NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

Document Clustering as an approach to template extraction

André Miguel Fernandes Rodrigues

Dissertation presented as partial requirement for obtaining the Master's degree in Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

DOCUMENT CLUSTERING AS AN APPROACH TO TEMPLATE EXTRACTION

by

André Miguel Fernandes Rodrigues

Dissertation presented as partial requirement for obtaining the Master's degree in Information Management, Specialization in Knowledge Management and Business Intelligence

Advisor: Professora Doutora Mariana Sá Correia Leite de Almeida

Co Advisor*:* Ricardo Costa Dias Rei

July 2021

ACKNOWLEDGEMENTS

I want to express my gratefulness to my advisors Mariana Almeida and Ricardo Rei, to Cleverly for the opportunity, to my friends, and to my family.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 873904.

ABSTRACT

A great part of customer support is done via the exchange of emails. As the number of emails exchanged daily is constantly increasing, companies need to find approaches to ensure its efficiency. One common strategy is the usage of template emails as an answer. These answers templates are usually found by a human agent through the repetitive usage of the same answer. In this work, we use a clustering approach to find these answer templates. Several clustering algorithms are researched in this work, with a focus on the k-means methodology, as well as other clustering components such as similarity measures and pre-processing steps. As we are dealing with text data, several text representation methods are also compared. Due to the peculiarity of the provided data, we are able to design methodologies to ensure the feasibility of this task and develop strategies to extract the answer templates from the clustering results.

KEYWORDS

Document Clustering; Similarity Measures; Text Representation; Template; Natural Language Processing

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Inteligence
Seq2Seq	Sequence-to-Sequence
RNN	Recurrent Neural Networks
DBSCAN	Density-based spatial clustering of applications with noise
ARI	Adjusted Rand Index
BoW	Bag-of-Words
TF-IDF	Term Frequency-Inverse Document Frequency
CBOW	Continuous Bag-of-Words
BERT	Bidirectional Encoder Representations from Transformers
KLD	Kullback-Leibler Divergence
SVM	Support Vector Machines
KNN	K nearest neighbors

1. INTRODUCTION

Ever since the world has entered the digital age, the number and quality of technologies of information and communication has been steadily increasing. One of the most known ways of communication is through email. Even after the appearance of social media, the usage of emails has been increasing. As this number continues to increase, so does the number of exchanged emails. Daily, this number is expected to be billions of emails.

Consequently, not only the average person but also companies receive and send numerous emails each day. Dealing with customers via email is a great component of a company's Customer Service. As such, it is vital to the company's success to be able to understand and extract valuable information from such data. While not the focus of this work, tasks such as Sentimental Analysis and Spam Detection are examples of an ever-increasing number of Text Mining techniques used on emails.

1.1. MOTIVATION

When dealing with Customer Service, attempting to receive and answer to many emails can be challenging. However, several efforts can be made to mitigate the situation. As most companies provide a service or a product, emails sent to them should, for the most part, be inquiries regarding said subjects. As such, common problems or questions regarding the company or its services would be presented in emails similar to each other. Therefore, its response would also be alike. By using the same answer repeatedly, instead of each employee tailoring its own version of the response, template emails are created to increase efficiency. Template emails are premade emails that contain the layout and core of a response email and may be missing the specific elements of the particular case (i.e., client name, process number, etc.). The creation of these templates can be time-consuming for a human agent. Therefore, having a system capable of identifying unknown templates from a group of emails would save time and money to companies.

1.2. GOAL AND CONTRIBUTIONS

The objective of this thesis is to develop a method capable of finding template candidates from a group of response emails to clients. Although the goal is to find new templates, some of the found emails can already be in use by the company.

One approach to discover templates is to group together similar emails and from such groupings extract potential templates. This grouping task is called Clustering. Since emails are text data, they must be represented in a manner so that they can be used in clustering algorithms. These algorithms would also need a method to detect whether two emails are similar in its representations. As such, this work needs to give a great deal of importance to these three components: the clustering, text similarity and text representation. Hence, this thesis's contributions are the following:

- Discover the best text representation to the given context.
- Experiment with text comparisons and choose the best one for this problem.
- Apply different clustering techniques and justify the best one.
- Develop a system capable of extracting useful templates from groups of emails
- Creation of auxiliary corpus specially annotated for this work.

The first three goals are intrinsically connected since they may depend on each other and together help achieve a better result for the fourth and major goal. The last contribution helps assess the quality of the original data as well as gain insights regarding text comparison in the given problem.

A real-life case is used in this work with the help of Cleverly. Cleverly is a company that uses artificial intelligence (AI) to help Customer Service platforms automatically label incoming messages considering its urgency and type of inquiry. This AI is also able to automate and help construct new responses. With the provided corpus by Cleverly, we will be able to verify the feasibility of this approach and contribute with some insights regarding the practicality of each component of the clustering methodology.

This document follows the according structure: chapter 2 presents the theoretical background, and chapter 3 the related work. In chapter 4 we introduce the corpora available and do some experiments. In chapter 5 we report and evaluate the approach. Chapter 6 is the conclusion where we conduct an overview of the entire research. Lastly in chapter 7, we talk about some limitations we had on our work and suggest some ideas for future work regarding this topic.

2. BACKGROUND

In this chapter, the core concepts behind this work are presented. From an introduction to the relevant Machine Learning, to examples of text representation and similarity measures.

According to (Mitchell, 1997), Machine Learning can be defined as "the study of computer algorithms that improve automatically through experience". It is a sub-field of artificial intelligence that aims to develop algorithms that learn from data, to extract information and identify patterns. (Shalev-Shwartz & Ben-David, 2014) state that depending on the nature of the interaction that occurs during the learning task between the learner and the environment, two key categories of machine learning algorithms can be distinguished: unsupervised machine learning and supervised machine learning.

2.1. SUPERVISED MACHINE LEARNING

Supervised Machine Learning belongs to the category of methods that aims to learn implicit relations between two different sets of variables. Therefore, the purpose of these algorithms is to from the learnt relationships from the supplied data make future predictions in unseen data.

Supervised Models can be divided into regression and classification models, depending on the domain of the variable we want to predict, named target variable. Regression models have a target variable composed of continuous variables of real values, while classification problems have a target variable of previously defined set of classes.

Although not the main focus of this project, some methods used rely on supervised Machine Learning and in particular depend on artificial Neural Networks. Artificial Neural Networks are a type of machine learning model inspired by the neurobiology understanding of the human brain. These models try to reproduce the processes happening inside biological nervous systems, particularly within a single neuron.

2.1.1. Perceptron

The most basic element in Neural Network architecture is the perceptron (Rosenblatt, 1958) and it is used as a building block in other, more complex Neural Network architectures. The Perceptron is formally represented by the equation:

$$Perceptron(x) = g(xW + b)$$

Where x is the input vector, W is a weight matrix, b a bias term and $g(z)$ is the activation function. The premise of the Perceptron is to apply a linear transformation to the input vector x via the weight matrix W and sum the bias terms b to the result. This output is then passed through the activation function $g(z)$ which depends on the given problem. Softmax, logistic (sigmoid) and hyperbolic tangent (tanh) are some of the most common activation functions. The softmax function is often used as the last layer of a neural network as it normalizes a vector into a vector of values between 0 and 1 which are often interpreted as probabilities.

To optimize the Perceptron, we can define a loss function and apply the Gradient Descent algorithm which is an iterative optimization algorithm for finding a function minimum. If we take the loss function

$L(\hat{y}, y; \theta_{model})$ as the computed difference between the model prediction $\hat{y}$ and the expected value y, taking W and b as parameters; we can define the gradient descent algorithm as follows:

Gradient Descent Optimization Algorithm

$\theta_{model} \leftarrow$ any point in the parameter space

while $L(\hat{y}, y; \theta_{model}) > \varepsilon$ **do**

for $\boldsymbol{w_i} \boldsymbol{\epsilon}\ \theta_{model}$ **do**

$$\mathrm{w}_{\boldsymbol{i}} \leftarrow \mathrm{w}_{\boldsymbol{i}} - \boldsymbol{\alpha} \frac{\boldsymbol{d}}{\boldsymbol{d}\, \boldsymbol{w_i}} L(\hat{y}, y; \theta_{model})$$

end for

end while

The α parameter indicates how much we want θ_{model} to change per update and is called the learning rate.

2.1.2. Multi-Layer Perceptron

A Multi-Layer Perceptron, or MLP, is a neural architecture comprised of multiple Perceptrons.

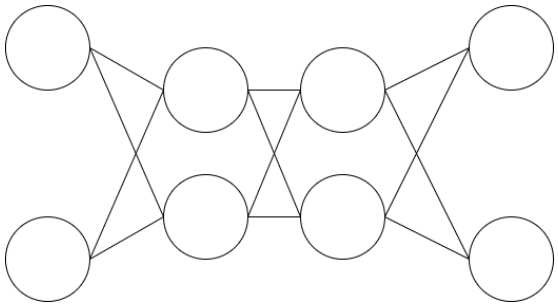

Figure 2.1: Representation of an MLP with two hidden layers.

This kind of architecture entails that while one Perceptron had only one layer of transformation, in MLPs there are more than one layer of transformations. The basic composition of MLPs are an input layer, an hidden layer and an output layer. Similarly to the single perceptron, there is an input which in turn is transformed by a set of weights and added to the bias term. This transformed input goes through activation functions which results in the layer output. This computed output is then used as an input to the following layer. This is repeated until the final output. As with the perceptron, MLP can be optimized by the use of the gradient descent algorithm.

2.1.3. Transfomers

Sequence-to-Sequence (Seq2Seq) models are neural network models that aim to transform a sequence of elements, like a sequence of words in a phrase, into another sequence. Initially introduced for machine translation by (Sutskever, Vinyals, & Le, 2014) seq2seq models are examples of encoder and decoder architectures. The aim of the encoder is to process the input sequence and map it into a dimension space. This abstract vector is then sent to the decoder which in turn decodes it into an output sequence. Recurrent Neural Networks (RNNs) are used as encoder/decoder. RNN's are Neural Networks that can process sequential data.

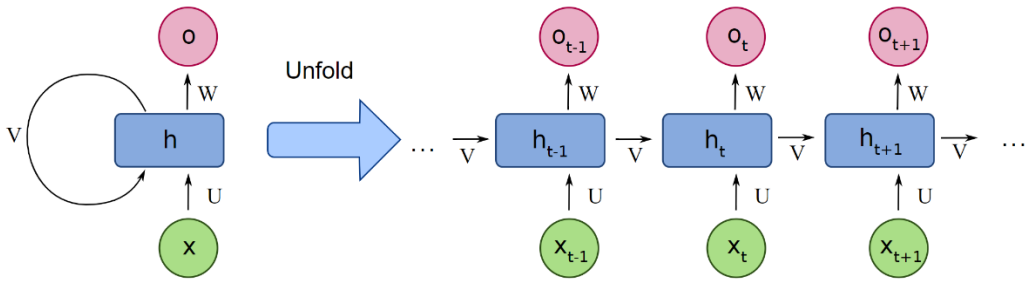

Figure 2.2: Graphical representation of an RNN. Taken from https://tinyurl.comm/4jkguspv

At every step, the RNN takes part of the sequence and a hidden state from the previous part of the sequence as input. The hidden state is then reupdated and used together with the next element of the sequence. This effectively makes it so that these internal structures have memory.

