

DSpace Institution

DSpace Repository

<http://dspace.org>

Information Technology

thesis

2022-08

TOPIC CLUSTERING ON AMHARIC COMMENTS USING BERT EMBEDDING AND PARTITIONING ALGORITHM

HAIMANOT, HAILU

<http://ir.bdu.edu.et/handle/123456789/14784>

Downloaded from DSpace Repository, DSpace Institution's institutional repository



**BAHIR DAR UNIVERSITY
BAHIR DAR INSTITUTE OF TECHNOLOGY
SCHOOL OF GRADUATE STUDIES
FACULTY OF COMPUTING**

MSc. Thesis On:

**TOPIC CLUSTERING ON AMHARIC COMMENTS USING BERT
EMBEDDING AND PARTITIONING ALGORITHMS.**

**BY
HAIMANOT HAILU**

August, 2022

BAHIR DAR, ETHIOPIA



BAHIR DAR UNIVERSITY
BAHIR DAR INSTITUTE OF TECHNOLOGY
SCHOOL OF GRADUATE STUDIES
FACULTY OF COMPUTING

**TOPIC CLUSTERING ON AMHARIC COMMENTS USING BERT EMBEDDING AND
PARTITIONING ALGORITHMS**

BY
HAIMANOT HAILU

A thesis submitted to the school of graduate studies of Bahir Dar Institute of Technology, BDU in partial fulfillment of the requirement of the degree of Masters in Information Technology in the Faculty of Computing.

ADVISER: DR. GEBEYEHU BELAY (ASSOCIATE PROFESSOR)

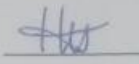
August, 2022

Bahir Dar, Ethiopia

©2022 HAIMANOT HAILU

DECLARATION

This is to certify that the thesis entitled “**Amharic Comments Topic Clustering Using Bert Embedding and Partition Based Clustering Algorithms**”, submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Technology under the Faculty of Computing, Bahir Dar Institute of Technology is a record of original work carried out by me and has never been submitted to this or any other institution to get any other degree or certificates. The assistance and help I received during this investigation have been duly acknowledged.

<u>Haimanot Hailu</u>	<u></u>	<u>25/07/2015</u>
Name of the candidate	Signature	Date

**BAHIR DAR UNIVERSITY
BAHIR DAR INSTITUTE OF TECHNOLOGY
SCHOOL OF GRADUATE STUDIES
FACULTY OF COMPUTING**

Approval of thesis for defense result

I hereby confirm that the changes required by the examiners have been carried out and incorporated in the final thesis.

Name of Student: Hammarot Hailu Signature [Signature] Date 9/28/2022

As members of the board of examiners, we examined this thesis entitled "Amharic comments topic clustering using BERT embedding and partitioning based clustering algorithm" by Hammarot Hailu. We hereby certify that the thesis is accepted for fulfilling the requirements for the award of the degree of masters of Science in "Information Technology".

Board of Examiners

Name of Advisor	Signature	Date
<u>DR. GEBEYEHU BELAY</u>	<u>[Signature]</u>	<u>05-10-2022</u>

Name of External examiner	Signature	Date
<u>Tibebe Beshah (PhD)</u>	<u>[Signature]</u>	<u>9/27/2022</u>

Name of Internal Examiner	Signature	Date
<u>Dr. Abdulkarim Mohammed</u>	<u>[Signature]</u>	<u>05-10-2022</u>

Name of Chairperson	Signature	Date
<u>Mr. Dagnachew Melesew</u>	<u>[Signature]</u>	<u>05-10-2022</u>

Name of Chair Hokler	Signature	Date
<u>Abdulkarim Mohammed (PhD)</u>	<u>[Signature]</u>	<u>05-10-2022</u>

Name of Faculty Dean	Signature	Date
<u>Asegetachew E.</u>	<u>[Signature]</u>	<u>05/10/2022</u>



ACKNOWLEDGMENT

First, I want to thank my GOD for his forgiveness and for always giving me new chances for change. Secondly, I want to express my deepest thanks to my Advisor DR. GEBEYEHU BELAY (ASSOCIATE PROFESSOR) for his time, comments, and support during this research. The third is for my dear parents for their support throughout my life. finally, I want to thank my friends who always help me with this work.

ABSTRACT

Topic clustering is one of the methods to organize comments posted on Online Media Service (OMS) news. Online news such as social media news has many comments every day. However, most comments are not well-organized to easily find relevant information on a specific topic. And topic clustering for short text documents is a very challenging task, especially for comments that are very concise and contain few words per document. In addition, the short text has the problems of data sparsity and irregularity, and most words only appear once in a short text. To the best of our knowledge, there is no work for clustering short Amharic comments. To address the aforementioned problems, we have developed an Amharic comments topic clustering model using contextual sentence representation and partition-based algorithms. This thesis aims to design and develop a topic clustering model for Amharic comments on OMS news. We used BERT (Bidirectional Encoder Representations from Transformers) models for contextual sentence representation. The transfer learning method is used for sentence embedding of Amharic comments using English BERT. Finally, we applied mini-batch k-means and Fuzzy c-means clustering algorithms. We conducted experiments on the two models and the experiment results show that BERT embedding with mini-batch K-means clustering algorithm and BERT with fuzzy C-means clustering has equal values of 1.0 of the v-measure score, adjusted-rand-score, and adjusted-mutual-information-score. But fuzzy C-means have a lower silhouette-score value of 0.996 than mini-batch K-means which have a 0.998 score value. Mini-batch K-means clustering is more accurate and takes less time to compute. Fuzzy C-means clustering shows similar results that are comparable to mini-batch K-means clustering, but it takes longer to compute. Therefore, the mini-batch K-means clustering algorithm was found to be more appropriate to cluster Amharic comments to news.

TABLE OF CONTENTS

DECLARATION	ii
ACKNOWLEDGMENT.....	iv
ABSTRACT.....	v
LIST OF FIGURES	ix
LIST OF TABLES	x
ABBREVIATIONS	xi
CHAPTER ONE.....	1
INTRODUCTION	1
1.1 Background of the study	1
1.2 Statement of the Problem.....	3
1.3 Objective	5
1.3.1 General objective	5
1.3.2 Specific objectives	5
1.4 Scope and Limitation of the Study.....	5
1.5 Significance of the Study	6
1.6 Organization of the Thesis	7
CHAPTER TWO	8
LITERATURE REVIEW	8
2.1 Text Clustering.....	8
2.2 Text Clustering Approaches.....	8
2.3 Algorithms for Word Embedding	10
2.3.1 Contextual Word Embeddings.....	14
2.3.1.1 Transfer Learning	15
2.3.1.2 Cross-lingual Contextual Word Embeddings	17

2.4	Text Similarity Measurement Metrics	18
2.5	Text Clustering Algorithms	20
2.6	Clustering Evaluations Metrics	28
2.7	Amharic Language	31
2.7.1	Amharic Morphology	32
2.7.2	Amharic punctuation marks and numbers	32
2.8	Related works for short text clustering	33
CHAPTER THREE		37
METHODOLOGY		37
3.1	Introduction	37
3.2	Dataset Collection	38
3.3	Development Tool.....	39
3.4	The architecture of Topic clustering on Amharic Comments	40
3.4.1	Text Preprocessing.....	41
3.4.2	Semantic Word Representation	46
3.4.2.1	Cross-Lingual Contextual Word Embedding	47
3.4.3	Text similarity calculation	50
3.4.4	Clustering Algorithms	50
3.4.5	Clustering Performance Evaluation	53
CHAPTER FOUR.....		54
EXPERIMENTATION, RESULT, AND DISCUSSION		54
4.1	Introduction	54
4.2	Experimentation Setup	54
4.2.1	Dataset Description and Distribution.....	54
4.2.2	Environment and hyper-parameter Setups.....	54

4.2.3	Experimental Setup.....	59
4.3.	Experimentation Result of Amharic Comments Clustering Model	59
4.4.	Comparison of the two models for Amharic Comment Clustering	67
4.5.	DISCUSSION	69
CHAPTER FIVE		71
CONCLUSION AND RECOMMENDATION.....		71
5.1	Conclusion.....	71
5.2	Contribution	72
5.3	Recommendation.....	72
References		74
Appendix A.....		80
Appendix B		81
Appendix C		82
Appendix D.....		83
Appendix E		84

LIST OF FIGURES

Figure 1 Process for K-means Algorithm	23
Figure 2 Architecture of Amharic Comments Topic clustering	41
Figure 3: Tokenization algorithm	42
Figure 4 Normalization algorithm	43
Figure 5: Stop-word removal Algorithm	44
Figure 6 prefix Removal algorithm.....	45
Figure 7 suffix removal algorithm	46
Figure 8: Evaluation Results of Mini-Batch K-Means model	61
Figure 9 Results of Mini-Batch K-means with Different Number of Clusters.	63
Figure 10: Fuzzy C-means Model Evaluation Result for Three Experiments	65
Figure 11 Results of Fuzzy C-Means Model with Different Number of Clusters.	67
Figure 12 Comparison of the two models for Amharic Comment Clustering	68

LIST OF TABLES

Table 1 Hyper-parameter setups for BERT embedding.	55
Table 2: Hyper-parameter setups for mini-batch K-Means	56
Table 3: Hyper-parameter setups for Fuzzy C-Means	57
Table 4: Experimentation Results of BERT embedding and Mini-Batch K-Means Model	60
Table 5 Mini-Batch K-Means Model Evaluation Results for Different Cluster Numbers	62
Table 6 BERT and fuzzy C-Means model testing performance for Amharic comments clustering	64
Table 7 Fuzzy C-Means Model Evaluation Results for Different Cluster Numbers	66

