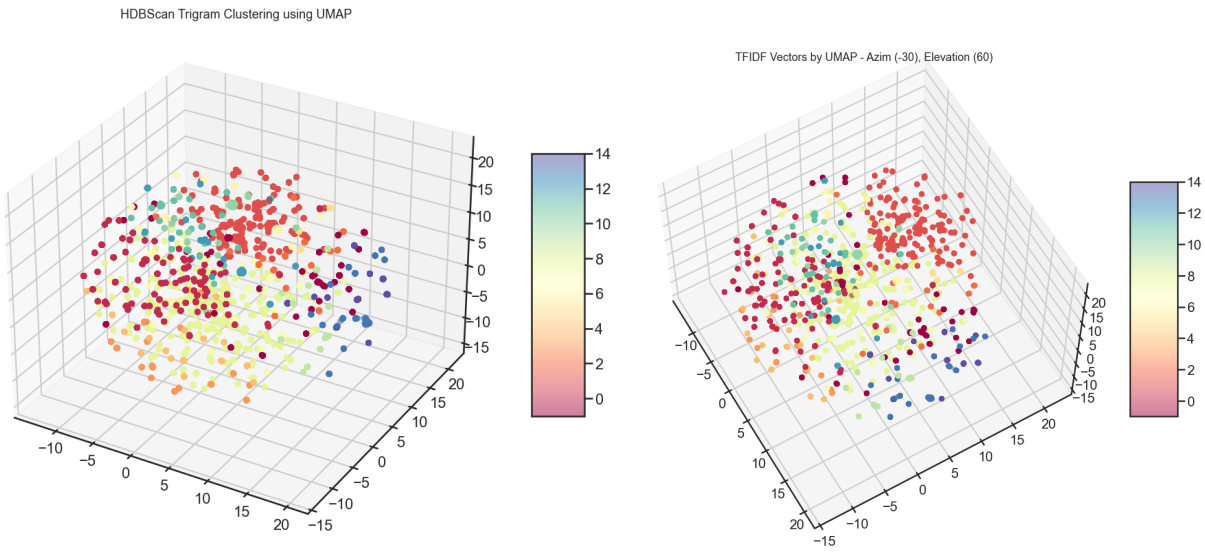


Figure 10. 3D plot of HDBScan Clustering with Trigram Features.

Table 5. Evaluation of HDBScan Clustering on Both Bigram and Trigram Features

| | **Homogeneity** | **Completeness** | **V Measure** | **ARI** | **NMI** | **FMI** |
|---|---|---|---|---|---|---|
| **Bigram** | 0.79 | 0.75 | 0.77 | 0.55 | 0.77 | 0.60 |
| **Trigram** | 0.84 | 0.79 | 0.81 | 0.62 | 0.81 | 0.67 |

### 4.5 Consensus Clustering

With the implemented Consensus Clustering, the implementation is based on the K Means algorithm. It is important to note the amount of difference seen between this implementation and that of the K Means. With regards to the Consensus Clustering algorithm, there was experimentation with different numbers of neighbours with whom to run the consensus.. The parameters three, five and seven were utilized for the number of neighbours. In terms of bigram features, there were 2696 different node labelings when the algorithm was run with three neighbours. With trigram features, there were 1178 different node labelings. For five neighbours, there were 2692 different node labelings when the algorithm was run with bigram features and 1175 different node labelings when the algorithm was run with trigram

features. With seven neighbours, there were 2964 different node labelings with bigram features, and a total of 1175 different node labelings with trigram features. As the amount of differences with the adjusted parameters is minimal, the data and metrics remained consistent and no changes were seen amongst the different parameters. As such, the 3D plots below are based upon the five neighbour implementation. In this implementation, it can be seen that clusters are still somewhat based on proximity; however, there is more sparseness to the solution as a whole. The 3D plot of Figure 9 closely resembles that of Figure 1, however there is a larger distinction in unique clusters, so the clusters appear to be more firmly defined. In terms of the 3D plot of the clustering with trigram features in Figure 10, the same can be said, when this plot is compared with Figure 2. The solution poses more distinct clusters, with seemingly less overlap. In total, this algorithm is very consistent with both bigram and trigram features, and is also very effective. The algorithm's effectiveness is most notable in terms of the completeness score being 0.91, which is fairly close to the optimal value of 1.