When using the encoder-decoder approach, all the information of the sequence must be compressed into a fixed-length vector. According to (Cho, Van Merriënboer, Bahdanau, & Bengio, 2014) this is problematic when using long sequences such as the task of translating long sentences. To address this problem, (Bahdanau, Cho, & Bengio, 2015) proposed an extension to the encoder-decoder model named attention layer which allows the decoder phase to focus on the most important information from the input sequence. The concept of the attention layer is to instead of the decoder reading only the last hidden state that comes out of the encoder, it reads all the hidden states that come from the encoder and applies to them a softmax function to determine which of them it pays more attention to.

Along the years, several deep learning models were introduced based on the previous concepts. These models used some variations of the encoder/decoder approach and different ways of constructing the attention layer such as presented by (Luong, Pham, & Manning, 2015). One of these models is the Transformers. This model was proposed by (Vaswani, et al., 2017) and can perform a wide variety of NLP-related tasks that can be framed as sequence-to-sequence problems such as machine translation and named entity recognition. It differs from previous models as it does not use Recurrent Neural Networks while still using the encoder-decoder approach and attributes great importance to attention-mechanism. Figure 2.3 illustrates the transformers architecture.

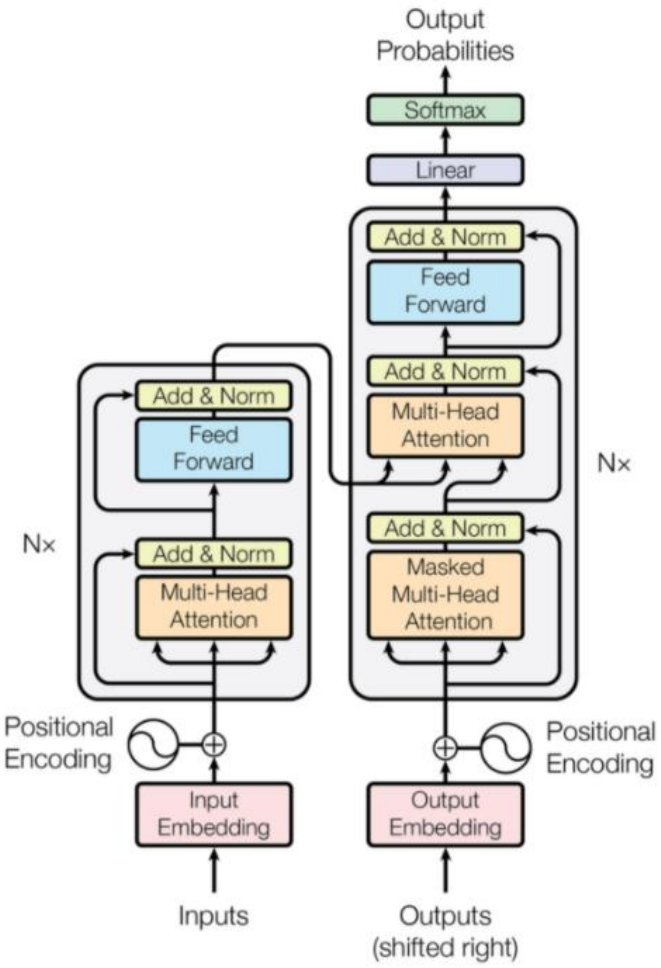

Figure 2.3: Graphical Representation of the Transformers model architecture. Taken from (Vaswani, et al., 2017).

Describing the model considered in the figure, the left block represents the encoder and the right block the decoder. While portraited as single module in the image, the encoder and the decoder components are composed of stacks of simpler encoder modules on top of each other multiple times, which is described by Nx in the figure. Each encoder module has two sub-layers. A multi-head attention layer, which is a module that runs through an attention mechanism several times in parallel, and a feedforward neural network. Apart from those two sub-layers, each decoder module also has a third sub-layer which performs multi-head attention over the output of the encoder component.

The Transformers architecture has been used as the basis of other methodologies which have been successfully applied to a wide variety of tasks achieving state-of-the-art results.

2.2. UNSUPERVISED MACHINE LEARNING

Unsupervised Machine Learning algorithms differ from Supervised ones in that the target variables are not known. Therefore, this learning consists in discovering a structure or relationship between the different inputs.

Although there are several methods of Unsupervised Machine Learning, the most known algorithms are related to Clustering which is the main focus of this work.

Clustering is the task of separating data into groups of similar points. (Kaufman & Rousseew, 1990) state it as "the art of finding groups in data". These resulting groups are called clusters. One goal when clustering is to make points belonging to a cluster similar between themselves (i.e., have a high intra-cluster similarity) and reduce the similarity between points from other clusters (i.e., have a low inter-cluster similarity). Similarity is a measure of the relationship between two points.

Clustering algorithms can be divided in two main categories: hard clustering and soft clustering. The difference between them is that while in hard clustering each point belongs solely to one cluster, in soft clustering a point can be part of more than one cluster.

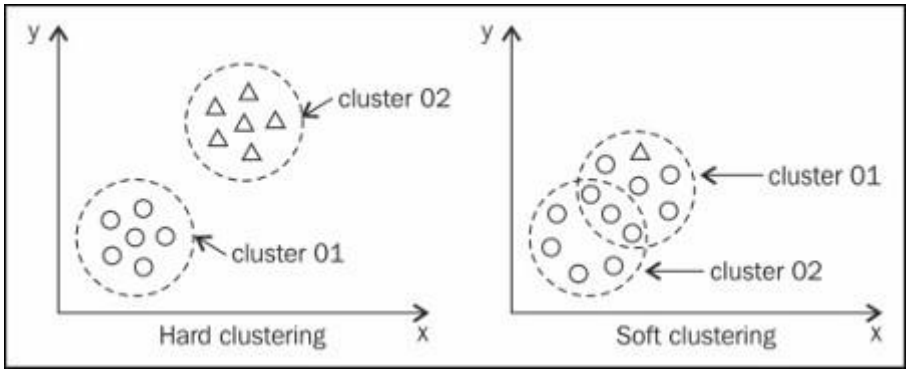

Figure 2.4: Graphical representation of hard clustering vs soft clustering. Taken from https://tinyurl.com/19vic4av

In this work, we will be focusing on hard clustering.

As mentioned before in Section 1.2, the clustering results depend on the representation of each point, the similarity measure and the clustering algorithm used (Jain, Murty, & Flynn, 1999). It is worthy to note that while literature offers a large portfolio of clustering algorithms, there is no universal methodology that applies perfectly to every case and dataset.

Three types of clustering algorithms can be distinguished, which will be discussed and given examples in this chapter:

- Partitioning Algorithms
- Hierarchical Algorithms
- Density-based Algorithms

While each category aims to have a clustering result that satisfies the similarity constraint, they all do it differently. The partition algorithm creates a flat partition of a specified number of clusters based on

a similarity measure. Hierarchical algorithms aim to create a hierarchical tree of clusters (dendrogram). And density-based algorithms as the name suggests, finds clusters by using the dense region of data points and utilizes the low-density regions as boundaries between clusters.

2.2.1. Partitioning Algorithms

Partition clustering algorithms aim to optimize a certain criterion function to separate (or partition) the data objects into a number of k clusters, based on a similarity measure. A common disadvantage of these types of algorithms is that, depending on the initialization, they can reach different solutions.

2.2.1.1. K-means

Perhaps the most well-known clustering algorithm, K-means was first used by Macqueen (1967). Its popularity is due to its simplicity and ability to be used for any type of problems. The idea behind k-means is that a cluster can be represented by a central point. This algorithm uses the concept of centroid which is the mean point of the elements of a cluster.

The k-means algorithm begins by setting k initial random centroids. Afterwards, each point is assigned to its closer centroid, based on a similarity measure usually being the squared Euclidean distance. The centroids are then recalculated by averaging all the points in each cluster, and the iteration begins anew. This algorithm keeps running until convergence is obtained which is when no reassignment of the centroid occurs or when the maximum number of iterations has occurred.

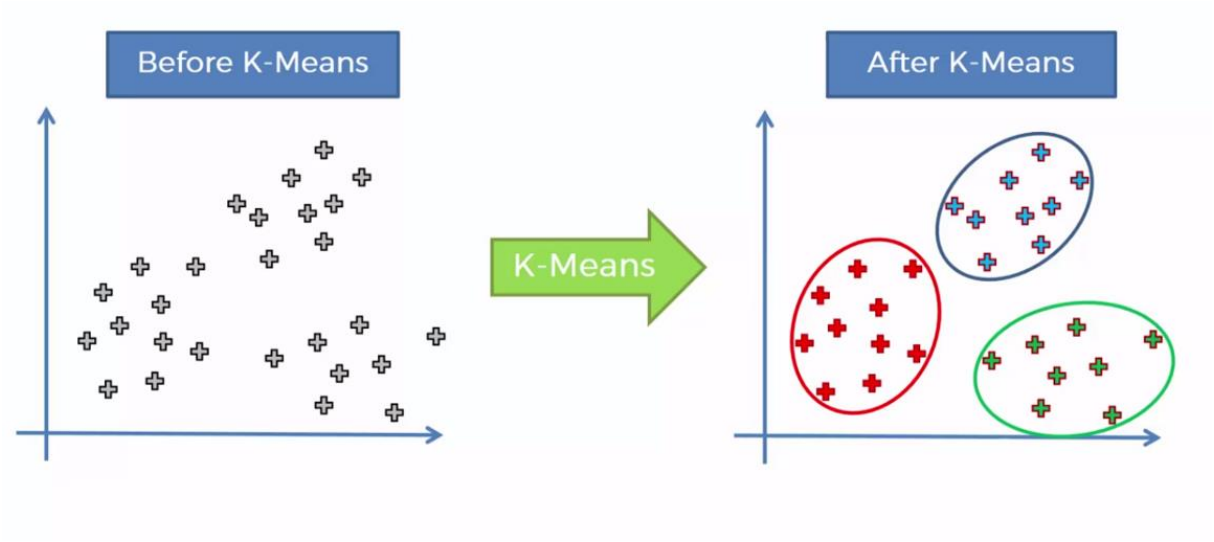

Figure 2.5: Example of K-means algorithm with k=3. Taken from https://tinyurl.com/3mxykbu9

The two main disadvantages of k-means are as follows: 1) the need to indicate the k number of clusters before the clustering (which may bring bad clustering accuracy). Outliers can also greatly disturb the computation of the centroid. 2) the fact that the initial cluster centroids may be a bad choice which in turn can get the centroid to be trapped in a local minimum.

To mitigate these disadvantages, there are some methods that can be applied, which will be discussed in this work.

2.2.1.2. K-medoids

K-medoids is a variation of k-means that instead of using the mean (centroid) uses the median (medoid). To note that in k-means the centroid rarely corresponds to an actual data element while the medoid in k-medoids must be an element of the data (for example a document in document clustering) which makes the interpretation easier. This feature of k-medoids might be great in this project as we want to extract from each cluster a real document that represents it. The k-medoids algorithm is also less sensitive to outliers and noise.

2.2.1.3. K-means++

K-means++ algorithm is a variant of k-means proposed by (Arthur & Vassilvitskii, 2007). This variation seeks to lessen its original's disadvantage of possible bad initialization by altering its initial conditions. Let $d(x)$ represent the minimal distance from data point x to a cluster center already chosen. The algorithm for center initialization is as follows:

1. Randomly choose one of the data points as a cluster center c_1.
2. For each data point x, determine $d(x)$.
3. Choose the next cluster center c_i, based on the x with highest probability $\frac{d(x)^2}{\sum_{x \epsilon X} d(x)^2}$
4. Repeat steps 2 and 3 until k centers have been selected.
5. Proceed with the standard k-means algorithm.

The K-means++ is as such designed to enhance the cluster center initialization for k-means.