ABBREVIATIONS

- AMI.....Adjusted Mutual Information
- ARI.....Adjusted Rand Index
- BERT..... Bidirectional encoder representation from the transformer
- CBOW....., Continuous Bag of words
- CLARAClustering Large Applications
- CLTLCross-lingual transfer learning
- CNN.....Convolutional neural network
- DBSCAN.....Density-Based Spatial Clustering of Applications with Noise
- DMM.....Dirichlet Multinomial Mixture
- EK.....Encyclopedic Knowledge
- Eps.....Epsilon
- GloVe.....Global Vectors for word representation
- GSDMM.....Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model for short text clustering.
- HDBSCAN.....Hierarchical Density-Based Spatial Clustering of Applications with Noise
- IDF..... Inverse document frequency
- LDA..... Latent Dirichlet Allocation
- LE..... Laplacian Eigenmaps
- mBERT..... Multilingual BERT
- MCL..... Markov Clustering Algorithm
- ML Machine Learning
- NLP..... Natural language processing
- NMI.....Normalized Mutual Information
- OMS..... Online Media Service
- RoBERTa..... Robustly Optimized Bidirectional Encoder Representations from transformers Pretraining Approach
- SBERT..... Sentence Bidirectional Encoder Representation from the transformer

- SIF.....Smooth Inverse Frequency
- STCC.....Short Text Clustering using Convolutional Neural Networks
- TF.....Term frequency
- TF-IDF..... Term frequency by Inverse document frequency
- TRTDTopic Representative Term Discovery
- TV..... Television
- VSMVector Space Model
- XLM..... Cross-Lingual Language Model
- XLM-R..... XLM-RoBERTa

CHAPTER ONE

INTRODUCTION

1.1 Background of the study

Web-based information on news is essential to address many users in a short period without geographic barriers and computer network bandwidth load. News on websites has many comments every day from users. The comments are vast in number including volumes and a quality of information that is quite essential to various stakeholders in the media industry (Nunzio, 2016). However, most comments are not well-organized to easily find relevant information on a specific topic. Organizing these texts (e.g., grouping them by topic) is an important step toward discovering trends (political, economic) in conversations and Data Mining /Machine Learning tasks. Clustering the texts into groups in their similarity is the foundation for many of these organizational strategies (Rakib, 2017).

Clustering is an unsupervised descriptive data mining technique that groups data instances into clusters, with related examples grouped and unrelated instances grouped apart (Waiyamai, 2020). It is essentially a grouping of items based on their similarity. Clustering is sometimes mistakenly referred to as automatic classification; however, this is incorrect because clusters are unknown before processing, whereas classes are predefined or well-known in classification. In contrast to classification, where the classifier learns the relationship between objects and classes from a so-called training set, cluster membership is guided by the distribution and nature of data (Assefa, 2020).

Short text clustering has become an increasingly significant task of Natural Language Processing (NLP). The methods are particularly essential for grouping information retrieval results, primarily to disclose different meanings within groupings of results. Short text can be organized into meaningful clusters (groups) by using various techniques and this facilitates the usage of short texts in different application areas. Clustering is done utilizing short text data such as tweets, Facebook comments, and various news feeds, among other things. Short text, as the name implies,

is a text that is only a few words long. for instance, a short text on Twitter is fewer than 140 characters long (Siddiqui & Aalam, 2019).

Small text is usually quite brief and contains few words per document. Problems with data irregularity and sparsity are present in the short text, where most terms only occur once. And this makes clustering of comments a very difficult operation. Text ambiguity and a lack of a substantial number of features in document vectors are the causes of short text clustering challenges. As a result, the classical TF-IDF (term frequency-inverse document frequency) or bag-of-words leads to sparse vector representations. Using vector space for high dimensional data results in sparsity and results in a lot of computing and memory storage. And, other vector representations like word2vec, GloVe, and fastText, are word-level vector representations that cannot disambiguate the word senses based on the surrounding context and express all possible meanings of a word as a single vector representation; these properties did not match with the nature of the short text. In addition, directly applying normal text clustering methods may not work well when applied to short texts (Dai, 2020).

Given the issues existing in short texts and traditional vector representation techniques. BERT (Bidirectional Encoder Representations from Transformers) has significant advantages in feature extraction. BERT a contextual word embedding is released in 2018 by a team at Google AI (Bao et al., 2021; Kenton, 2019). BERT can understand different embeddings for identical words according to the context. The word embedding differs if the word denotes several meanings in various phrases (Hui Yin, 2022).

Clustering techniques have a high impact on the effectiveness of topic clustering and various techniques have been proposed. Such as affinity propagation, density-based clustering, hierarchical clustering, partitioning clustering, and topic modeling. Density-based A cluster in data space is defined as a contiguous region of high point density that is isolated from other clusters by contiguous regions of low point density. This is the foundational idea behind clustering. Sometimes, data points in the low point density dividing regions are referred to as noise or outliers. The disadvantage of density-based clustering is that, particularly if clusters have different densities, it is vulnerable to the choice of epsilon and minimum points. If eps is set too low, sparser

clusters will be misinterpreted as noise. If eps are very big, denser clusters may be joined. This implies that if there are clusters with varied local densities, a single eps value might not be sufficient (Lasek, 2019). Clusters are produced in a hierarchical tree-like structure using hierarchical clustering (also called a Dendrogram). This describes how a subset of linked data is arranged into a tree-like structure, with the root node representing the entire data set and branches going to other clusters. Since the similarity matrix must be saved in RAM, the Hierarchical Clustering Technique requires a lot of space when there are many data points (Team, 2020).

A common clustering technique is partitioning clustering, which divides a collection of N data points into a set of k non-overlapping subsets (clusters), each of which only contains one data point. For short texts, partitioning-based clustering is preferable because it is unaffected by high dimensionality data and requires less time than other algorithms (Waiyamai, 2020). A technique called topic modeling uses unsupervised machine learning to automatically cluster word groupings that best describe a batch of papers by scanning them for word and phrase patterns (Pascual, 2019).

The stimulation and promotion of real-world applications can be greatly increased by a well-designed short text topic grouping technique. They include subject detection, answering service recommendations, image or video tagging, and information retrieval. And it would be very helpful, allowing users to get a broad view of the conversation and focus on areas of interest to themselves, especially if high-quality and narrowly-focused topics were associated with the clusters, allowing them to easily understand what the comments within a cluster were about.

1.2 Statement of the Problem

Short text, as the name implies, is a text that is only a few words long. for instance, a short text on Twitter is fewer than 140 characters long (Siddiqui & Aalam, 2019). And microblogs, Tweets, news headlines, comments, etc. are also short texts (Dai, 2020). Newly posted comments are constantly published on users' timelines and it's difficult to find useful information from a huge group of comments. So, clustering those constantly posted comments to their related topic is a

significant task. But no prior work was conducted to cluster Amharic comments to news. As a result, to find the most fascinating information, stakeholders must go through all the newly posted content. The task of clustering short texts is more difficult than long texts. This is because of instantaneous features and briefness of the text brings sparsity, noise, and high dimensionalities throughout the text analytics process. Short text clustering has issues due to text ambiguity and a dearth of significant document vector properties. In addition, no prior work was conducted to cluster Amharic comments to news and it's difficult to find useful information from a huge group of Amharic comments.

Short text clustering is a difficult problem to solve using traditional methods. For instance, TF-IDF has less significance, due to the lower number of words in a sentence compared to a paragraph. If the frequency of each word in a sentence is one, the sentence is considered complete (because the sentence is short with no repeating words). Sparsity is the effect of using vector space for high-dimensional data. This results in a significant increase in computation and memory storage requirements. When there is no contextual information and only a small number of words in the content, it is challenging to achieve acceptable semantic comparisons because most words only appear once in a short text.

Previously various short text clustering techniques have been proposed. The research which focused on using topic modeling particularly LDA for short Amharic text topic clustering was proposed by (Assefa, 2020). But it doesn't perform well for short texts since these models suffer from data sparsity when applied to short documents (estimating reliable word co-occurrence statistics). Additionally, these techniques typically need at least a few hundred words to be precise.

Authors (Heu, 2018; Nunzio, 2016; Waiyamai, 2020; Wang, 2016) applied a method to expand short texts to long texts by the use of external knowledge sources to solve the sparsity issue of short texts. However, it has the following three drawbacks. The first is, that creating and maintaining such resources may be highly costly. Second, this poses a new problem in terms of determining how to best utilize those external resources. The third is the lack of those resources for the Amharic language. In most circumstances, fixing this new difficulty is time-consuming and

difficult in and of itself. Another method (Jiaming, 2015) takes a lot of practice and fine-tuning of various parameters and hyper-parameters.

Following the above paragraph, the research questions are summarized as follows.

- How effective are Cross-Lingual Transfer learning models for the contextual embedding of Amharic comments?
- How to sequence BERT embeddings with partition-based Algorithms to enhance the performance of the Amharic comments' topic clustering?
- To what extent the proposed model enhances the performance of Amharic comments topic clustering?

1.3 Objective

1.3.1 General objective

The general objective of this thesis is to design the Amharic comments topic clustering model using BERT embedding and partition-based algorithms.

1.3.2 Specific objectives

- To augment BERT-based embeddings on short Amharic text vector representation using online news comments.
- To sequence BERT-based embeddings with partition clustering Algorithms for an enhancement of clustering performance and accuracy.
- To Test and Evaluate the proposed topic clustering model.

1.4 Scope and Limitation of the Study

Short texts include news headlines, comments, status updates, web page snippets, tweets, question/answer pairs, etc. But the focus of this work is Amharic comments on the news only

because a vast number of comments are available freely and they are not in a well-organized format. In this thesis, we collect text files only and cluster them into different categories based on their semantic similarity.