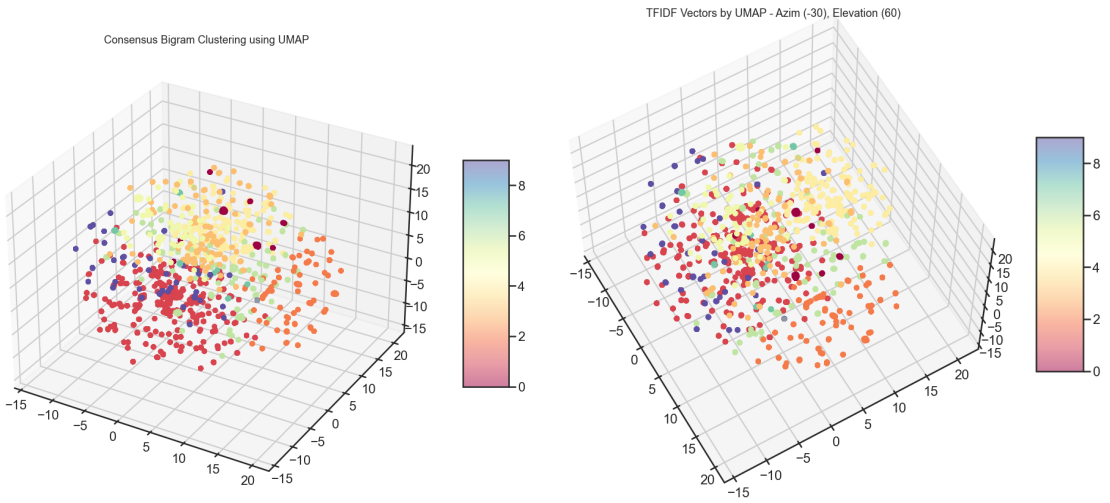


Figure 11. 3D plot of the Consensus Clustering with Bigram Features.

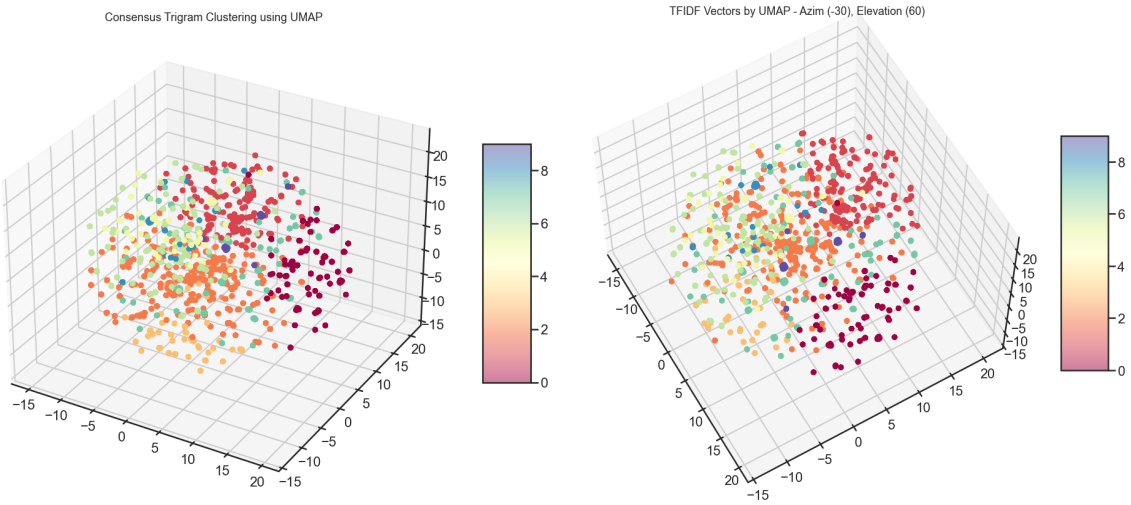


Figure 12. 3D plot of the Consensus Clustering with Trigram Features.

Table 6. Evaluation of the Consensus Clustering Algorithm on Both Bigram and Trigram Features

| | **Homogeneity** | **Completeness** | **V Measure** | **ARI** | **NMI** | **FMI** |
|---|---|---|---|---|---|---|
| **Bigram** | 0.81 | 0.92 | 0.86 | 0.68 | 0.86 | 0.73 |
| **Trigram** | 0.81 | 0.91 | 0.86 | 0.68 | 0.86 | 0.73 |

## 5. Conclusion & Future Works

In this section a discussion of all the results in comparison to each other is provided, as well as the outcome of the newly implemented algorithm and its overall effectiveness. Future works, or changes that can be made to improve upon this project will also be presented. This will be done with the previously stated objective in mind, and will address how well the objective has been achieved.