2.2.1.4. Spherical K-means

In an effort to exploit the sparsity of text data (the type of data we will explore in this work) and drawing insights of its distribution in high-dimension spaces, (Dhillon & Modha, 2001) propose a variant of the well-known "Euclidean" k-means by using cosine similarity as its similarity measure.

Firstly, this algorithm has the assumption that the document vectors have been normalized to have unit L^2 norm (for each data point z, $\|z\| = 1$) which in turn means they can be represented as points on a high-dimension unit sphere (hence the name spherical k-means). As such, documents with different lengths but with equivalent subjects will be regarded as similar vector-wise.

The cosine similarity can be derived from the Euclidean dot product formula and is represented as such:

$$similarity = \cos(\theta) = \frac{A * B}{||A|| \, ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

However, if the element (for example documents) vectors are normalized, the cosine similarity can be represented simply as the dot product between the vectors.

The spherical k-means algorithm follows the same pattern as regular k-means with small differences:

1. Randomly select k document vectors as the concept vectors (which in regular k-means would be named centroid)
2. For each document vector x_i, compute the closest concept vector in cosine similarity and assign x_i to its cluster.
3. Compute the new concept vectors as the normalized mean in each cluster.
4. Repeat steps 2 and 3 until some stopping criteria is achieved.

2.2.2. Hierarchical Clustering

Distinctly from partition clustering methods, hierarchical clustering seeks to build a tree of cluster usually represented by a dendrogram. These methods can be further categorized as agglomerative or divisive depending on how the clustering tree is built. While the agglomerative approach begins with each point being an individual cluster and is progressively joined with its next most similar cluster in a bottom-up fashion until all points are in one cluster, the divisive approach begins with all the points together in one cluster and iteratively splits them until single points, in a top-down flow.

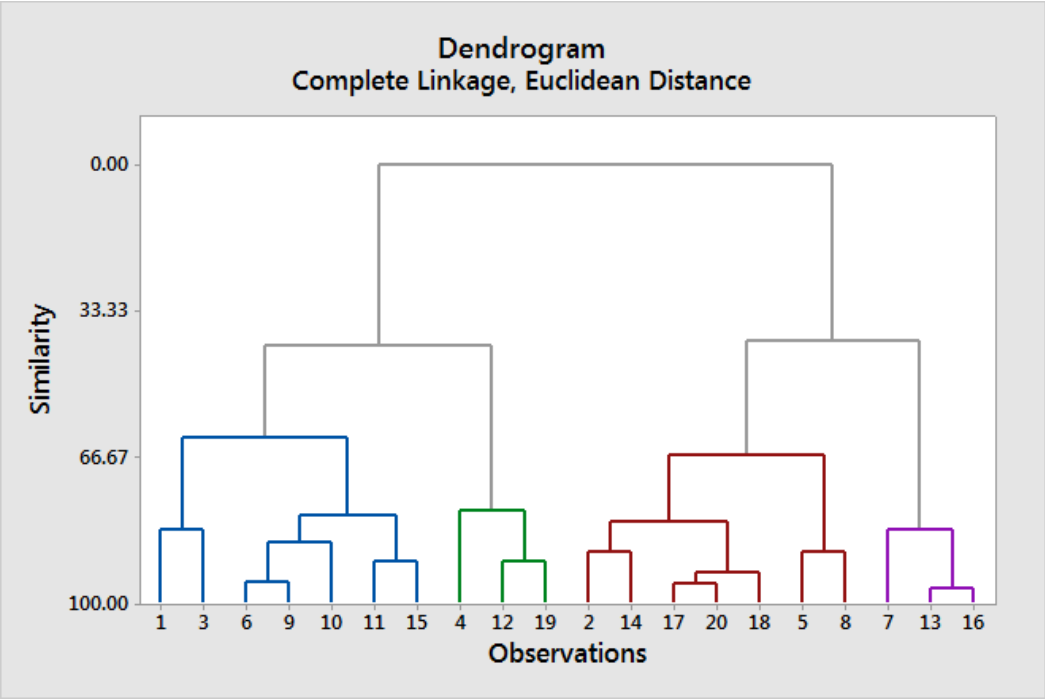

Figure 2.6: Example of a Dendrogram. Taken from https://tinyurl.com/cralh7qq

One main advantage over previous clustering algorithms is the fact that there is no need to specify the number of clusters. In contrast, these types of algorithms can have bad outputs if wrong merges or splits are made as those steps are irreversible.

Agglomerative hierarchical clustering algorithms are usually as follows:

1. Take each point as a single cluster.
2. Compute the similarity between each of the cluster to find the closest (most similar) pair.
3. Merge the similar cluster to create a new cluster.
4. Repeat steps 2) and 3) until there is only one cluster.

The most important step in this kind of algorithm is made between step 1 and 2 which is to choose how to calculate the similarity between two clusters. There are many ways to calculate this similarity between them such as: *min*, where the similarity between two clusters is the minimum distance between two points from different clusters; *max*, where the similarity is given by the maximum distance between two points in different clusters; or *average*, where the similarity is the calculated average of all the distances between all points from the first cluster with all the others from the second one.

Divisive hierarchical clustering works as the opposite of Agglomerative algorithms, where it begins with one cluster and keeps on splitting until clusters are single data points.

Bisecting k-means is an example of divisive hierarchical clustering. As the name entails, it is a variant of K-means. This method's main idea is to keep applying the k-means algorithm to each partition. The algorithm is as follows:

1. All documents are placed into one Cluster.
2. Pick a cluster to split.
3. Apply K-means with k=2. (Bisecting step)
4. Repeat steps 2 and 3 until the desired number of cluster k is achieved.

In step 2, there are several approaches to choose which cluster to split, it could be the one with the least overall similarity, the largest cluster, or a mixture of both criteria. According to (Steinbach, Karypis, & Kumvar, 2000) there are not significant differences in results between the criterions. They also conclude that the bisecting K-means technique is better than the standard K-means approach and is as good or better than the usual hierarchical algorithms.

2.2.3. Density-Based Clustering Algorithms

Density-based clustering algorithms instead of using proximity to group objects, are based on the notion of density. The core concept is to keep merging points with a cluster as long as the density remains high.

Density Based Spatial Clustering of Application with Noise or DBSCAN is the most well-known algorithm of this type. Like Hierarchical Clustering, it does not need the number of clusters as an input parameter. It has however two user defined parameters: the epsilon (ε) and the minimum number of sample M. Regarding ε, two points are considered neighbors if the distance between them is smaller than epsilon. M is the minimum number of neighbors a certain point must have to be categorized as a core point. There are three classifications for points: core, border, and outlier.

A core point has at least m points in its neighborhood (including itself). A border point does not have at least m points in its neighborhood but is the vicinity of a core point. Finally, an outlier or noise point is a point that does not belong to the neighborhood of a core point.

The standard DBSCAN algorithm starts with no pre-selected clusters and can be seen as:

1. Pick a random point that has not been assigned to a cluster and is not assigned as an outlier. Analyze the neighborhood according to M. If the point is a core point, it is a new cluster. If it is not, assign as an outlier and pick another point.

2. Add the neighborhood points of the core point to the cluster. Check the neighborhood points for core points and add its neighborhood to the cluster. Repeat until no more new points are added to the cluster. Previously assigned outliers may be now assigned as border points.

3. Repeat step 1 and 2 until every point has been visited and they are either part of a cluster or an outlier.

As the name states, this algorithm is resistant to noise and can handle outliers. However, it is does not handle high dimensional data too well, which is a problem in document clustering.

2.3. Cluster Evaluation

We have presented different algorithms and methods to be used in data clustering. However, as it is an unsupervised learning task, in real problems there are usually no a priori information about the dataset. This makes the validation of the clustering results a challenging phase when applying a clustering algorithm and often regarded as important as the clustering itself (Hassani & Seidl, 2017). One type of information not usually known is the number of clusters. Algorithms like K-means (section 2.2.1.1.) depend on such parameter and the search for the optimal k is a highly non-trivial task. Hence, many cluster validity methods have been introduced in literature to solve these problems and to evaluate the partitions of the data to represent its original structure.

Two main types of evaluation measures can be distinguished: external evaluation and internal evaluation. Along this subchapter we will be going through each of them, give some examples and state their advantages and disadvantages.

2.3.1. External Cluster Evaluation

External Validation Measures are based on previous knowledge about data. This usually means to compare the results of a clustering algorithm to a provided labeled partition of the data. Its intent is to measure how similar the cluster labels match a set of predefined classes deemed as the true clusters (commonly referred to as the gold standard).

2.3.1.1. F-measure and Rand Index

Evaluation measures used to usually evaluate classification methods can be applied to pattern recognition when the ground truth is known. The most common measures are Recall, Precision and its harmonic mean, F1-measure. These evaluation measures usually use the concepts of True Positive

(TP), True Negative (TN), False Positive (FP) and False Negative (FN) in its calculation and are calculated over pairs of points. Given two documents, they are considered as a TP decision if they are similar and were assigned to the same cluster. If the documents are not similar and were assigned to different clusters, they are considered a TN decision. On the other hand, if two not similar documents are assigned to the same cluster, it is a FP decision. Finally, a FN decision is when two similar documents are assigned to different clusters. By knowing the ground truth, we can ascertain if two documents are similar if they belong to the same class (or label). Table 2.1 illustrates these concepts.

Table 2.1: Decision Meaning

	Assigned Same Cluster	Assigned different clusters
Belong same class	TP	FN
Belong different classes	FP	TN

As we will be using these concepts in a clustering problem, the Precision and Recall can be written as the following:

$$Precision(i,j) = \frac{TP}{TP + FP} = \frac{nij}{ni}$$

$$Recall(i,j) = \frac{TP}{TP + FN} = \frac{nij}{nj}$$

Where n_{ij} is the number of elements of label i that are in cluster j; n_i is the number of elements in label i and n_j is the number of objects in cluster j.

As the F-Measure is the harmonic mean of the previous concepts, it is given by the following equation:

$$F(i,j) = \frac{2 * Recall(i,j) * Precision(i,j)}{Precision(i,j) + Recall(i,j)}$$

Label i has the F-measure equivalent to the maximum value at any cluster. The overall F-measure of the clustering is given by the weighted average of all the labels' F-measure:

$$F = \sum_i \frac{ni}{n} \max F(i,j)$$

One of the first clustering validation measures was the **Rand Index**, also known as accuracy. Presented by (Rand, 1971),it is calculated as the proportion of accurate assignments over all assignments. Using the previous notions, it is defined as:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

An improvement on this index was developed named Adjusted Rand Index (ARI). One of its advantages over its predecessor is its ability to compare partitions of the data with different number of clusters. Using the previous notions, it can be defined as (Santos & Embrechts, 2009):

$$ARI = \frac{\binom{n}{2}(TP + TN) - [(TP + FP)(TP + FN)(FN + TN)(FP + TN)]}{\binom{n}{2}^2 - [(TP + FP)(TP + FN) + (FN + TN)(FP + TN)]}$$

Greater values in these measures suggest higher quality of the clustering result.

2.3.1.2. Entropy and Purity

Entropy measures how spread a distribution is. If all clusters have only objects of a single label, the calculated entropy will be 0. As the variety of labels per cluster increases the worse the clustering is and the higher the entropy. Using the previous notation, the probability of a member of cluster j belonging to the label I is $pij = \frac{nij}{nj}$. The entropy of each cluster j is calculated as:

$$E_i = \sum_{i=1}^{L} pij * \log pij$$

Where L is the number of labels. The entropy of the cluster results is given as the weighted average of each cluster by the size of each cluster.

$$Entropy = \sum_{i=1}^{k} \frac{dj}{d} * Ej$$

Where k is the number of clusters, d the number of elements in the dataset used for clustering and d_j the size of the cluster j.