Due to the time limit, the scope of this thesis work is limited to collecting Amharic comments on the news only from Amharic news agencies' websites (i.e., Facebook, Twitter, Instagram). For this research, we are not considering documents that are in an image, video, or audio format.

1.5 Significance of the Study

The most popular type of communication today is short text documents, particularly for user-generated content on social media. The number of such documents increases together with the popularity of social media use. Therefore, grouping those often-submitted comments by topic is a challenging process. For example, it enables the extraction of knowledge from a mass of text data: It is one of the most crucial text analysis approaches for drawing knowledge from the voluminous text data available online, such as Facebook comments and tweets. Easy access to the numerous topics discussed within a huge set of comments would benefit all user groups interested in online news commenting. Real-world applications can all benefit from a well-designed short text topic clustering method such as answering service recommendations, image or video tagging, and information retrieval.

At the end of the study, Users, Journalists, and Editors can get the following benefit:

- For Users: it allows quickly understand what the comments within each cluster were about and get a broad overview of the conversation and select sections of interest to them.
- Journalists and Editors: they would have access to several conversation topics sparked by their article, allowing them to engage with their audience in a more concentrated manner. They would be able to monitor the topics that are most interesting to readers or they can easily identify which topic is mostly discussed by readers.
- Governments: keep the public informed and updated about the idea of people on important issues and engage with an audience on a deeper level.

1.6 Organization of the Thesis

The remaining part of this thesis is presented as follows; Chapter two presents related works of short text topic clustering and different text clustering approaches. Chapter three presents the research methodology followed by development tools, data set collection, the architecture of the proposed thesis, clustering algorithms, and performance evaluation metrics for evaluating the performance of the clustering model. Chapter four presents the experimentation, results, and discussion. This chapter includes dataset description, experimentation setups, the performance result of the selected models with experiments, and discussion. Chapter five presents the conclusion and future works of the proposed work. in the end, the references and appendix are presented.

CHAPTER TWO

LITERATURE REVIEW

2.1 Text Clustering

The goal of cluster analysis also referred to as clustering, is to group objects into clusters that are more similar (in some ways) to those in other clusters (clusters). It is a common technique for statistical data analysis used in many domains, including pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics, and machine learning, and it is the main role of exploratory data analysis (Caiyan, 2018). Since it establishes the natural grouping among the unlabeled data, clustering is important. For effective clustering, no conditions must be met. The user oversees selecting the criteria they will utilize to meet their needs.

The method of creating groupings from unlabeled data is known as text clustering. The number of cluster groups is typically predefined by the user in most clustering algorithms, however, in this situation, the number of cluster groups must fluctuate dynamically. The main idea is that documents can be quantitatively represented as feature vectors. One way to compare text similarity is by measuring the distance between these feature vectors. Nearby objects ought to belong to the same cluster. Things that are far apart ought to be arranged in different groups. Any text clustering approach involves text pre-processing, feature extraction, and clustering (Wang, 2016).

- Text pre-processing: Information can be obscured by the noise of text, which might include stop words, inflections, and sparse representations. Once the dataset has been pre-processed, handling it is simpler.
- Feature extraction: For extracting vector representations from textual data.
- Clustering: grouping of distinct text documents using the features produced.

2.2 Text Clustering Approaches

Techniques for clustering can be divided into partition-based, hierarchical, density-based, and grid-based algorithms. These approaches differ in how similarity is measured (both within and

between clusters), how thresholds are used to create clusters, how objects are clustered, whether they allow objects to belong strictly to a single cluster, or whether they can belong to multiple clusters to varying degrees and algorithmic structure. Regardless of the technique, the cluster structure that is created is then used to assist in object retrieval or for user inspection (Suyal, 2016).

Clustering is divided into two groups: hard clustering and soft clustering, depending on whether they let objects belong solely to one cluster or can belong to several clusters. Data objects are grouped using a procedure called hard clustering, where each item is only assigned to one cluster. For instance, we want the algorithm to examine every tweet and determine if it is positive or negative. In hard clustering, each data point is a whole or partial member of a cluster. A popular hard clustering technique called K-Means separates the data into K clusters, with each item only belonging to one cluster. We do not always need a yes or no answer. On the other hand, soft clustering is a method of grouping data items so that any item can reside in more than one cluster. The process of arranging data objects into multiple clusters is known as soft clustering. One well-known soft clustering approach is fuzzy C-means. FCM works by giving things probabilities, which are essentially expressions of how strong the items in the cluster are (Malik, 2019; Sukemi, 2019).

Clustering approaches can be divided into document-based and keyword-based clustering. The distinction lies in the features that were used to group the documents. The clusters created by keyword-based clustering algorithms only choose specific document attributes and a small number of them. Those few traits were chosen since they are the most important differences between the documents. Similar documents have the same qualities. Choosing the most important feature is thus a crucial stage in keyword-based clustering. The document vector space model is used to apply document-based clustering methods, with each entry presenting the word weighting of a term in the matched document. As a result, each term serves as an axis and each document is represented as a data point in an extremely high-dimensional space. In this space, the distance between points can be calculated and compared. It is essential to map the documents into the correct space and employ the right distance computation algorithms because document-based clustering is based on "document distance" (Assefa, 2020).

2.3 Algorithms for Word Embedding

The ultimate level of data for any machine learning or deep learning model must be in numerical form because models do not immediately comprehend a word or visual data like individuals do. We need sophisticated ways in the NLP field to describe the conversion of text input into numerical data. Vectorization, also referred to as word embeddings in the NLP field, is a complex method of converting text input into numerical data. Word embedding is a collective term for a variety of language modeling and feature learning techniques used in natural language processing (NLP), in which words or phrases from a lexicon are translated into real-number vectors. Using a real-valued vector to capture the meaning of the word, word embedding is a technique for expressing words in text analysis. Words that are near each other in the vector space are taken to have similar meanings (Heidenreich, 2018; Waiyamai, 2020).

By converting words or phrases from a lexicon to real-number vectors using a variety of language modeling and feature learning algorithms, word embeddings are produced. A multidimensional space must be mathematically embedded into a continuous vector space with a much smaller dimension. Then, several machine learning models are built using the numerical vectors. We describe this as extracting features from text to create models for various natural languages and processing. We can transform text data into numerical vectors in a variety of ways (Brownlee, 2019). The most popular word embedding methods to extract features from the text are Bag of words, TF-IDF, Word2vec, Glove embedding, FastText, and the recent one BERT.

Bag of Words

The bag of words technique is simple to comprehend and use. Text classification and language modeling are the two main applications of this technique. This approach has a simple notion behind it. We'll use this strategy to turn sentences into vectors based on the frequency of terms that appear in them. It is a format that counts how many times each word appears in arbitrary text to generate fixed-length vectors. A bag of words functions as a baseline model and can thus be used to test the findings and learn more about the data that is fed into the model. After that, deep learning approaches can be pursued further. When the data is context specific, a bag of words can also be

used. The arrangement of the words in the sentence, as well as how the word is related to other sentences, are not recorded in the bag of words. The bag of words is mostly determined by the text's vocabulary. In this case, as the number of phrases grows, the vocabulary grows exponentially, making the model complicated and both computations more difficult (Kabaap, 2019).

TF-IDF

The TF-IDF (term frequency-inverse document frequency) statistic assesses a word's significance to a document within a group of documents. This is done by averaging two metrics—the frequency with which a word appears in a document and its inverse document frequency—across a set of documents (Heidenreich, 2018; H. Yin et al., 2021).

The TF*IDF logarithm is a calculation of the term frequency and inverse document frequency:

- **TF: Term Frequency-** this measures how frequently the term is used in a single document. The higher the phrase frequency, the longer the document. The total number of terms in the document is then split by this number.

$TF = (\text{Number of times the term appears in the document}) / (\text{Total number of words in the document})$

- **IDF: Inverse Document Frequency** - This metric assesses the significance of a phrase in terms of its relevancy within the corpus. Stopwords such as "is," "of," and "the" are less important because they appear often in all documents in the corpus. The IDF can be calculated as follows:

$IDF = (\text{Total number of documents}) / (\text{total number of documents containing the keyword})$

Word overlap is a key component of TF-IDF, yet it is uncommon in short text texts like this one. So, TF-IDF is inappropriate. This method might produce a highly sparse document vector, which would lead to subpar clustering outcomes and a lengthy runtime (Waiyamai, 2020).

Word2vec

A shallow neural network called Word2vec is trained using inputs that include each instance of a target word and its neighbors. The embedding vector for the target word is then created using the network weights between the input and hidden layer. Words are represented in vector space by Word2vec. Words are represented as vectors, and placement is done so that ones with similar meanings are grouped and different words are placed far apart. A semantic relationship is another name for this. For each word, the Word2Vec embedding approach only offers a single, independent embedding vector. Only ONE vector for each word is saved by Word2vec in the output model. Word2vec is trained using contextual neighbors but used non-contextually for a downstream NLP task. Since the representation is only kept as a single vector per word. This restricts the ability to understand a word's meaning across two contexts (Ankit, 2020).

Word2vec uses two architectures: Skip Gram and Continuous Bag of Words (CBOW). Continuous Bag of Words (CBOW). CBOW is an algorithm that guesses a target word based on its surrounding context. It tries to predict the output (target word) based on the words around it (context words). CBOW is a word2vec version that predicts the center word from a set of context words. CBOW would tell us the most likely word in the center based on all the words in the context window (excluding the middle one)(Ezra, 2022; Shristikotaiah, 2020).