Table 7. Evaluation of the All Clustering Algorithms on Bigram Features

| | **Homogeneity** | **Completeness** | **V Measure** | **ARI** | **NMI** | **FMI** |
|---|---|---|---|---|---|---|
| **K Means** | **0.83** | 0.86 | 0.85 | **0.70** | 0.85 | **0.73** |
| **Bregman** | 0.69 | 0.86 | 0.77 | 0.47 | 0.77 | 0.59 |
| **Consensus** | 0.81 | **0.92** | **0.86** | 0.68 | **0.86** | **0.73** |
| **Agglomerative** | 0.79 | 0.85 | 0.82 | 0.60 | 0.82 | 0.66 |
| **HDBScan** | 0.79 | 0.75 | 0.77 | 0.55 | 0.77 | 0.60 |

Table 8. Evaluation of the All Clustering Algorithms on Trigram Features

| | **Homogeneity** | **Completeness** | **V Measure** | **ARI** | **NMI** | **FMI** |
|---|---|---|---|---|---|---|
| **K Means** | 0.83 | 0.89 | **0.86** | 0.67 | **0.86** | 0.71 |
| **Bregman** | 0.70 | 0.80 | 0.75 | 0.48 | 0.75 | 0.56 |
| **Consensus** | 0.81 | **0.91** | **0.86** | **0.68** | **0.86** | **0.73** |
| **Agglomerative** | 0.80 | 0.88 | 0.84 | 0.66 | 0.84 | 0.71 |
| **HDBScan** | **0.84** | 0.79 | 0.81 | 0.62 | 0.81 | 0.67 |

For both table 7 and table 8, the highest scores for each metric has been bolded so as to easily distinguish which algorithms performed best overall. From these tables, it can be noted that with trigram features, the implemented Consensus Clustering algorithm performed the best out of the five tested algorithms. Similarly for trigram features, the same observation can be made; however in terms of the ARI and homogeneity metrics, the K Means algorithm performed slightly better for bigram features. For trigram features, the HDBScan algorithm performed slightly better for homogeneity. Therefore, as the consensus clustering algorithm is based upon the K Means algorithm, and includes both the Agglomerative and HDBScan algorithms, it can be seen that the stated goal has been accomplished. This

is noticeable with the implementation of the new algorithm that is more effective than the K Means, which it is based upon. So, through the development of a new Consensus Clustering algorithm, we have implemented a more effective algorithm, while also determining which algorithms are more effective than others when compared to their ground truths. In conclusion, a more effective algorithm has been implemented that is an innovation on multiple pre-existing algorithms.

It is important to note that the HDBScan algorithm did not perform as well as other algorithms because there is no predefined number of clusters input as a parameter. This is important, as the Consensus Clustering algorithm was not susceptible to this same dilemma thanks to the correspondence issue. This issue was effectively addressed by translating the labeling from HDBScan into terms of K Means.

In future, this study can be built upon for new implementations of the Consensus Clustering algorithm, or to even improve upon it. An improvement could be a version where the Consensus Clustering algorithm does not require the number of clusters as a parameter, such that the algorithm is more versatile. This could be accomplished by further utilizing the Silhouette coefficient for Silhouette analysis, where the optimal number of clusters can be determined through average Silhouette scores (Scikit Learn, 2017).

**References**


Banerjee, A., Merugu, S., Dhillon, I., & Ghosh, J. (2004). Clustering with Bregman divergences. *Proceedings of the 2004 SIAM International Conference on Data Mining*. https://doi.org/10.1137/1.9781611972740.22

Dabbura, I. (2020, August 10). *K-Means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks*. Towards Data Science. Retrieved December 12, 2021, from https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.

Eriksson, H. (2016). Clustering Generic Log Files Under Limited Data Assumptions. *KTH Royal Institute of Technology School of Computer Science and Communication*.

Gavin, B. (2018, July 27). *What is a log file (and how do I open one)?* How-To Geek. Retrieved December 12, 2021, from https://www.howtogeek.com/359463/what-is-a-log-file/.

HungarianAlgorithm.com. (2013). *The Hungarian Algorithm: An example*. An Assignment Problem solved using the Hungarian Algorithm - HungarianAlgorithm.com. Retrieved December 12, 2021, from https://www.hungarianalgorithm.com/examplehungarianalgorithm.php.

Kassambara, A. (2018, October 20). *Agglomerative hierarchical clustering*. Datanovia. Retrieved December 12, 2021, from https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/.

Leland, L., Healy, J., & Astels, S. (2016). *How HDBSCAN works*. How HDBSCAN Works - HDBScan 0.8.1 Documentation. Retrieved December 12, 2021, from https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html.

Ogbuagu, N. (2021). A Comparative Study of Clustering Algorithms for Device Logs.

Scikit Learn. (2017). *Selecting The Number of Clusters With Silhouette Analysis On KMeans Clustering*. Scikit. Retrieved December 12, 2021, from https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html.

Vega-Pons, S., & Ruiz-Shulcloper, J. (2011). A survey of Clustering Ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, *25*(03), 337–372. https://doi.org/10.1142/s0218001411008683

Yamamoto, H. (1997). Rate-distortion theory for the Shannon Cipher System. *Proceedings of 1994 IEEE International Symposium on Information Theory*, *43*(3). https://doi.org/10.1109/isit.1994.394669