Purity measures how greatly clusters contain elements of a single label. The purity of a cluster is given by the highest probability p_{ij} in the given cluster j. Using previous terminology, it is represented as $purity_j = \max p_{ij}$. Similarly to the overall entropy, the overall purity of the clustering is given as:

$$Purity = \sum_{i=1}^{k} \frac{dj}{d} * purity_j$$

2.3.1.3. V-measure

Based on entropy, (Rosenberg & Hirschberg, 2007) presented V-measure which is an external cluster evaluation measure centered around the ideas of homogeneity and completeness.

The homogeneity criteria's basis is that clustering must assign only datapoints that belong to a single label belong to a single cluster. Assuming there are N data samples, C different classes, K different clusters and n_{ck} represents the number of elements that belong to class c and cluster k, the homogeneity can be defined as:

$$h = 1 - \frac{H(C, K)}{H(C)}$$

Where:

$$H(C,K) = -\sum_{k=1}^{K}\sum_{c=1}^{C}\frac{n_{ck}}{N}\log\left(\frac{n_{ck}}{\sum_{c=1}^{C} n_{ck}}\right)$$

And:

$$H(C) = -\sum_{c=1}^{C}\frac{\sum_{k=1}^{K} n_{ck}}{C}\log\left(\frac{\sum_{k=1}^{K} n_{ck}}{C}\right)$$

While both previous Purity and Entropy measures represent evaluations of homogeneity of clustering results, they do not address the completeness concept. According to (Rosenberg & Hirschberg, 2007), *all* elements that belong to a single class must be assigned to a single cluster so that completeness is achieved. Considering previous notations, completeness can be seen as:

$$c = 1 - \frac{H(K,C)}{H(K)}$$

Where:

$$H(K,C) = -\sum_{c=1}^{C}\sum_{k=1}^{K}\frac{n_{ck}}{N}\log\left(\frac{n_{ck}}{\sum_{k=1}^{K} n_{ck}}\right)$$

And:

$$H(K) = -\sum_{k=1}^{K}\frac{\sum_{c=1}^{C} n_{ck}}{C}\log\left(\frac{\sum_{C=1}^{C} n_{ck}}{C}\right)$$

The V-measure is the computed weighted harmonic mean of these two concepts and can be given as such:

$$V = \frac{(1+\beta) * h * c}{(\beta * h) + c}$$

Where β can be given different values to attribute different weights in the calculations.

2.3.2. Internal Quality Measures

As opposed to external quality measures, internal quality measures aim to assess how good a clustering result is using only features and intrinsic information from the data set (Hassani & Seidl, 2017), meaning without respect to external "gold" information regarding the correct clusters. These measures try to quantify previously shown cluster properties: cluster cohesion, meaning how similar are two objects in a cluster; and cluster separation, meaning how dissimilar are clusters from one another.

A commonly used measure is the **Sum of Squared Error** (SSE). It can be used to measure cohesion by comparing each element of a cluster to its centroid. With a similar concept, the separation of a cluster can be measured by comparing each centroid to the centroid of the dataset which we will call Between cluster sum of squares (BSS). They can be defined as follows:

$$SSE = \sum_{i} \sum_{x \epsilon C_i} (x - m_i)^2$$

$$BSS = \sum_{i} |C_i|(m - m_i)^2$$

Where C_i represents cluster i with $|C_i|$ being its size; $m_i$representing the centroid of cluster i, and m the centroid of the whole dataset.

One measure that combines both cohesion and separation ideas is the **Silhouette Coefficient**. It is a sample-based coefficient and is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a, b)}$$

Where $a(i)$ is the average distance between sample i and all other points in the same cluster, and $b(i)$ is the minimum average distance between sample i and points belonging to another cluster. The coefficient result ranges from -1 to 1 and we can infer the following: if $s(i)$ is close to 0 the sample is between two clusters. If close to -1 then the sample should be assigned to other clusters while if close to 1, then the sample likely belongs to the correct cluster. This coefficient can be further applied by computing the average for each cluster or for all the dataset, which can give some general insights when analyzing the clustering results.

Another measure that also considers both cohesion and separation properties is the **Davies-Bouldin Index** (DBI). Presented by (Davies & Bouldin, 1979) it aimed to "infer the appropriateness of data partitions" while "not depend(ant) on either the number of clusters analyzed nor the method of partitioning". This index uses the concept of cluster similarity between two clusters i and j defined as follows:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

Where s_i is the average distance between the cluster's centroid and each point belonging to it - also known as the cluster diameter; and d_{ij} is the distance between the centroids of cluster i and j. The DBI then takes the maximum cluster similarity for each cluster and averages it:

$$DBI = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$

As it takes the highest similarity of each cluster, the smaller the index, the more distinct the clusters are from each other, therefore the better the partition is.

The presented internal measures can be used as criterions in clustering related tasks. As they are non-dependent on external knowledge, these coefficients can, for example, be used to help find the optimal number of clusters when using the k-means algorithm. By conducting different iterations of k-means with different k, one can select the iteration with the better evaluation in the previous measures.

One drawback of using only internal criterion in cluster evaluation is that high scores may not reflect on a desired partition of the data.

2.4. TEXT REPRESENTATION

In the previous sections, several models and techniques have been presented. However, to be able to apply such algorithms to text data, we need to represent data in a vector form capable of being read by the mentioned models. Hence, in this section we explore different ways of representing text as vectors, be it in sparse or dense vectors. We finish this section by referring to some similarity measures, that although do not represent text, are the main input for clustering algorithms used in this work.

2.4.1. Sparse Models

One of the simplest approaches for representing text is the bag-of-words (BoW). Bag-of-words represents text by considering the occurrence of words within each document. If we consider V a fixed vocabulary (all words used in all documents) with a finite size n, we can generate vectors to represent documents in which each entry corresponds to a word w in V. Consequently, each document's representation is a vector with size n, where each entry is a weight indicating the importance of the corresponding word in the document.

The simplest way to compute theses weights is to consider binary values, where the weight is 1 if the word exists in the given document, or 0 if it does not. However, by attributing the same weight to all words present in the document, it will give the same importance to words that carry more semantic meaning than others in the document. We could consider how many times the term (or word) t appears in the document d and use it as weights. This is called term frequency and can be denoted as $tf(t, d)$. It is also noteworthy that it is more useful to have informative terms with the biggest weights, and a term that appears in every document is not informative (consider for example stop words that are generically frequent, such as "the", "a", etc.). As such, if we consider the notion of document frequency (DF) as the number of documents in the dataset that contain the term t, and we want to give the terms that appear in fewer documents higher weight, we can define the notion inverse document frequency of a term as:

$$idf(t, D) = log \frac{N}{DF}$$

Where N is the number of documents in the dataset.

TF-IDF (Term Frequency – Inverse Document Frequency) is one of the most popular term-weighting schemes that, as the name implies, combines the term-frequency and the inverse document frequency, as follows:

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

This type of representation intuitively attributes a large weight to relevant terms that appear frequently in the document but scarcely across the dataset. This approach gives a vector representation able to check similarities between two documents. However, as it happens with the BoW approach, it does not express complex semantic features.

2.4.2. Dense Models

The other approach we will use in this project is the representation of text into an embedding space by using neural networks. These neural networks aim to represent words that appear in the same context as similar. They also aim to reduce the high dimensionality problem that the previous approaches suffer from.

One of the most used methods is Word2Vec. Proposed by (Mikolov, Chen, Corrado, & Dean, 2013) it is an unsupervised method capable of representing word embeddings. In (Mikolov, Chen, Corrado, & Dean, 2013) two model architectures are presented: The Continuous Bag-of-Word model (CBOW) and the Skip-gram Model. The CBOW model is obtained by predicting a target word w_t based on the ith word before and after it in the given sentence. Its objective function can be defined as:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t | w_{t-i}, w_{t-i+1}, \dots, w_{t+1}, w_{t+2}, \dots, w_{t+i})$$

The other method, Skip-gram, works in the inverse as it from the center word, tries to predict the surrounding words. Its objective function is as follows:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \sum_{-i \le t \le i, \neq 0} \log\ p(w_{t+1} | w_t)$$

This and similar models can capture word embeddings with semantic and syntactic meaning which in turn have some interesting properties like word arithmetic's (the representation of the words queen – woman + man = king). Nevertheless, these embeddings can fall short since words can have different definitions depending on the context (word polysemy). One model capable of producing such contextual embeddings is BERT.

2.4.3. BERT

BERT, or Bidirectional Encoder Representation from Transformers is a language representation model developed by (Devlin, Chang, Lee, & Toutanova, 2018) and it is designed to pre-train deep bidirectional representations of words from unlabeled text. BERT differs from other methods, such as ELMo models (Peters, et al., 2018), as it combines two methods: it is bidirectional and uses the Transformers architecture mentioned in section 2.1.3.

As the name entails, its architecture can be represented as a stack of identical Transformer Encoder layers with each layer having two types of sublayer: a multi-head self-attention mechanism and a feed-

forward network (similarly to 2.1). In (Devlin, Chang, Lee, & Toutanova, 2018) two versions of BERT are presented:

- BERT-base with L=12, H=768 and A=12 with total parameters of 110M
- BERT-large with L=24, H=1024 and A=16 with total parameters of 340M

Where L is the number of layers (Transformer blocks), the hidden size is H, and the number of self-attention heads is A.

To make the BERT model able to be used in a variety of downstream tasks, its input representation can represent both a single sentence and a pair of sentences in one token sequence. In Figure x we can see a representation of the transformations made to the input to obtain embeddings to pass to the model.

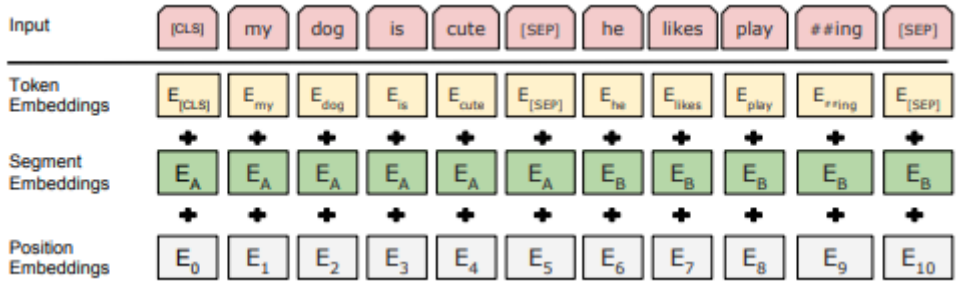

Figure 2.7: BERT input representation. Taken from (Devlin, Chang, Lee, & Toutanova, 2018)

Before calculating the Input Embeddings, each input sequence is preceded by an [CLS] token. This token is used in classification tasks as its corresponding final hidden state is used as the aggregate sequence representation. It is otherwise ignored in non-classification tasks. Also, pairs of sentences can be packed into one sequence having a special token [SEP] used a delimiter. The Token Embedding is the representation of each word using WordPiece Embeddings (Wu, et al., 2016). If dealing with a pair of sentences, the Segment Embedding represents whether a token belongs to sentence A or sentence B. The Positional Embedding identifiers the position of the token in the inputted sequence. The sum of these three embeddings is the Input Embedding used as input to the BERT model.