The skip-gram model trains a neural network to predict the probability of a word within a sentence window, and the word embeddings are retrieved from the weights of the skip-gram model. The main principle of the Skip-Gram model is that it selects each word from a large corpus (we'll call it the focus word) and extracts the words that surround it within a specified "window" one at a time to feed a neural network that, after training, predicts the likelihood for each word to appear in the window around the focus word. Word embeddings are computed with the Skip-gram Word2Vec architecture. Unlike CBow Word2Vec, Skip-gram Word2Vec predicts the surrounding words using the central word. The CBOW model integrates the scattered representations of context to predict the word in the middle (or surrounding words). On the other hand, the Skip-gram model guesses the context using the input word's dispersed representation (Heidenreich, 2018; Shristikotaiah, 2020). Both Word2Vec's predictive architectures fail to account for the global context and ignore the fact that some context words occur more frequently than others.

GloVe

Global Vectors for Word Representation is known as GloVe. It is an unsupervised learning technique created at Stanford University that tries to produce word embeddings by combining global word co-occurrence matrices from a corpus. The GloVe word embedding's primary objective is to use statistics to ascertain the link between the words. In contrast to the occurrence matrix, the co-occurrence matrix provides information on the frequency of a certain word pair occurring together. Each value in the co-occurrence matrix represents a pair of words that frequently appear together. The GloVe is built using a matrix factorization technique and a word context matrix. The process begins by creating a large matrix of (words x context) co-occurrence data, in which each word is counted as having appeared in a certain context throughout a huge corpus (Heidenreich, 2018).

Word-to-word cooccurrence statistics from the corpus are aggregated globally and used as training data for the GloVe, a word vector representation technique. As a result, it comprehends word representations and generates them based on context, just like word2vec. More successful than learning the raw occurrence probability were learning ratios of these co-occurrence probabilities. When storing the co-occurrence probability ratio between two words, GloVe Embeddings, a type of word embedding, uses vector differences instead (Ezra, 2022).

FastText

Facebook AI research produced FastText, a vector representation approach. As the name implies, it is a quick and efficient technique for performing the same operation, and because of the nature of its training approach, it also learns the morphological details. FastText is unique in that it can generate word vectors for unknown words as well as words from the lexicon. This is because it may use the morphological characteristics of words to construct the word vector for an unknown word. Since morphology pertains to the structure or syntax of words, FastText typically performs better for these tasks while word2vec performs better for semantic tasks. FastText embeddings generate word embeddings using subword information. Character n-gram representations are learned words representation as the sum of the n-gram vectors. This adds subword information to the word2vec

type models. This aids in the understanding of suffixes and prefixes via the embeddings (Mohanty, 2019).

FastText employs n-gram letters as the smallest unit, whereas Word2Vec and GLOVE use each word as the smallest unit to train on. The word vector "apple," for example, might be divided down into individual word vector units. Because the n-gram character vectors are shared with other words, FastText provides better word embeddings for rare words or even words that were not observed during training. This is something that neither Word2Vec nor GLOVE can do (Heu, 2018).

Generally, skip-gram trains a log-bilinear model to predict words within a specific window size using only the center word, whereas CBOW trains a similar model to predict the center word using a bag of context words. With noisy contrastive estimation and negative sampling, skip-gram and CBOW both approximate the word prediction softmax loss. Instead, Glove trains word embeddings to forecast statistics on the worldwide co-occurrence of words. Word2Vec is additionally improved by FastText by including subword data. Word embeddings, trained with local co-occurrence signals independent of order, represent each word type with a single fixed-dimensional vector due to efficiency considerations. Contextual word embeddings would eventually be able to overcome these restrictions as the deep learning framework and computer infrastructure advance.

2.3.1 Contextual Word Embeddings

Contextual word embeddings depict a word utilizing its context as processed by a deep neural network, in contrast to word embeddings. In numerous downstream tasks, contextualized representation beats stand-alone word embeddings, such as Word2Vec and Glove, with the same task-specific design.

BERT

A machine learning method for pre-training in natural language processing developed by Google is called BERT (Bidirectional Encoder Representations from Transformers) (NLP). BERT

embeddings are just vectors that represent a phrase's meaning; the vectors for similar-sounding words have nearer numbers. BERT's input embeddings are made up of three different embeddings. Those are token embeddings, segment embeddings, and positional embeddings. The pre-trained embeddings for various words are known as token embeddings. Segment embeddings are a vector's encoded sentence number. And Position embeddings are the position of the word within that sentence that is encoded into a vector (Kenton, 2019).

The majority of the aforementioned embedding techniques either represent words as singularly indexed values (one-hot encoding) or, more usefully, as neural word embeddings, where vocabulary words are matched to the fixed-length feature embeddings produced by Word2Vec or FastText models. Compared to models like Word2Vec, BERT has an advantage since it creates word representations that are dynamically influenced by the words surrounding them. Under Word2Vec, GloVe, and FastText each word has a fixed representation regardless of the context in which it appears. The context-informed word embeddings not only capture obvious variations like polysemy, but also other types of information that lead to more accurate feature representations and higher model performance (Ryan, 2019).

The BERT model has generated a buzz in the fields of natural language processing and machine learning. A Transformer, an attention mechanism that can learn the contextual relationships between words, is used to learn the text representation (or sub-words). According to the context, BERT can interpret different embeddings for the same word. If a word has distinct meanings in different phrases, the word embedding will be diverse as well. BERT has greatly improved the expressiveness of short text representations with more condensed, low-dimensional, and continuous features so, for the proposed work BERT was employed as a word embedding tool using the Transfer learning approach (Reimers, 2019).

2.3.1.1 Transfer Learning

Transfer learning is the process of employing a model that has already been trained to solve a new problem. It is presently particularly well-liked in deep learning because of its capacity to train deep neural networks with fewer data. This is very helpful in the data science field because most real-

world scenarios frequently do not have millions of labeled data points to train such complex models. Transfer learning's core principle is to apply what has been discovered in one activity to improve generalization in another. We apply the weights that a network learns at "task A" to a new "task B" (Donges, 2022).

The general concept is applying what a model has learned from one task with a lot of labeled training data to another task with little to no training data. The process of transferring as much knowledge as is practical from the task the model was trained on to the current task is known as transfer learning. This knowledge may be expressed in a variety of ways, depending on the situation and the available information. For instance, the construction of models may facilitate our ability to identify new objects. The construction of machine learning models benefits greatly from transfer learning. Transfer learning's primary advantages include resource savings and increased effectiveness while developing new models. Additionally, since most of the models will have already been trained, it can aid with model training when only unlabeled datasets are available (Seldon, 2021). Transfer learning for machine learning has the following primary advantages:

- Removing the requirement for each new model to have a significant collection of labeled training data.
- Enhancing the deployment and development of machine learning for a variety of models.
- A broader strategy for computer problem solving that uses many techniques to address new problems.
- Instead of training in real-world settings, models can be trained in simulations

Cross-Lingual and Multilingual Transfer Learning

The cross-lingual transfer is the process of applying learning to solve problems in another language, typically one with fewer resources, by using data and models accessible for one language for which there are plenty of such resources (e.g., English). By using annotated data from other languages, cross-lingual transfer learning (CLTL) enables the development of models for a target language (source languages). Cross-lingual transfer learning seeks to transfer models and resources from one language to another (Hassan, 2020).

A form of transfer learning called cross-lingual transfer learning has a different source and destination domain. It tries to transmit knowledge from one language, which is typically referred to as the source language, to another language, which is typically referred to as the target language. Multiple source languages or multiple target languages are both possible. Any cross-lingual signal, such as a bilingual dictionary or bitext, is further eliminated by a stronger assumption. The cross-lingual transfer is predicated on a cross-lingual representation space, and the cross-lingual space's quality, particularly for zero-shot transfer, is crucial. A single NLP system that supports several languages is what multilingual NLP aims to create. The development of multilingual NLP benefits from cross-lingual representation since it makes it easier to learn how to complete a particular task (Wu, 2022).

2.3.1.2 Cross-lingual Contextual Word Embeddings

The evolution of cross-lingual representation learning is comparable to that of NLP representation learning. Surprisingly, a cross-lingual transfer is successfully made possible by multilingual language models like XLM, mBERT, and XLM-R. It is possible to learn a model using supervised data in one language and utilize it for a range of tasks in another without any explicit cross-lingual signal. Using concatenated Wikipedia data for 104 languages without any explicit cross-lingual signal, such as pairs of words, sentences, or pages related across languages, Multilingual BERT (mBERT), a multilingual model offered by BERT, was pretrained (András, 2021; Wu, 2022).

The only difference between it and BERT's model architecture and training process is that it uses data from Wikipedia in 104 languages. The WordPiece modeling approach used in mBERT enables the model to share embeddings between languages. To account for the varying amounts of Wikipedia training data in different languages, the training applies a heuristic to subsample or oversample words when running WordPiece as well as sampling a training batch, random words for cloze, and random sentences for next sentence classification (Moberg, 2020).

The Transformer-based model XLM is trained with the MLM (masked language modeling) objective. XLM is also trained with a Translation Language Modeling (TLM) aim to make the model acquire analogous representations for several languages. Simple input of the same sentence

in two distinct languages and standard token masking constitutes TLM. The model then has the option of employing tokens from the other language to forecast a disguised token. XLM is trained with both MLM and TLM, with MLM using Wikipedia data in the 15 languages and TLM using various datasets based on language. Keep in mind that TLM needs a dataset of parallel sentences, which may be challenging to obtain (Karthikeyan, 2019).