The BERT model is pre-trained on two tasks: masked language modeling (MLM) and next sentence prediction (NSP). MLM is the task of predicting input tokens that are randomly masked. In BERT MLM consists of randomly masking 15% of the inputted tokens and using the last hidden state of the model to predict the masked token. This enables BERT to learn bidirectional representations. BERT is also trained in the NSP task, which consists of predicting if a certain sentence B is the following sentence to a certain sentence A in the original corpus. This helps to understand the relationship between two sentences which it not directly captured by language modeling and is very important for several downstream tasks such as question-answering. The corpus to pre-train BERT was the BookCorpus (800M words) (Zhu, et al., 2015) and the English Wikipedia (2,500M words).

This BERT pre-trained model can be further fine-tuned to be used in several downstream NLP tasks such as Classification Tasks and Question Answering tasks. Besides fine-tuning, there is also the feature-based approach. BERT can be used to generate contextualized embeddings by extracting the outputs from one or more hidden layers. In (Devlin, Chang, Lee, & Toutanova, 2018) the BERT model is used in several NLP tasks where it achieves state-of-the-art-results.

It is worthwhile to mention that since the appearance of BERT, several BERT-based models such as RoBerTa (Liu, et al., 2019) and language variations like CamemBERT (Martin, et al., 2019) have emerged.

2.5. Similarity Measures and Text Comparison

In this chapter we will be presenting distance and similarity measures that can be used in most document clustering algorithms. Also, apart from comparing documents using its document representation, there are other ways of analyzing and comparing text without having to represent it in vector form. Most of these methods involve using distances that use words or characters as the basis.

2.5.1. Similarity Measures

As mentioned before, most clustering methods depend on similarity (or distance) measure.

When we think of distance, Euclidean distance is what we usually think about. For any given two documents a and b, represented as vectors with n dimensions, the Euclidean distance is as follows:

$$d(a.b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Another well-known distance is the Manhattan distance is defined as the sum of the absolute difference across all dimensions. It defined as such:

$$d(a,b) = \sum_{i=1}^{n}|a_i - b_i|$$

Due to their simplicity these distances are popularly used. However, in the context of document clustering, they have one big disadvantage. If two documents which are similar in context but greatly vary in size, they would be deemed different according to the previous distances. The cosine similarity measure that has been mentioned before (in 2.2.1.4.) can measure how similar the documents are regardless of their size.

$$similarity = \cos(\theta) = \frac{A * B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

2.5.2. Text distances

The previously shown metrics require documents to be represented as vectors. Some distances do not require such step and can be applied to text data.

Edit Distance, also known as Levenshtein Distance measures the similarity between two strings. This distance is the minimum number operations (insertions, deletions, or substitutions) used to transform one string into the other. For example, the edit distance between the words “Money” and “Monkey” is 1 as the minimum number of operations to transform one into the other is to delete the “k” character. This distance is commonly used in plagiarism detection and spell checking.

Another measure to compare documents is the **Jaccard Similarity Coefficient** or Jaccard Index. Jaccard Index measures similarity as the intersection divided by the union of the objects. When using this measure with text documents, the Jaccard Coefficient compares the sum of shared words to the sum of words that appear in either document. The definition is as follows:

$$Jaccard(A,B)\frac{|A \cap B|}{|A \cup B|}$$

The Jaccard Distance measures the dissimilarity between both documents, it is obtained by subtracting the Jaccard coefficient from 1.

3. RELATED WORK

In this chapter, we start by presenting works that compare components of document clustering such as similarity measures and clustering algorithms. Then we present BERT-based models that can be used in clustering. Finally, we present works that apply clustering methods in customer support related tasks.

3.1. CLUSTERING METHODOLOGY

The process of Clustering depends on a multitude of factors. Some were already referenced in this work, while others will be discussed in this chapter. Although it would be ideal, it is not possible to test every possible combination of factors. The following works aim to compare some of these factors and analyze its impact on the results.

One of the main factors in Document Clustering is the similarity measure used when comparing two different documents. (Huang, 2008) attempts to compare and analyze the impact of different similarity/distance measures when clustering. Firstly, each document is pre-processed following three strategies: stop words are removed; stemming is applied and terms that appear below a certain threshold are also removed. The result of this pre-processing is transformed into vector form using the TFIDF approach mentioned in section 2.4.1. Secondly, the clustering algorithm used is k-means. There are seven datasets used with the number of documents ranging from 878 to 18828 and the number of classes from 4 to 20. When applying the k-means algorithm, the selected k is the same as the number of classes as it is assumed that the number of clusters is consistent with the manually created classes.

Then, the work compares five similarity metrics: Euclidean Distance, Cosine Similarity, Jaccard Coefficient, Pearson Correlation Coefficient and Averaged Kullback-Leibler Divergence (KLD). As such, for each dataset of the seven, the five measures are used in the clustering algorithm, resulting in 35 experiments. The clustering results are then evaluated using the entropy and purity evaluation measures.

According to the results, the Jaccard and Pearson measures have better purity while KLD has better entropy. Apart from the Euclidean Distance Measure, which is the worst in every case, the author concludes that all Similarity Measures are comparatively effective when using partitional document clustering as the results are overall similar.

The selection of the Clustering Algorithm is another relevant step in this type of works. The following works aim to compare different document clustering algorithms to select the best one given the dataset.

In (Ariff, Bakar, & Rahman, 2018),the data used consists of sport articles from a website. There are sixty articles from four different sports. The author follows a pre-processing methodology like previous works as stop words are removed and stemming is applied to the documents. The three used clustering techniques are: Agglomerative Hierarchical Clustering, K-means Clustering and K-medoids Clustering. Another technique meant to be evaluated in this paper is the usage of only the most significant words as variables. As such, two words that are bias to each type of sport as selected.

Each clustering technique is then applied to the dataset and evaluated in efficiency rate and its calculated mean silhouette index. When using only the significant words, the k-medoids algorithm has the best efficiency rate while k-means has the best silhouette index. By adding the general words to the representation of the documents and evaluating the results, the hierarchical clustering has better results in both evaluation measures. The author concludes that while the k-means methods show the most significant silhouette index, it is sensitive to general words and as such, the hierarchical clustering method is deemed to be the better text clustering method as it achieves great results while not being sensitive to the addition of general words.

The purpose of (Balabantaray, Sarma, & Jha, 2015) was to compare the k-medoids and k-means clustering algorithms. 100 documents equally distributed between 5 domains were pre-processed. Tokenization, stop-words removal and TF-IDF transformations were applied to the data. The k-means and k-medoids algorithm were then applied and evaluated with an efficiency rate. It was observed that the k-means algorithm yields better result than the k-medoids.

3.2. BERT-BASED MODELS

After the release of BERT, various BERT-based models have emerged. One of them is Sentence-BERT (Reimers & Gurevych, 2019). Sentence-BERT or SBERT is a modification of the BERT network that uses Siamese structure and triplet networks to derive semantically meaningful sentence embeddings (meaning that semantically similar sentences are closer in vector space). This enabled BERT to be used in tasks which were not applicable for BERT such as clustering and large-scale semantic similarity comparison. In previous chapters, we stated that in document clustering, a common method is to map each sentence to a vector space. The clustering algorithm would then group close vectors as semantically similar sentences. Using regular BERT, a way to get this vector space embeddings is to input individual sentences into BERT and use the output layer as embeddings. In an effort to further increase the quality of these embeddings, SBERT was developed.

The architecture of SBERT is as follows. Firstly, SBERT adds a pooling layer to the output of BERT to create a fixed size sentence embedding. Several strategies such as using the output of the [CLS] token and the mean all output vectors were experimented. Secondly, to fine-tune BERT, SBERT uses a Siamese neural network that uses the same weights while working in parallel on two different input vectors so that the produced embeddings can be compared. This network objective function depends on the available training data. In the paper, they present three different objective functions: the classification objective Function, where the two sentence embeddings are concatenated and multiplied by a trainable weight and the cross-entropy is then optimized; the regression objective function, where the cosine similarity of the two sentence embeddings is computed and the mean-squared-error loss is optimized; and the triplet objective function, where given the sentence a, a sentence p and a sentence n with positive and negative intent regarding a, the triplet function tunes the network so that the distance between a and n is greater than a and p.

SBERT is trained on the combination of the SNLI and the Multi-Genre NLI dataset. Together they add up to one million sentence pairs annotated with the following labels: contradiction, entailment and neutral. This SBERT version is fine-tuned with the Classification Objective Function.

Throughout the paper, SBERT is evaluated in different tasks against other models specifically such as Universal Sentence Encoder and BERT itself. SBERT is able to achieve improvements over other state-

of-art sentence embeddings methods in tasks such as Semantic Textual Similarity (STS). In these STS tasks, the dataset provides a label between 0 and 5 on the semantic relatedness of sentence pairs. When checking the correlation between the cosine similarity of sentence representations and the gold labels of various STS the improvement is quite significant regarding other models.

Apart from SBERT using BERT as the base of the model, the authors also use RoBERTa (Liu, et al., 2019) as the base transformer encoder, creating the SRoBERTa model. According to them, using RoBERTa instead of BERT did not yield a significant improvement in the experiments.

Another great advantage stated by the authors is the fact that SBERT is computationally efficient, being able to compute sentence embeddings faster than other methods which will be relevant in this work.

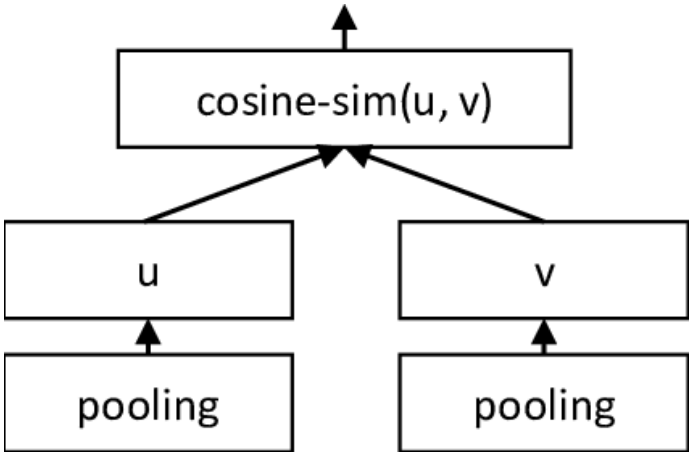

Figure 3.1 - SBERT architecture at inference, taken from (Reimers & Gurevych, 2019)

3.3. Document Clustering

The following works have clustering as a major component in their methodology. They present different strategies where document clustering is relevant and served as inspiration to the present work.

3.3.1. Spam Filter

Emails have become a target of spam attacks due to its popularity and necessity. As such, a need to develop more complex spam filtering methods to counter these attacks has started to grow. The following works applied different algorithms in different ways.

Document clustering has been also applied to detect spam emails. That is the case of (Sasaki & Shinnou, 2005), that uses the Ling-Spam corpus with 481 spam messages out of 2893 emails. After representing the document in a vector space, the email set is divided into k groups by the spherical k-means algorithm. Then, for each created cluster, a label is assigned. If the ratio of spam email to all email in the cluster is higher than 70%, the cluster is considered spam. Centroid vectors for each cluster are calculated and used as the cluster representation. New emails are then compared to each centroid via cosine similarity. The new email gains the same label as its most relevant cluster.

The authors also evaluate the impact of pre-processing techniques such as lemmatization and stop words removal, as well as the representation of the documents via Term Frequency or TF-IDF is experimented. Classification methods such as Support Vector Machines (SVM) are also used for comparison. All these experiments are evaluated via the precision of spam and non-spam assignment. While SVM results are higher, the various experiments using this clustering method achieve approximately similar results as most variations of the experiment achieve near 100% precision for both spam and non-spam. The authors conclude that is it as effective method for detecting spam.