The most recent multilingual model is XLM-R, where the R stands for RoBERTa. By avoiding the TLM target and stepping back from XLM, XLM-R merely trains RoBERTa on a sizable, multilingual dataset. 2.5 TB of unlabeled text in 100 languages is extracted from CommonCrawl databases. It was trained to utilize solely the MLM objective in a RoBERTa-style manner. The vocabulary size is the only significant distinction from RoBERTa. Without considering scale variation, the main difference between XLM and XLM-R is that XLM-R is entirely self-supervised while XLM requires parallel examples, which might be challenging to obtain at a large enough scale (Wu, 2022).

2.4 Text Similarity Measurement Metrics

Text similarity indicates how closely two text documents are related in terms of context or meaning. Distance metrics are the most prevalent techniques for determining word similarity. While there is a section called Document Similarity that searches for similarities between sentences or paragraphs of text. A distance with dimensions that represent the attributes of the data object serves as the similarity measure in a dataset. If the distance is little, there will be a high degree of similarity; but, if the distance is vast, there will be a low degree of similarity. There is various text similarity metrics exist such as Cosine similarity, Euclidean distance, and Jaccard Similarity. Each of these metrics has a definition that quantifies how similar two searches are to one another (Wibisono, 2021).

Cosine Similarity

Cosine similarity is used to compare two vectors in an inner product space. By measuring the cosine of the angle between two vectors, it can tell if they are generally pointing in the same

direction. Through text analysis, document similarity is frequently evaluated. A similarity metric called cosine similarity can be used to compare texts or, for instance, to order items according to a vector of search terms. A dataset's data objects are treated as a vector for cosine similarity calculations (Ma, 2018). The following is the formula to determine the cosine similarity between two vectors (Elton, 2022).

$$\text{Cos}(x, y) = \frac{X \cdot Y}{\|X\| * \|Y\|} \quad (2.1)$$

Where,

- $x \cdot y$ = product (dot) of the vector's 'x' and 'y'.
- $\|x\|$ and $\|y\|$ = length of the two vectors 'x' and 'y'.
- $\|x\| * \|y\|$ = cross product of the two vectors 'x' and 'y'.

The two comparable data objects are separated by the Euclidean distance due to their sizes, but they could have a lower angle between them if they share a cosine similarity. Angle decreases with increasing similarity (Saini, 2021).

Euclidian distance

In Mathematics, the Euclidian distance or Euclidean Metric represents the length of a line segment between two locations that can be determined using the Pythagorean Theorem. As a result, words are used to express these points in NLP. By calculating the distance between two objects, Euclidean distance determines their similarity (Ma, 2018). Euclidean distance is the foundation of many similarities and dissimilarity measurements. The distance between vectors X and Y is defined as follows (Ojha, 2020):

$$D(X, Y) = \sqrt{\sum_{i=0}^n (X_i - Y_i)^2} \quad (2.2)$$

In other terms, Euclidean distance is the square root of the total of the squared differences between comparable elements of two vectors. It's worth noting that the formula takes the values of X and

Y very seriously, with no scale adjustments. Only data measured on the same scale are suitable for Euclidean distance (Ojha, 2020).

2.5 Text Clustering Algorithms

Generally, Short text topic clustering techniques clustering can be classified into affinity propagation, density-based clustering, hierarchical clustering, partitioning clustering, and topic modeling (Waiyamai, 2020).

Density-Based Clustering

Density-Based Clustering is a type of unsupervised learning that is commonly employed in model construction and machine learning techniques. Noise is defined as data points in a region separated by two clusters of low point density (jvatpoint, 2022). Data is organized into regions with high data point densities that are surrounded by regions with low densities in density-based clustering. The program finds regions with a lot of data points and labels those regions as clusters. The clusters can take any shape, which is a nice feature of this. Expected circumstances are not a restriction on you. And outliers are disregarded since the clustering algorithms of this type don't attempt to group outliers with clusters.

Density-based spatial clustering of applications with noise (DBSCAN) is one of the most well-known and commonly used density-based clustering approaches, particularly for text document clustering. DBSCAN requires two basic inputs to execute clustering: the radius Epsilon (Eps) and the minimum points (Mints). The procedure starts with an arbitrary point p and uses the two input values to get all neighbor points that are density-reachable from point p (within distance Eps) and have not yet been visited. Clusters are regions of densely arranged objects that are separated by low-density or noisy zones. When the number of neighbors of point p is more than or equal to MinPts, a cluster of tightly connected points is produced. The starting point p is tagged as visited and added to this cluster alongside its neighbors. This technique is repeated for all of point p 's neighbors recursively. If the number of neighbors of point p is less than MinPts, the point is classed as noise. DBSCAN will recognize some data points as noise and not assign them to any cluster for

cluster completeness. Because of the non-linear time complexity of this technique, it takes a long time to run (Waiyamai, 2020).

Another form of density-based clustering approach is self-adjusting or hierarchical density-based spatial clustering of applications with noise (HDBSCAN). By transforming DBSCAN into a hierarchical clustering algorithm and then employing a method to extract a flat clustering based on the stability of clusters, it expands on DBSCAN. To separate clusters of varied densities from sparser noise, numerous distances are used. Because HDBSCAN is the most data-pushed clustering approach, it requires the least customer input (Berba, 2020).

Hierarchical-based clustering

Hierarchical-based clustering is a clustering approach that creates a tree-like structure from a layered sequence of divisions. Usually, hierarchical groups are depicted using the hierarchical tree known as a dendrogram. Everything is arranged top-down by creating a tree of clusters. It seeks to identify natural grouping based on the data's properties. Building a hierarchy is the first step in the hierarchical clustering algorithm's quest to identify nested groups of the data. It resembles the biological taxonomy used to classify plant and animal kingdoms. Typically, hierarchical-based clustering is applied to hierarchical data, such as that found in taxonomies or enterprise databases. Although it is more restrictive than the other clustering types, this is ideal for certain classes of data sets. There are two types of hierarchical clustering algorithms: Divisive and Agglomerative (McGregor, 2020).

Divisive which is a top-down method evaluates the complete set of data as one group at first before breaking it down into smaller groups iteratively. When the desired number of clusters is reached, division ceases if the number of a hierarchical clustering technique is known. Otherwise, the process comes to a stop when it can no longer divide the data, so the subgroup that is produced by the current iteration is identical to the one that was produced by the previous iteration (one can also consider that the division stops when each data point is a cluster). Agglomerative is a bottom-up strategy that depends on cluster fusion. In the beginning, the data is separated into m singleton clusters, where m is the total number of samples or data points. The number of clusters in each

iteration decreases as two clusters are iteratively combined into one. The cluster-merging process is complete when every cluster has been merged into one or when the desired number of clusters has been attained (Pedamkar, 2022).

CHAMELEON a revolutionary agglomerative hierarchical clustering technique or strategy for dealing with sparsity was proposed in 1999. In contrast to earlier agglomerative hierarchical clustering methods, CHAMELEON is a new clustering technique. It utilizes a sparse K-nearest neighbor graph, where nodes stand in for data items and weighted edges denote similarities between those data items. CHAMELEON uses a specific method known as the "two-phase algorithm" to create clusters from the data set. In the first phase, the K-nearest neighbor graph is subjected to a graph partitioning algorithm to cluster data items into several small sub-clusters. By continually merging these sub-clusters, an algorithm is utilized in the second step to find real clusters. CHAMELEON employs interconnectedness and closeness to determine the clusters' similarity in this two-phase process (Waiyamai, 2020).

Partitioning-based clustering

A typical method of clustering called partitioning divides a set of N data points into k non-overlapping clusters, with each data point belonging to exactly one cluster. This clustering technique categorizes the data into several groups based on their characteristics and similarities. The data analysts specify how many clusters must be created to complete the clustering procedures. The partitioning method separates the data into user-specified (K) divisions when a database (D) contains multiple(N) items, with each partition standing for a cluster and a region. Partitioning methods cover a wide range of algorithms, but some of the more well-known ones are K-Mean, PAM (K-Medoids), CLARA algorithm (Clustering Large Applications), etc. (Geeksforgeeks, 2020). Problems with text representations of brief text documents, which usually have large dimensions, do not respond well to density-based and hierarchical clustering techniques (Berba, 2020). On the other hand, partitioning clustering is unaffected by the high dimensionality of short texts (Waiyamai, 2020).

K-means clustering

One of the most well-known and often applied distance-based partitioning clustering algorithms is K-means clustering, notably for text document clustering. The K-means clustering algorithm requires many clusters to operate. The technique begins by selecting k arbitrary points at random to serve as cluster centers (centroids) for k clusters (Waiyamai, 2020). It separates things into several groupings, or clusters, to make objects within a cluster as similar as possible—that is, with a high intra-class similarity—and as different from one another as possible (i.e., low inter-class similarity). The mean of the points assigned to each cluster serves as the centroid, or center, of each cluster in k-means clustering. The primary tenet of k-means clustering is to minimize total intra-cluster variance, also known as total within-cluster variation while defining clusters. The process for K-means Algorithm is presented in figure 1 (Anand, 2020).

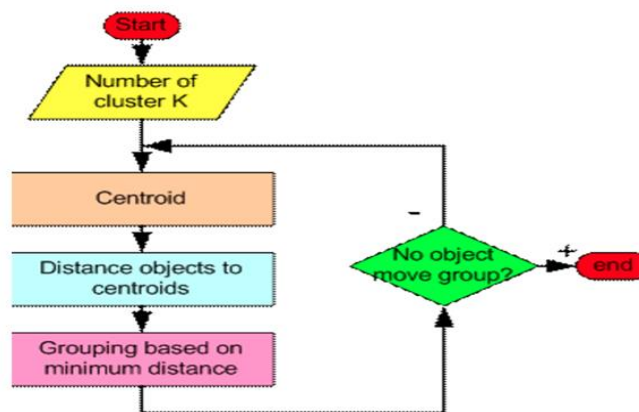


Figure 1 Process for K-means Algorithm

Steps involved in K-Means Clustering:

Step 1: Choosing the number of clusters (k) that will be created in the result is the first stage in using k-means clustering.

Step 2: The technique randomly selects k items from the data set as the initial cluster centers. Other terms for the selected objects include centroids or a cluster.

Step 3: The centroid closest to each of the remaining items is then selected; the centroid closest to an object is determined by its Euclidean distance from the cluster mean. We refer to this stage as "the cluster assignment step".

Step 4: Following the assignment phase, the algorithm determines the new mean value for each cluster. The cluster "centroid upgrade" refers to this phase. Once the centers have been recalculated, each observation is reexamined to see if it might be closer to a different cluster. The objects are again distributed using the updated cluster means.

Step 5: The cluster assignment and centroid update stages are iteratively repeated until the cluster assignments stop changing (i.e until convergence is achieved). In other words, the clusters produced in this iteration are exact replicas of the ones obtained in the prior iteration.

Pros of K-means algorithm

- **Simple:** K-means can be used to quickly identify unknown data categories in huge datasets. The results are simply reported.
- **Flexible:** The K-means method is amenable to modifications. Altering the cluster segment will allow for swift algorithmic fixes if any issues arise.
- **Suitable for a large dataset:** When compared to smaller datasets, K-means can handle multiple datasets and computes much more quickly. Larger clusters can also be produced by it.
- **Efficient:** The algorithm utilized can segment a huge data set efficiently. The shape of the clusters determines their efficacy. K-means perform well in clusters that are hyper-spherical.
- **Time complexity:** Because the number of data items in K-means segmentation is linear, the execution time increases. Unlike hierarchical algorithms, it does not take longer to classify similar qualities in data.
- **Tight clusters:** K-means yield tighter clusters than hierarchical algorithms, especially for globular clusters.
- **Simple to interpret:** The results are simple to comprehend. It provides simplified cluster descriptions to make the data easier to understand.
- **Cost of computation:** When compared to other clustering methods, the k-means clustering technique is both quick and efficient in terms of calculation.
- **Accuracy:** K-means analysis provides the availability of data regarding a problem domain and improves clustering accuracy. Based on this information,

the k-means algorithm is changed to increase the clusters' accuracy (Mary & Selvi, 2014).

Cons of K-means algorithm

- No optimal set of clusters: The greatest outcomes come from pre-selecting your clusters because K-means does not allow for the construction of an ideal collection of clusters.
- Lacks consistency: Results from different algorithm runs of K-means clustering are inconsistent. Consistency is produced when clustering results are determined by a random selection of cluster patterns.
- Consistent effect: It creates clusters of a consistent size even when the input data is of different sizes.
- As the size of the datasets being analyzed grows, the computation time increases since the complete dataset must be kept in the main memory.

Mini Batch K-means algorithm

As the size of the datasets under study grows, K-means' computation time grows as well because it needs to store the complete dataset in the main memory. As a result, a variety of methods for decreasing the algorithm's time and space cost have been offered. An additional technique is the Mini batch K-means algorithm.

A machine learning adaptation of the classic K-means clustering algorithm is called Mini-batch K-means. It gathers a random sample of the data to update the clusters after each iteration and maintains the data in memory in brief, random, fixed-size batches. It performs better than the standard K-means method when working with huge datasets because it does not cycle over the entire dataset. To update the clusters on each cycle, it first generates random batches of data to be stored in memory and then gathers a random batch of data. The Mini-batch K-means algorithm's main advantage is that it makes cluster detection less expensive to compute.

Although the mini-batch method is preferable when working with a large dataset, you might choose to employ the K-means algorithm (KHARWAL, 2021).

The main goal of the Mini Batch K-means method is to temporarily store small random batches of data. The clusters are updated using a fresh random sample from the dataset at the beginning of each iteration, and this procedure is continued until convergence. Each mini-batch uses a convex combination of prototype values and data to update the clusters, with the learning rate decreasing as the number of iterations rises. The learning rate is equal to the inverse of the number of data assigned to a cluster during the procedure. Convergence can be seen when there are no changes in the clusters for several iterations in a row since the impact of incoming data decreases as the number of iterations grows. The technique uses tiny, randomly chosen batches of the dataset for each iteration. Each piece of data in the batch is categorized into one of the clusters based on the centroids' prior placements. The cluster centroids' coordinates are then changed using the fresh points from the batch. The update is made via gradient descent, which is much faster than a batch K-Means update (Geeksforgeeks, 2021b).

Fuzzy C-Means Clustering

The Fuzzy C-means clustering algorithm is another partition clustering algorithm. Fuzzy K-Means clustering, also known as Fuzzy C-Means clustering, is a variant of K-Means clustering. The Fuzzy K-means clustering algorithm uses several points that are exclusively part of one cluster. Fuzzy clustering is a powerful unsupervised technique for data analysis and model building. Hard clustering is less natural in many cases than fuzzy clustering. Instead of being required to fully belong to one class, objects within the boundaries of many classes are given membership degrees between 0 and 1 to indicate their partial membership. The fuzzy c-means algorithm is the most widely used. This algorithm determines the membership of each data point in each cluster center based on the distance between the cluster center and the data point. The closer the data is near a cluster center, the more likely it is that it belongs to that cluster center. It should be evident that one should result from adding the membership of each data point (Cannon et al., 2012). The algorithm Fuzzy K-Means is identical to K-means, a popular simple clustering algorithm. The only difference is that instead of assigning a point to one cluster exclusively, it may have fuzziness or

overlap between two or more clusters (Edureka, 2019). The following are the main ideas that define fuzzy k-means:

- A single point in a soft cluster could belong to numerous clusters, each with a distinct affinity value.
- The affinity is proportional to the distance between the cluster centroid and that point.
- Fuzzy K-Means is like K-Means in that it works on objects that have a defined distance measure and may be represented in n-dimensional vector space.
- Choose the number of clusters.
- Assign coefficients at random to each point to create the first k clusters.
- Repeat these steps up until the algorithm converges.

The **Pros** and **Cons** of the fuzzy k-means algorithm are presented by (Getahun, 2021), as follows:

Pros of fuzzy k-means algorithm

- Performs better than the k-means algorithm and offers the best outcomes for overlapping data sets.
- Each cluster center is given a membership, allowing a data point to belong to more than one cluster center, as opposed to just allowing it to be a member of one.

Cons of fuzzy k-means algorithm

- The quality of the clusters formed is difficult to compare.
- With a lower termination criterion value, we achieve a better result, but it comes at the expense of additional iterations.
- Managing enormous data sets and a high number of prototypes can be challenging.

Topic modeling

An unsupervised machine learning technique called topic modeling can scan a collection of documents, identify word and phrase patterns within them, and then automatically cluster word

groupings and associated expressions that most accurately describe the set. The general "themes" that appear in a collection of texts can be found using a statistical modeling method called topic modeling (Qiang et al., 2019). It may use your extensive collection of documents to sort the words into word clusters and discover subjects by applying a similarity approach. It makes it easier to comprehend, arrange, and summarize huge text collections. But keep in mind that automated topic models work best with big amounts of content. It could be best to use another method if your document is short (ANALYTICS, 2017).

Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA) are the two topic modeling techniques that analysts employ the most frequently (LDA). According to the distributional hypothesis, understanding a word's semantics can be accomplished by examining the contexts in which it appears. On this concept, Latent Semantic Analysis (LSA) is founded. In other words, according to this theory, two words will have identical semantics if they frequently appear in comparable contexts. A probabilistic generative model of a corpus is called Latent Dirichlet Allocation (LDA). The fundamental assumption is that documents are modeled as mixes of latent subjects that are randomly chosen and each specified by a word distribution (Assefa, 2020).

2.6 Clustering Evaluations Metrics

Clustering is assessed using metrics of similarity and dissimilarity, such as the distance between cluster points. The algorithm has worked well if it can unify similar data points while also separating different data points. Assess the success of a clustering method, it is more complicated than just counting the number of errors or the precision and recall of a supervised classification system. Any evaluation metric should focus on whether this clustering defines data separations, such as a ground truth set of classes, or satisfying some assumptions that members of the same class are more similar than members of different classes according to some similarity metric, rather than the absolute values of the cluster labels.

Classification evaluation metrics cannot be used to assess the effectiveness of clustering methods. Because a dependent variable or target variable is required in any classification assessment metric, whether it be a confusion matrix or a log loss. The metrics are calculated using observed and

expected data in various evaluation procedures. The performance evaluation could be computed using evaluation metrics that do not require any ground truth labels to calculate the efficiency of the clustering algorithm(Sourabh, 2022).

The difficulty in assessing the efficacy of any clustering technique is one of the biggest disadvantages. To tackle this problem, the different metric has been developed. Some are purity, entropy, V-Measure, Silhouette Coefficient, and Rand Index. Purity and entropy can be used to compare partitioning with the same number of clusters, but they are unreliable when comparing partitioning with different numbers of clusters. This is because homogeneity will result from the way entropy and purity analyze the partitioning of sets of phrases inside each cluster. When there are too many clusters, the highest purity and lowest entropy scores are typically acquired, and this stage results in the least completeness. Therefore, the next metric takes both the consistency and completeness approaches into account (Systems, 2022).

V-Measure-Score

The harmonic mean of the clustering's homogeneity h and completeness c is the V-Measure, also known as normalized mutual information. Both metrics can be expressed in terms of the mutual information and entropy measures from information theory.