3.3.2. Template Finder

In (Mota, Martins, & Coheur, 2013) the aim is to build an Answer Template Retrieval System. This system retrieves a reply from a corpus of email/reply pairs to be used by a human agent as the core to the response of a new email. As the available corpus did not have annotations, such as labeling of the email/reply pairs, the system must be able to perform another task apart from the answer retrieval. This task is the identification of similar answers to be used as templates. To perform such task, a cluster-based methodology was considered. The idea consists of identifying similar answers using a modified version of Jaccard Distance created for this problem named Edit Percentage. This measure dictates how much of a given email must be edited to be transformed into another one. The authors consider two emails to be in the same cluster if the edit Percentage is at most 10%. Given emails A and B which have an edit percentage lower than 10% to email B, the authors considered A and B to be in the same cluster even if their edit percentage is higher than 10%. Clusters with only one element were removed. From the 12k email/reply pairs, 1150 clusters were found. The authors then defined the retrieval task as a classification problem and as such tested several algorithms (Logistic Regression, k nearest neighbors (KNN), and others) with a 5-fold cross validation. Due to the good results and efficiency of KNN, further experimentation was done with such algorithm. Using the precision@10 metric (the right cluster must be in a list of 10 possibilities), the best result was 62%.

4. CORPUS AND DATA ANALYSIS

In this chapter we will present the corpus, have an in-depth analysis, and explain all the processing stages applied to it. Due to the nature of the dataset, some auxiliary corpora are created and used in some validation tasks.

4.1. CLEVERLY CORPUS

The main goal of the thesis is to study unsupervised machine learning methods to discover new email templates. For that purpose, we used a proprietary corpus of costumer support that was provided by Cleverly after being anonymized. The corpus comprises information regarding emails as well as a set of existing templates of frequent replies.

The variables are as follows: Title, Text_Question and Text_Answer which represent the title of the email, the content of the sent email and the content of the response email, respectively. The Template_Number variable which indicates which template has been used in this answer and is a null value if no template was assigned. The Template_Title variable represents the title of the template and can be seen as a small summary of the text. Template_Text is the core text of the template and used as basis in the Text_Answer response.

In 1.1 we mentioned that one of the strategies to enhance customer support would be the usage of templates, we also stated some companies may already be using templates and this is one of those cases.

When analyzing the data, some inconsistencies and problems were found, therefore the following transformations and filtering were done: some emails were either blank or had non impactful text (emails comprised of only a link or a word or were incomplete) and so emails with less than 20 words were removed; when looking at the Template_Number variable, some emails had more than one template assigned to it. This could have different meanings, like the templates used were different versions of the same answer or the answer was built out of more than one template. As we want to find new templates, entries that had more than one template assigned to it were removed. The inconsistency where Template variable being True while having no assigned Template_Number was also addressed. This leaves us with a dataset of 54723 emails.

Nº of Emails	Nº of Answers without Template	Nº of Answers with Template	Nº of Different Templates
54723	29937	24786	721

Table 4.1: Corpus Statistics

As represented in Table 4.1, around half of the dataset has an assigned template. This does not necessarily mean no templates were used on emails without an assigned template. They could just not have been documented. However, we know for certain that emails with an assigned template were written with it as its basis.

4.2. Created Corpora

There exist several methods to evaluate clustering results (section 2.3). While some methods depend on the intra and inter distance between clusters, others depend on the ground truth. In the present work, we believe that we cannot take the emails with assigned templates as a perfect ground truth. Nevertheless, we can however use a subgroup of emails to use metrics that rely on ground truth. There is also the need to evaluate the presented similarity measures in the given problem and understand if they achieve conclusions similar to the human agent. As such, we created two new corpora.

4.2.1. Silver Corpus

This development corpus is comprised only by the emails that have an assigned template. This new dataset was dubbed Silver Dataset (allusion to gold standard) and consists of the 24786 emails with 721 different templates.

This corpus will be used mainly to validate results.

4.2.2. Annotated Pairs of Templates

Templates are created from the constant usage of similar emails. They could be found through luck and repetition or through the usage of similarity metrics to connect similar emails. To be able to evaluate the veracity of this metrics and complement this work, we created a dataset based on our notion of similarity.

Due to the great number of emails in the dataset, if a human agent decided to compare every pair of emails, besides unfeasible, most of these paired emails would be extremely different from one another. By using templates, the number of possible comparisons is greatly reduced and as such a more notable range of differences is allowed. Therefore, the basis of this new corpus is the list of templates. Since we have 721 templates, there are 259560 possible pair combinations. We selected 146 random comparisons (i.e., 146 different pairs of templates) and gave them a score for how similar its texts are to each other. This score goes from 0 to 4 and its criteria is as follows:

0	Completely different Templates
1	Not very Similar Templates
2	Related Templates
3	Very related Templates but differ in a small detail
4	Equal Templates

Table 4.2: Annotation criteria

As this corpus was created by one person, these annotations cannot be seen as the truth and/or perfect. However, they can be used to evaluate the quality of similarity measures. The score distribution can be seen in figure 4.1. Templates are mostly different from one another as they tend to respond to different subjects. This can be seen in the distribution as most of the pairs were not related at all. On the other hand, the level 4 of similarity was granted to 14% of the pairs, meaning that in the opinion of the evaluation agent, some templates are essentially the same and could be merged into one.

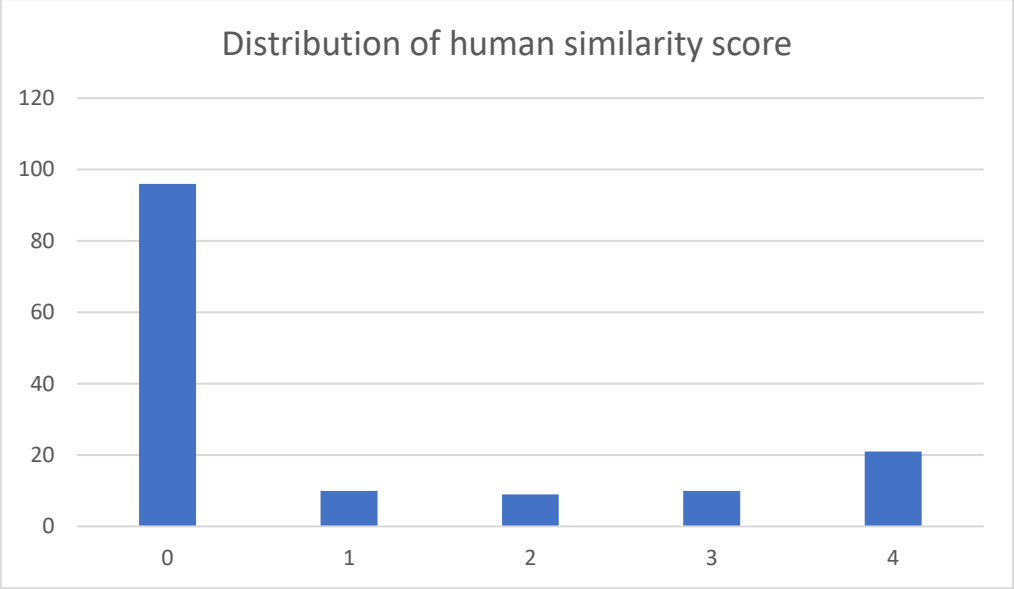

Figure 4.1: Distribution of human similarity score in auxiliary corpus

4.3. Evaluation Tasks

In the present work, we know there at least 721 clusters, which are represented by the known templates. We have however no knowledge of how many templates are in the remaining emails. If the number of templates is directly proportional to the number of emails, and since about half of our dataset has a template, we could be looking at doubling our number of known templates. On the other hand, this number could be small, and most of the unlabeled emails belong to an already known template. This issue is especially relevant to algorithms like k-means where we need to input the number of clusters to find. To further increase our familiarity with the presented measures and algorithms we perform different experiments on the data to gain insights for the problem at hand. As such, we will create tasks with the goal of finding the most appropriate combination of metrics and methods to use in our final approach.

4.3.1. Tasks and Experiments

Throughout chapters 2 and 3 we presented different ways of representing and comparing text. We chose four of them to compare and analyze. The first one is TF-IDF due to its simplicity and popularity in other works. In section 2.4, contextualized word embeddings were presented, and great importance was given to the BERT model. In section 3.2, a faster, more efficient model than BERT in some similarity tasks was presented. According to (Reimers & Gurevych, 2019), this SBERT model can create sentence

embeddings which are comparable with cosine similarity. Most works covered in section 3 use cosine similarity as the similarity measure. We decided to use cosine similarity for comparing both TF-IDF and SBERT text representations. Finally, we chose two other text comparison techniques, Edit distance and Jaccard distance to evaluate its effectiveness in the given problem. For the sake of clarity, when referring to TF-IDF and SBERT metrics, we are referring to the usage of cosine similarity jointly with the TF-IDF and SBERT representations.

4.3.1.1. Grouping Task

The Silver corpus was created from the provided Email Corpus as an effort to have a representation closer to reality. While not a perfect representation, it is as close as possible to the ground truth. To better understand if this corpus can be used as an accurate representation of reality, we created the following algorithm on the basis that a given email has greater similarity to emails belonging to the same template than to emails belonging to different templates. The algorithm is as follows:

- Step 1: Choose a group of emails represented by a unique template which has not been analyzed yet.
- Step 2: Pick a random element e in that group that has not been chosen.
- Step 3: Pick another random element a in that group, and a random element b outside the group.
- Step 4: Use the similarity measure to compare the element chosen in step 2 with the elements chosen in step 3.
- Step 5: According to the chosen measure, if e and a and more similar (higher value) than e and b, add 1 to the count of correct grouping.
- Step 6: Repeat steps 3-5 until all elements of the chosen group have been considered.
- Step 7: Repeat steps 1-6 until all groups have been analyzed.
- Step 8: Compute the score of correct grouping by dividing the count of correct grouping by the number of possible groupings.

With this algorithm, we can evaluate how well the four chosen metrics and representations accurately compare text in the given problem. By obtaining a high result in this experiment, we can assume the Silver data is correctly built and the metrics used can be used when clustering. To further analyze the viability of this Silver Data, we computed the Average Similarity between all groups as well as lowest and highest similarity in the given metric. This may provide further understanding of which metric to use.

Metric	Grouping Task	Average Sim
TFIDF	0.9952	0.228
SBERT	0.995	0.651
Edit Distance	0.98	0.264
Jaccard Distance	1	0.652

Table 4.3: Grouping Task Evaluation

Table 4.3 represents the above-mentioned experiments. The grouping task presents fantastic results to all different metrics, with the Jaccard Distance stating that the grouping is perfect. Although the other results do not have an objectively better metric, the TF-IDF and Edit Distance metrics may be used more intuitively by humans as they have bigger interval of similarity. We conclude that the Silver Data has great viability and will be used in experiments to test the effectiveness of other components of the Clustering process.

4.3.1.2. Relationship between Metrics

We have ascertained that the Silver data is a good representation of the ground truth, and the metrics can be used to compare emails. In the next experiment we want to test if the similarity seen by the metrics is close to that of a human agent. Meaning that the relationship and comparison of text with the given metrics should have similar results to those of a human agent.

The corpus created in section 4.2.2 should help analyze such relationships as it provides an intuitive approach of comparing the metrics with our previously labeled similarity score. We measured the Spearman correlation between the values given by the human agent and each of the measures in the given pairing.