(2.3)

Were,

- h = homogeneity
- c = completeness

The calculation of the V-Measure first requires the calculation of two terms. Homogeneity measures how much the sample in a cluster is similar. Homogeneity is the **ratio** between the number of samples labeled c in cluster k and the total number of samples in cluster k . A perfectly homogenous clustering is one in which all data points belong to the same class label. Homogeneity refers to how near the clustering method comes to achieving this perfection. Completeness measures how much similar samples are put together by the clustering algorithm. A complete

clustering is one in which all data points from the same class are grouped. The clustering algorithm's completeness describes how close it is to perfection. Completeness is the ratio between the number of samples labeled c in cluster k and the total number of samples labeled c (Geeksforgeeks, 2019).

Silhouette Coefficient

The silhouette analysis method could be utilized to investigate the separation distance between the clusters created by the algorithm. Different sorts of distance metrics can be used to calculate the distance between the clusters (Euclidean, Manhattan, Minkowski, Hamming). The average silhouette coefficient applied to all samples is returned by silhouette score. The Silhouette Coefficient is determined for all samples by averaging the intra-cluster and nearest cluster distances. $[-1,1]$ is the range of the Silhouette Coefficient. The Silhouette Coefficients are higher (closer to +1) the larger the distance between clusters. If the value is zero, the sample is on or very close to the boundary that determines which of the two clusters is the neighbor; however, if the value is negative, the samples might have been put in the wrong cluster (Sourabh, 2022). The formula is as follows:

$$\frac{nc - ic}{\max(ic, nc)} \tag{2.4}$$

where ic = mean of the intra-cluster distance

nc = mean of the nearest-cluster distance

Adjusted Rand Index

The Rand Index is a function that determines how similar two clusterings are to one another. All sample pairs are considered when calculating the rand index, and pairs that belong to the same or different clusters according to the anticipated and actual clustering are counted. It has two parameters: labels-true, which are class labels used as the basis for comparison, and labels-predicted, which are label clusters (Tutorialspoint, 2019). the Rand Index is calculated as:

$$R = (a+b) / ({}_nC_2) \tag{2.5}$$

Where:

- **a:** the number of times a pair of components uses two different clustering techniques to belong to the same cluster.
- **b:** the number of times a pair of components use two separate clustering techniques to belong to different clusters.
- ${}_n C_2$: The number of unordered pairs in a set of n elements.

The Rand index always takes on a value between 0 and 1 where:

- **0:** demonstrates that no element pair is clustered using two different clustering algorithms.
- **1:** shows that every pair of components is perfectly clustered according to two different clustering algorithms (ZACH, 2021).

2.7 Amharic Language

A Semitic language called Amharic is used in northern Ethiopia. It has official status throughout the entire Federal Democratic Republic of Ethiopia because it is the most widely used and spoken language there. Several federal states and territories, like Amhara and the multiethnic Southern Nations, Nationalities, and Peoples Region, also use it as their official or working language. Outside of Ethiopia, millions of emigrants speak Amharic, which is also spoken in Eritrea. It is written using a Fidel or abugida writing system adopted from the now-extinct Ge'ez language (Seid Muhie, 2019).

Amharic has a semi-syllabic writing system of its own. The current Amharic writing system consists of a core of thirty-three characters (ፈደል, Fidel), each of which has a basic form as well as six supplementary orders. Through a series of routine modifications, non-basic forms are derived from basic forms. There are consequently 231 unique characters. The seven orders denote syllable combinations with a consonant and a vowel after it. There are also forty more that have a distinctive trait that usually represents labialization, such as, and so on. There are 275 characters in all (ፈደል, Fidels), although not all of them are strictly necessary for the spoken language's pronunciation

patterns; some were just transmitted from Ge'ez without any meaning or phonetic differentiation in modern Amharic. Only roughly 233 of the script's 275 symbols remain once the unnecessary ones are deleted. There are no upper- and lower-case variations in the Amharic writing system (MULUGETA, 2021).

2.7.1 Amharic Morphology

Amharic is one of the most morphologically difficult languages in the world. Number, definiteness, gender, and case are all marked on Amharic nouns and adjectives. They also used prepositions. The inflections and derivations of Amharic verbs, which consist of a stem and up to four prefixes and suffixes, are much more complicated than those of nouns and adjectives. The stem itself is made up of two parts: a root, which represents the verb's solely lexical component, and a template, which has spaces for the root segments' vowel-and-consonant-adjacent segments. The template represents tense, aspect, mood, and one of a few derivational categories: Iterative, causative, transitive, passive-reflexive, and causative reciprocal (Assefa, 2020).

Hundreds of words can be produced from a single verb root in Amharic by marking verbs for any combination of person, gender, number, case, tense/aspect, and mood. As a result, a single word can be used to represent a whole phrase including subject, verb, and object. Amharic, like other Semitic languages, has a morphological feature known as root-pattern morphology. A root is a group of consonants (also known as radicals) with lexical meaning. In Amharic, a stem is constructed by adding vowels or vowel patterns into the consonants of a root. This is the process of non-concatenative morphological features. In addition to this, Amharic uses different affixes to create inflectional word forms (Assefa, 2020; MULUGETA, 2021).

2.7.2 Amharic punctuation marks and numbers

Amharic also has its punctuation that is used in the Amharic writing system. There are a lot of punctuation marks in Amharic and roughly there are seventeen punctuation marks. However, only a few are commonly used and have software equivalents in Amharic (MULUGETA, 2021). Amharic punctuation varies greatly from English punctuation marks such as ፡፡(አራትነጥብ) is used as

a full stop, ፤ (ሶስትነጥብ) as a question mark, ፣ (ነጠላሰረዝ) as a comma, ፡፡ (አራትነጥብ) as a paragraph separator). Below is a list of some punctuation in both handwritten and computer-generated text.

- ፡፡ (አራትነጥብ) is a symbol for the end of a sentence and it serves the same purpose as a full stop in English.
- ፣ (ነጠላሰረዝ) serves the same purpose as the English comma and is used for separate lists in Amharic text.
- ፤ (ድርብሰረዝ) is used as a sentence separator in the Amharic writing system and it is the equivalent of the semi-colon.
- ፡ (ሁለትነጥብ) this punctuation is used to separate one Amharic word from the other in the Amharic writing system. But most commonly space is used in place of this punctuation.

Amharic occasionally uses Ethiopian numerals to write dates. Like ፩ is 1, ፪ is 2, and ፫ is 100. The Amharic Number system writing has 20 single characters which represent one (1/፩ up to 9/፱), tenths (ten/፲ to ninety/፻), hundred (፳), and ten thousand (፳፻).

2.8 Related works for short text clustering

Different short text clustering approaches have been proposed by using different clustering approaches. To start; (Heu, 2018) suggests a semantic-based K-means clustering technique that examines both the vector space model similarity between the data and the semantic similarity between the data by using TagCluster for clustering. These heuristic approaches rely heavily on the information from TagCluster. Furthermore, brief communications like news comments could not have access to such metadata. In the other work; A K-means partitioning-based clustering technique was used to cluster short texts, in which document similarities are quantified by the word mover's distance and document similarity was determined by a document distance function. Their experimental result showed good accuracy. The method for Distributed Representation of Words that was employed in this work, which required aggregating short texts with a neural network model on the vast text corpus of a related topic, was its main shortcoming (Waiyama, 2020).

A method for automatically creating topic clusters of reader comments was introduced by (Nunzio, 2016). They provided graph-based methods that make use of the Markov Clustering Algorithm (MCL) to group reader comments into topic clusters and automatically determine the number of

clusters. They employ LDA trained on reader comments to extract subject phrases from each cluster, which are then used to generate a concept-based network using DBPedia. But their approach depended on DBPedia to abstract topics extracted from the clusters. In the other work, the authors proposed a model for grouping short texts. they used Smooth Inverse Frequency (SIF) embeddings to embed short texts, A deep autoencoder to encode and reconstruct the short text SIF embeddings during a pre-training phase and in a self-training phase, they used soft cluster assignments as an auxiliary target distribution and fine-tuned the encoder weights and clustering assignments together (Hadifar et al., 2019).

The Dirichlet Multinomial Mixture model for short text clustering (GSDMM) was given a compressed Gibbs Sampling technique by (J. Yin & Wang, 2016). The DMM is a probabilistic generative model for documents that can accurately and automatically infer the number of clusters. GSDMM can extract the key words for each cluster despite the sparse and high-dimensional nature of brief texts. But their method needs labeled data which is a time-consuming task compared to unsupervised clustering methods. To get the appropriate number of document clusters, a Dirichlet process mixture model-based clustering algorithm typically needs its parameters (i.e., α and β) tuned. Other authors (Dai, 2020) presented a deep embedded technique using an autoencoder of phrase distributed embedding for feature extraction and clustering allocation. They used BERT to execute vector conversion on-demand text. After BERT pre-training with an autoencoder is performed and they Finally used the self-training phase to improve clustering. But their experimental result showed that; it performs better for small datasets only.

A two-stage clustering method was proposed by (Wang, 2016). The method they proposed used a sliding window that moved with the flow of short texts. A hierarchical clustering method was utilized within the sliding window, and a cluster merging method based on information gain was applied between the sliding windows. But their work highly depended on user Dictionaries and network words. another work. (Yang, 2019)For short text clustering, the authors introduced the topic representative term discovery (TRTD) method. By utilizing the proximity and relevance of phrases, they discovered groupings of strongly tied-up topics representative terms using the TRTD approach. The relevance of the topic representative words is measured by their global term occurrences throughout the entire short text corpus, and the proximity of the topic representative

terms is determined by their interdependent co-occurrence. But the co-occurrence of words in the short text is low and they didn't put the method used to handle this issue.