Correlation	TFIDF	SBERT	Edit Distance	Jaccard
TFIDF				
SBERT	0.49			
Edit Distance	0.156	0.202		
Jaccard	0.159	0.216	0.441	
Human Score	0.822	0.812	0.801	0.783

Table 4.4: Correlation between metrics

In Table 4.4 we present such results. To note while the metrics have low correlation between themselves, they all have high correlation with the human score metric.

When considering the two previous tasks regarding metrics, we concluded they do not have a significant difference between one another. From this point on we decided to use only TF-IDF and SBERT metrics for future experiments. The reason being Jaccard distance has the lowest correlation to the human similarity score while Edit distance takes much longer than the other metrics to compute. The remaining methods can be used to evaluate if contextual word embeddings like SBERT are better in cases like the present one when compared to more simple text representation methods such as TF-IDF.

4.3.2. Text Pre-processing and Algorithm Evaluation

In the previous sections, different approaches were conducted to find differences in the chosen metrics. Due to related work presented in chapter 3 and to previous experiments, we have established that there are only minor differences in the presented similarity measures. In this subsection, we aim to select the optimal method for each of the other components.

We have tested the viability of the Silver Dataset as the real representation of the template grouping and determined it was a good representation. This has two consequences: firstly, it enables the use of External Quality Measures (section 2.3.1.) for clustering results since we have a representation of the original data structure (or ground truth); secondly, a rare case of clustering problem where the exact number of clusters is known is presented. In consequence, we can test different combinations of text representation, similarity measures and document pre-processing techniques in an objectively easy to evaluate environment. While this will not directly achieve the objective of finding new templates in the data, we are under the assumption that if we find a clustering methodology capable of having high evaluation in this sub dataset, the same methodology will translate comparable results when clustering with all the data.

Some Clustering Algorithms may use the number of clusters as an optional parameter; however, the k-means family of clustering algorithms need only the number of clusters to be executed. With one of its main issues solved, we can make use of this unique case to figure out the optimal type of k-means algorithm to be used in the final approach where the only issue would be finding the optimal number of clusters.

Since we are having this approach to decrease the number of parameters to optimize, we are focusing on k-means related algorithms and some hierarchical algorithms. Thus, clustering methods that use other parameters will not be used in this evaluation as the number of clusters in the result of these methods may greatly differ from the number of actual clusters present in the dataset and thus would behave poorly with ground truth related evaluations.

4.3.2.1. Text Preprocessing

Briefly mentioned in section 3.1., text preprocessing is a common module used in most NLP tasks. The act of splitting the text into words is called tokenization and is applied to all experiments. Several preprocessing techniques can be applied to the text documents. These transformations may reduce the high dimensionality of the feature space, as they reduce the number of words in the vocabulary. As they remove noisy features, they are expected to improve the clustering result, efficiency, and performance.

Words present in nearly all text documents do not add relevancy to the document. Examples of these words are prepositions, pronouns, and articles such as you, your, as and is. This group of words is commonly referred as stopwords. They do not contribute to the semantic value of text as their appearance in one document does not help differentiate from the others. Commonly stored in a list, the removal of stopwords is considered a preprocessing step.

Words that share a common lemma (canonical form of a set of words) can be represented with the same term. This process of taking into consideration the morphological structure of words and

transform them back to their lemma is called lemmatization. This is achieved by having a detailed dictionary where the algorithm can connect a word to its lemma.

Stemming algorithms on the other hand remove parts of a given word by considering prefixes and suffixes usually found in inflected words. This new term is not necessarily a real word.

	Pre-processing method	
Original Word	**Stemming**	**Lemmatization**
is	is	Be
Change	Chang	Change
changing	chang	Change
densely	dens	Densely
populated	popul	populated

Table 4.5: Differences between Stemming and Lemmatization

Lemmatization has the capability of producing real words which may be an advantage but takes longer than stemming to compute.

Taking advantage of the situation regarding clustering evaluation, we will experiment different combinations of these techniques and perform a comparison of its impact on the results.

4.3.2.2. Clustering External Evaluation

After presenting different methodologies, we will be now evaluating its effect with the External Quality Measures mentioned in 2.3.1. The experiments will go as such:

- Apply different combinations of preprocessing techniques to the corpus.
- Represent the documents with TFIDF and SBERT, the two remaining representations of text.
- Run the k-means, k-medoid and agglomerative clustering algorithms with number of clusters set to the same as the number of templates in the Silver Data (721).
- Use ARI and the V-measure (section 2.3.1.) to evaluate the clustering results.

In this experiment, we also had an iteration where apart from tokenization, no preprocess was applied.

Due to the preprocessing, the dimensionality of the representations suffered minor modifications. While the SBERT representation has the same dimensionality, 512, since it is a fixed vector; the dimensionality of the TF-IDF representations were reduced by a small margin.

		Pre-Processing Applied					
Clustering Model	Text Representation	No pre-processing	Stopwords Removal	Stopwords Removal + Lemmatization	Stopwords Removal + Stemming	Lemmatization	Stemming
K-means	TF-IDF	0.560	0.352	0.561	0.355	0.329	0.404
	SBERT	0.334	0.343	0.368	0.591	0.336	0.357
K-medoids	TF-IDF	0.731	0.750	0.729	0.717	0.720	0.746
	SBERT	0.745	0.709	0.729	0.713	0.756	0.742

Table 4.6: Average ARI in different pre-processing combinations

		Pre-Processing Applied					
Clustering Model	Text Representation	No pre-processing	Stopwords Removal	Stopwords Removal + Lemmatization	Stopwords Removal + Stemming	Lemmatization	Stemming
K-means	TF-IDF	0.858	0.823	0.858	0.824	0.818	0.833
	SBERT	0.745	0.822	0.825	0.860	0.823	0.822
K-medoids	TF-IDF	0.896	0.896	0.893	0.890	0.892	0.896
	SBERT	0.895	0.89	0.893	0.891	0.896	0.894

Table 4.7: Average V-measure in different pre-processing combinations

Tables 4.6 and 4.7 present the results of the experiments. Agglomerative Clustering with stopwords removal and Lemmatization pre-processing achieved 0.572 ARI and 0.867 V-measure with TF-IDF representation. With SBERT representation it achieved 0.324 and 0.763 respectively. The first conclusion we draw is that for all the Clustering Algorithm and Document Preprocessing iterations, the TF-IDF and SBERT approaches are similar with TF-IDF having a slight edge with the ARI metric. Secondly, all three clustering algorithms achieves similar results, with k-medoids somewhat better. Thirdly, the document preprocessing techniques do not change the results significantly. Nevertheless, we will apply the stop words removal and stemming techniques (as it is slightly better) to all the emails in the final approach and use the TF-IDF representation when executing Clustering Algorithms. It is also noteworthy that since SBERT utilizes contextualized word embeddings, it makes sense that such text pre-processing techniques add no further value to its representation and may even be detrimental.

5. RESULTS AND DISCUSSION

In this chapter, we start by analyzing the situation and applying the insights gained in previous experiments to the new clustering problem. Then we analyze the results and create strategies to extract templates from the results. Finally, we look at the extracted templates and evaluate the feasibility of the given task.

5.1. EXPERIMENTAL SETUP

With the document representation and similarity measure chosen, we will now tackle the main objective of finding new templates. We will apply the clustering algorithms presented and researched in this work and discuss its performance in the given problem. Contrary to what happened in the evaluation tasks in section 4.3.2.2, the full email corpus does not have all the data labeled with the assigned template. As such we are unable to use external quality measures for cluster evaluation and must rely on internal quality measures.

Throughout this work, we have presented different clustering algorithms. Due to the experimentation done in section 4.3.2, we have gained some insights regarding such algorithms.

The previous clustering experiences had the advantage of knowing the exact number of clusters pretended. This allowed the use of certain algorithms without the need of finding the optimal number of clusters. Nevertheless, some strategies to overcome this limitation can be done. Firstly, other algorithms that do not require a preselected number of clusters can be applied to the full data. Secondly, we can use the internal quality measures as a tool to discover the optimal number of cluster when using algorithms like k-means. Hence, this clustering problem would become a problem of optimizing the selected internal quality measure. To note that since there are 721 different templates on the silver dataset, we can assume there are more than 721 in the whole Email Corpus. However, since we are dealing with unsupervised learning, we cannot tell with absolute certainty that the clustering results represent our understanding of templates and as such will have to create strategies to analyze the results. Therefore, results that show a lesser number of clusters than the already known number of templates can still give insights to other not known templates.

Another issue is the scalability of the problem. In the previous clustering experiment, we had 24k entries which was still within the bounds of memory availability. With the whole dataset, the number of emails present is now more than triple. There are algorithms that require the input of a similarity matrix. This matrix is usually an n x n matrix, where n is the number of elements to check similarity, and each entry is the computed similarity between the column and row element. When using the full corpus, we would have a 54k * 54k = 2 916 000 000 similarity computations.

In this next phase of this work, there was an intent to apply different clustering algorithms to the full data. However, due to memory and scalability issues, we were unable to apply algorithms other than k-means and a created algorithm. While k-medoid and agglomerative hierarchical clustering achieved similar results to k-means in previous tasks, they might have brought different results in the following problem.

As mentioned before, the stopwords and lemmatization pre-processes will be applied to the full data.

5.1.1. Optimal K-means

The implementation of the k-means on the full data differs from the one in section 4.3.2.2, as we do not know the "optimal" number of clusters. As such, we will be using the Internal Quality Measures discussed in section 2.3.2 to determine such number.

Therefore, the strategy will be to initialize k-means with different numbers of clusters and assess its impact the Internal Quality Measures. These iterations will range from 500 to 2000 in an interval of 150. Table 5.1 presents the obtained results.

Number of clusters	Silhouette Score
500	0.115
650	0.169
800	0.178
950	0.105
1100	0.177
1250	0.165
1400	0.152
1550	0.134
1700	0.065
1850	0.065

Table 5.1: Average Silhouette Score per Number of Clusters

We observe that the algorithm achieves a slightly better performance than its counterparts when the number of clusters is 800. However, if we take the initialization randomness into account, we can argue that other number of clusters could achieve similar results. For example, when the number of clusters is 1100 the Silhouette Score is only 0.01 worse.

As we have such close scores, other factors must be considered. Firstly, more clusters would imply less impact from loose emails (outliers). However, the number of clusters with very few elements would increase. There is also the fact that we know there are at least 743 templates in the dataset. As each option has its advantages and disadvantages, either one would have been an acceptable choice. Even though we will further analyze both solutions, we decided to use 1100 as our optimal number of clusters.

5.1.2. Cosine Algorithm

The following created algorithm considers the insights given by the annotations in section 4.2.2, about the similarity measures. Given the annotations and feedback from the human agent, it was decided that if two emails have a cosine similarity of more than 0.8, they are very similar and could be part of

the same template. This algorithm is also based on the previously mentioned clustering algorithm proposed by (Mota, Martins, & Coheur, 2013) and it somewhat resembles the Agglomerative Hierarchical clustering algorithm.

Let already_clust be an empty list. The algorithm is as follows:

1. Select an email from the dataset. If it is not in already_clust, add it and calculate the cosine similarity with every other email.
2. For every cosine similarity, if it is higher than 0.8, add the corresponding email to the already_clust list. The selected email and newly added emails to the list are now a provisory cluster.
3. Repeat steps 1 and 2 until all the dataset has been added to the already_clust list.
4. Check provisory clusters. If an email appears on several provisory clusters, group all the provisory clusters it belongs to into a single cluster. Repeat until each email appears in only one cluster.

This algorithm has some advantages and disadvantages. Firstly, since the cosine similarity is calculated one row at a time, there are no memory issues. However, since not every pair of email is compared to each other and due to the algorithm being done sequentially, there might exist some pair of emails that satisfy the cosine similarity condition but are not in the same cluster. For example, the algorithm selects email A and verifies email B is similar to it. They are now on the same cluster and due to how the algorithm is set, it will not verify similar emails to B. The algorithm then selects email C and verifies email D is similar to it. The same situation applies, and the algorithm will not check for similar emails to D. The issue here is email B and email D might have been similar, which would have grouped these two clusters together.

When applying this algorithm to the full dataset, we obtain 24473 different clusters. In the given iteration, the order in which the cosine similarity was calculated was not random. The dataset was organized with the emails that possess macros appearing first (organized by macro, meaning first all emails from the first macro, then all emails from the second macro, and so forth), and the emails that do not belong to a macro appearing last. This may have skewed the results. Also, there are so many clusters due to most non-macro emails being alone in a cluster.

The silhouette score computed to the given partition was -0.35. While unexpected, this calculation can be a consequence of various factors such as a very large number of clusters.

5.2. Template Extraction

From the previous partitions of the data, each cluster can be categorized into various types. Firstly, there are clusters comprised only of emails with a given macro. Secondly, clusters where some emails that did not have an assigned macro were correctly grouped with a group of emails with an assigned macro. And thirdly, a group of emails that did not have an assigned macro and are all similar which would translate into the discovery of a new template. These would be the “correct” clusters in an ideal situation. The remaining types of clusters are those that either mix emails with different assigned macros (which in turn could mean that those different macros could be grouped in only one), and badly constructed clusters, where all types of emails are grouped and do not make logic sense to a human

agent. The final part of this project is to detect the third type of clusters and verify its validity in the given context.

5.2.1. Partition Analysis

Considering these factors, we present some statistics that may help gauge the effectiveness of the used algorithms to find the third type of clusters mentioned and help find candidates for new templates.

K-means Partition

The chosen k-means partition has 1100 clusters. By filtering out the clusters that have at least one email with a macro, we are left with 495 clusters. This also means 605 clusters have at least one email with a macro assigned to it, which is somewhat comparable to the number of clusters present in the silver dataset (721). The size of each of these 495 clusters ranges from 1 to 359. To further narrow down the list of candidates, we only considered clusters that had 10 or more elements. This left us with 59 possible templates.

Cosine Algorithm Partition

As mentioned before, this algorithm's result had 24473 clusters. Looking at its silhouette score and number of clusters, there might be some doubts regarding its effectiveness, nevertheless it can give some insights to the given problem. Making a similar filter to the previous one, we considered only clusters with more than 10 elements and with less than 30 percent of emails with macros. 153 clusters remained.

Partition Comparison

External quality measures can be used to compare different partitions. However, since both partitions have such a difference in number of clusters, the comparison would not be reasonable. As such, we will be comparing the partitions by hand.

As referred in section 5.1.2, the emails with an assigned macro appear first on the dataset. This allowed the identification of similar clusters of each partition. For example, the first email that appears belongs to the first macro. This first macro has 67 elements. If we look at the partitions, the cosine algorithm groups together 65 of those elements, while the k-means partition correctly groups the same 67 elements of the macro. This does not always happen. Another email belongs to the cluster number 448 of the cosine partition. This cluster has 665 elements. If we look at the k-means partition on those elements, there at least 5 different groups. There is also the example of macro number 500, where both partitions have very different groupings to one another, while neither being correct.

The aim of this comparison was to corroborate the fact that different partitions can give different insights and it is difficult to ascertain that one is better than the other.

5.2.2. Candidates Quality and Template Extraction

From the previous analysis, we extracted potential candidates for new templates. To further evaluate their validity, we calculated the average Silhouette Score for each of the remaining clusters and ordered them from highest to lowest.

Cluster nº	Percentage of emails with known template	Cluster Silhouette Score	Number of emails
1	0	1	28
2	0	0.98	54
3	0	0.978	58
4	0	0.903	76
5	0	0.852	13

Table 5.2: Top 5 clusters with higher average silhouette score

We further filtered the k-means with 1100 clusters partition and considered only clusters with average silhouette score higher than 0.5. This left us with only 13 template candidates. From these 13 only 5 had silhouette score higher than 0.8. As mentioned in section 2.3.2, this indicates the cluster has high similarity within itself and is well separated from other clusters. To further assure the quality of each cluster, the help of a human agent is required. While we did not have an extensive hands-on analysis of each template candidate and every possible pair, from the pairs we have compared, they do appear to belong to the same template.

To extract a template from each cluster, we retrieved a random email belonging to the cluster. Another strategy was computing the median email or choosing the closest email to the centroid.

Analysis of other template candidates was also done, but we arrived at no consensus since while some of the compared pairs were similar, others were not.

6. CONCLUSION

The goal of this project was to develop a methodology capable of extracting unknown templates from a group of customer support response emails. To do so, we propose a solution based on clustering techniques. As the focus of this work, we extensively researched and analyzed these techniques. Three main components could be gathered: the clustering algorithm, the document representation, and the similarity measure.

In the data provided to us, some emails already had an assigned template. Due to this uniqueness of the data, a smaller dataset with known ground truth was extracted. This enabled the use of external cluster validation measures. In addition, an annotated dataset was created. This contribution further enabled the evaluation of clustering components. Particularly, high correlation was observation between most similarity measures and the evaluation of the human agent.

To represent the documents into readable data by the algorithms, some techniques were considered. TF-IDF representations had better performance when compared with contextualized word embeddings, such as SBERT. Likewise, we compared different similarity measures, with cosine similarity achieving better results.

Several clustering algorithms were evaluated during this research. In initial experiments, better results were achieved by hierarchical clustering algorithms and medoid based algorithms. These types of algorithms are better suited for document clustering. While evaluating different clustering algorithms, we also tested the impact of different text preprocessing techniques such as stopwords removal and stemming. Overall, the impact was non-significant.

These insights were then applied to the full dataset. While we focused on the partition with better silhouette score, further analysis led to believe that almost every clustering partition could be used to give insights to the problem at hand. While we did find some unknown templates, as they had very high scores, we have no feasible way of evaluating every single cluster.

As most template candidates were comprised of various types of emails, we conclude that further clustering could be done, either with by initializing k-means with a higher number of clusters or applying hierarchical clustering with a specific cutoff.

7. LIMITATIONS AND FUTURE WORK

One of the limitations of this project is inherent to its approach. As mentioned by (Hassani & Seidl, 2017), clustering evaluation is often as much or more difficult than the clustering itself. While in this case we had the advantage of having some emails with assigned templates and used this particularity to verify the if clustering algorithm was able to detect some of the known templates, an optimal solution to evaluate the clustering results of the full data is still difficult without the need for human intervention.

Another limitation is that we were unable to apply some of the algorithms to the full dataset. When applied to the silver dataset, they achieved slightly better results than algorithms applied to the full dataset. As such it would be useful to explore methods of applying such algorithms to a higher quantity of dataset, perhaps in an implementation by batches.

Future approaches could include the exploration of other known clustering algorithms such as fuzzy clustering, or perhaps the creation of custom algorithms based on the insights given by this work.

It would also be interesting to measure the impact of other data pre-processing techniques. While SBERT embeddings have fixed dimensionality, representations based on bag-of-words have dimensionality that depend on the number of words in the given vocabulary. Using only specific words, such as the most used or the least used, could be interesting. Furthermore, processes of data dimensionality reduction such as Principal Component Analysis and Latent Dirichlet Allocation could be applied.

Most response emails can be divided in three main parts: the greeting, which is the beginning of the message, where we usually find the name to whom the given email is addressed to and a small opening introduction (such as Hello or Thank you for reaching out); the body, where the main content of the mail is; and the signoff, which contains a closing statement. Content wise, the greetings and the signoff offer no value to the email. While representations like TF-IDF somewhat counter this by assigning smaller weights to terms that appear in many different documents (which would be the case for words belonging to these two parts of the email), their removal from the document could further enhance the document representation phase of the work.

On an ending note, while we had issues fully evaluating new templates, we confirmed that some known templates were also correctly identified. As such it would also be interesting to apply this methodology to different datasets.

8. BIBLIOGRAPHY

Ariff, N. M., Bakar, M. A., & Rahman, M. I. (2018). Comparative study of document clustering algorithms. *International Journal of Engineering and Technology (UAE), 7(4)*, 246-251.

Arthur, D., & Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. *Society for Industrial and Applied Mathematics*, 1027-1035.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. Retrieved from https://arxiv.org/abs/1409.0473

Balabantaray, R. C., Sarma, C., & Jha, M. (2015). Document clustering using k-means and k-medoids. Retrieved from https://arxiv.org/abs/1502.07938

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. Retrieved from https://arxiv.org/abs/1409.1259

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. Retrieved from https://arxiv.org/abs/1506.07503

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence, (2)*, 224-227.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. Retrieved from https://arxiv.org/abs/1810.04805

Dhillon, I. S., & Modha, D. S. (2001). Decompositions for Large Sparse Text Data Using Clustering. *Machine Learning*, 143-175. doi:10.1023/A:1007612920971

Hassani, M., & Seidl, T. (2017). Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. *Vietnam Journal of Computer Science, 4(3)*, 171-183.

Huang, A. (2008). Similarity measures for text document clustering. *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, *4*, pp. 9-56.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR), 31(3)*, 264-323. doi:10.1145/331499.331504

Kaufman, L., & Rousseew, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons. doi:10.1002/9780470316801

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. Retrieved from https://arxiv.org/abs/1907.11692

Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (pp. 1412-1421). doi:10.18653/v1/D15-1166

Martin, L., Muller, B., Suárez, P. J., Dupont, Y., Romary, L., de la Clergerie, É. V., . . . Sagot, B. (2019). CamemBERT: a Tasty French Language Model. Retrieved from https://arxiv.org/abs/1911.03894

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. Retrieved from https://arxiv.org/abs/1301.3781

Mitchell, T. M. (1997). *Machine Learning.*

Mota, P., Martins, B., & Coheur, L. (2013). Designing an answer template retrieval system. Technical Report 35, INESC-ID.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. Retrieved from https://arxiv.org/abs/1802.05365

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association, 66(336)*, 846-850.

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. Retrieved from https://arxiv.org/abs/1908.10084

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

Rosenblatt, F. (1958). The perceptron: a probalistic model for information storage and organization in the brain. *Psychological Review, 65(6)*, 386-408.

Santos, J. M., & Embrechts, M. (2009). On the use of the adjusted rand index as a metric for evaluating supervised classification. *International conference on artificial neural networks*, 175-184. doi:10.1007/978-3-642-04277-5_18

Sasaki, M., & Shinnou, H. (2005). Spam detection using text clustering. *2005 International Conference on Cyberworlds (CW'05)*, (pp. 4-pp).

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms.* Cambridge: Cambridge University Press. doi:10.1017/CBO9781107298019

Steinbach, M., Karypis, G., & Kumvar, V. (2000). A comparison of document clustering techniques.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 3104-3112.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *Advances in neural information processing systems*, (pp. 5998-6008).

Wu, Y., Schuster, M., Chen, Z., L, Q. V., M. N., Macherey, W., . . . Dean, J. (2016). Google's neural machine translation system: Bridging the gap between. Retrieved from https://arxiv.org/abs/1609.08144v2

Zhu, Y., Kiros, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Filder, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *Proceedings of the IEEE international conference on computer vision*, (pp. 19-27).