Another author (Caiyan, 2018) developed a concept decomposition approach that detects semantic word communities using a weighted word co-occurrence network gathered from a short text corpus or a subset thereof to produce concept vectors. They employed the k-rank-D k-means-type technique to extract community centers from the word co-occurrence network while identifying semantic word communities. They used Knowledge based document similarity calculation which was called WorldCom; so, for their proposed work additional external documents are needed and they still use high-dimensional text representations, which causes a waste of space and expensive computations.

STCC (Short Text Clustering using Convolutional Neural Networks), without needing any external tags/labels was proposed by (Jiaming, 2015). Word embeddings were researched and fed into convolutional neural networks, with the output units fitting the pre-trained binary code during the training phase, to develop deep feature representations. After acquiring the learned representations, they clustered them using K-means. However, it overlooks the various ways that textual material contributes to clustering and only collects semantic information from the word context, not from any other unsupervised features (Smalheiser, 2019).

The authors used Laplacian Eigenmaps (LE), a dimensionality reduction technique, to find the many similarity information from the original data set and try to reduce the distance between comparable short phrases. They used similar data that LE had retrieved to direct CNN's training. LDA and Topic2Vec then use the brief text data to create the topic's semantic characteristics. To train the short text representations, the CNN model concatenates word embeddings with associated topic semantic characteristics. On the final representations, the K-means algorithm is used to execute clustering, and the effectiveness of the clustering is assessed using the metrics accuracy and normalized mutual information (Chen, 2019). The iterative classification was suggested to improve the clustering quality by (Rakib, 2017). Iterative classification uses outlier removal to produce outlier-free clusters from the clustering of brief texts produced by any clustering algorithm. Then, based on the cluster distributions of the non-outliers, it trains a classification

algorithm. Iterative classification reclassifies the outliers using the training classification model to provide a fresh set of clusters. Repeating this a few times allows them to get a much better clustering of texts. Create the initial clusters using dense and sparse similarity matrices, k-means, and hierarchical clustering. However, because of the lengthy calculation time needed to execute the combination of those methods iteratively, this strategy is difficult to apply to huge datasets.

Another work used topic modeling to discover latent/hidden topics from a collection of Amharic short texts through machine learning. They investigated the LDA method approach to cluster short Amharic texts with and without word embedding as feature extraction and they accurately extract latent topics. they used Skip-gram Word2vec for word representation and Spherical K-means for clustering short texts (Assefa, 2020). Even though LDA has been extensively used for normal text documents, their suggested method did not take word semantic similarity into account. However, for reliable determination, these techniques often need at least a few hundred words. Furthermore, these techniques ignore the word context and just use latent subject information. These techniques disregard the texts' word order information.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

This chapter explains a research methodology to build a dataset, design, and implementation of topic clustering on Amharic comments to achieve research objectives and answer the research question. The proposed research passed through phases; data collection, preprocessing, text representations, text similarity calculation, text clustering, and evaluation. The first phase of this work was collecting comments on Amharic news which contains different topics like sports, entertainment, health, politics, and science and technology. After those datasets were collected, the second task was text preprocessing, since the proposed model is unsupervised learning, there were no data annotation processes. So, text preprocessing was the second step. The preprocessing starts by tokenizing the sentence into separate tokens. Splitting the text into a set of tokens is referred to as tokenization (usually words). This procedure determines where a written text's borders are. The Amharic language uses several punctuation marks which demarcate words in a stream of characters which include. After the individual tokens/terms are identified, stop-word removal will be done to remove non-concept bearing terms from the sentence. Once non-concept bearing terms are removed morphological analysis will be used to extract the stem of a word that will have prefix, suffix, and infix derivations.

Document representations of short Amharic texts were the third step. For the proposed work, BERT has been used as a word embedding tool. But no BERT is pre-train by Amharic languages. So, we have used a cross-lingual transfer learning approach for contextual sentence embedding of Amharic comments. After short texts are represented in vector form the fourth phase was document similarity calculation to group short texts based on their contextual similarity. And then finally documents were clustered based on their topic category.

The experiments were conducted by comparing the two models in terms of clustering results and time complexity to cluster an equal number of datasets. And, we have conducted experiments by

using different hyperparameter values for both clustering algorithms. We conducted experiments on different hyperparameters to select the best fit hyperparameter for mini-batch K-means and fuzzy c-means. We used an equal number of datasets for both algorithms and the experimental results were compared against running time complexity and clustering quality. The experimental results were evaluated in terms of different clustering evaluation metrics. In general, using the preprocessed dataset the actual detection of Mode was developed and evaluated.

For the proposed research we employed an experimental research approach. The use of experimental research methodology allows researchers to examine and comprehend the effects of various variables on the research methods used to arrive at a solution. Different factors are modified in this research process to see how they affect other variables. Datasets, experimental settings, and model hyperparameters are the variables in this study (Maxion, 2009). These variables are modified in our research work to see how they affect the outcome. Experimental research methodology, in general, aids us in measuring and analyzing the impact of factors on our research.

3.2 Dataset Collection

The dataset for the proposed work was collected and prepared by crawling comments from different news agencies' websites. We have used a Facebook comment extractor, and Twitter API for collecting comments from Facebook and Twitter respectively. All comments on a Facebook page can be exported to a CSV/Excel file using the Facebook Comment Extractor / Scraper Software Tool. We easily collected Amharic comments on Facebook Page postings, filtered them, and exported them to a file using Facebook Comment Extractor. Twitter API enabled us programmatic access to Twitter in unique and advanced ways.

The main data sources of this thesis were Fana Broadcasting Corporation, Walta TV, Ethiopian Broadcasting corporation, EBS, ARTS TV, Amhara TV, Balageru TV, and Addis TV. In general, comments were collected from all Amharic news agencies' websites which are available now. We collected comments from the Facebook, Twitter, and Instagram pages of those news agencies. The

collected comments were 7000 in number and clustered into five categories. Those categories were selected by analyzing comments manually.

3.3 Development Tool

For implementation and data analysis, we have used the Python programming language. The general-purpose programming language Python is incredibly abstract. Code readability is given top attention, and there is a lot of indentation used. Programmers may write clear, logical code for both small and large projects with the aid of its language features and object-oriented methodology. Python is intended to be a simple programming language. In comparison to other languages, it contains fewer syntactical structures. Python has a lot of NLP packages, plus it's a powerful text processor. Python is an extremely abstract general-purpose programming language. Python was chosen throughout the development process of our research because it has a robust standard library, multiple open-source frameworks and tools, and code that is understandable and manageable (Interviewbit, 2022). Different python software Numpy, pandas, and matplotlib were used to analyze and tabulate the created model results. Transformers are used to import BERT Model for word embedding. And by receiving inputs as a multi-dimensional array called Tensor, TensorFlow allowed us to create dataflow graphs and structures to define how data goes through a graph.

Numpy: Numpy is an acronym for "Numerical Python" or "Numeric Python," and it is a Python package. We may perform rapid mathematical operations on arrays and matrices with this Python library. By offering arrays and matrices as representations of a dataset, Numpy completes the Python Machine Learning ecosystem together with other Machine Learning modules like sklearn, Pandas, Matplotlib, TensorFlow, and others. To represent our text collection numerically for this work, we used Numpy.

Pandas: Like Numpy, Pandas is one of the most widely used Python libraries for NLP studies. It provides user-friendly data analysis tools and high-performance structures. In contrast to the Numpy library, which offers objects for multi-dimensional arrays, Pandas provides an in-memory 2D table object called Dataframe. It has column and row labels, just like a spreadsheet. In this study, we loaded and worked with our dataset using these Python packages.

Matplotlib: Matplotlib, together with its Numpy numerical mathematics library, is a visual charting library for the Python programming language. Various outcomes are plotted using this during the model's training. It is employed in the Python programming environment to display various graphs. Plotting the training and validation losses and accuracies was done using this library.

Different applications are available for source coding, writing, and running the code such as PyCharm, Jupiter Notebook, Python Anywhere, and others. Python Anywhere has a solid track record of meeting SDLC requirements from beginning to end. With the help of this PaaS (Platform as a Service), you may create, launch, and host Python applications online. As an IDE, PyCharm provides user-friendly auto-completion, suggestions, PEP8 checks, and other tools that improve code quality. Among other capabilities, you may count on it for clever automatic code rewriting, testing support, and code inspections (SIYAL, 2022). We used Jupiter Notebook for source coding, writing, and running the code. Jupiter Notebook was a free and open-source web application that lets us create and share documents with live code, equations, visualizations, and narrative text. And it is very simple to use and understand its usage than the other tools.

3.4 The architecture of Topic clustering on Amharic Comments

A conceptual model called system architecture outlines the viewpoints, organization, and behavior of the system. A system architecture is a representation and description of how a system works and interacts with other system components, to put it another way. The entire system is made up of components and subsystems that work together to create the system that was intended to be in the first place. When we talk about system components, we are talking about the hardware and software that make up the system, as well as their interaction and data transmission and production. The system architecture diagram is a diagrammatic representation of the system architecture. This graphic shows the parts of the system and how they work together to make the system work (Geeksforgeeks, 2021a).

The general architecture of topic clustering on Amharic comments using BERT embedding and partitioning-based algorithms is shown in Figure 1 below. The general architecture contains five

components that are preprocessing, word embedding, text similarity calculation, applying clustering algorithms, and finally Evaluating the clustering algorithms based on the result. The preprocessing component contains; tokenization, normalization, stopword removal, stemming, and lemmatization. Following this, a transfer learning-based technique known as contextual word embedding employing BERT (Bidirectional Encoder Representations from Transformers) was used. By using those vectorized sentences; similarity calculation was conducted using cosine similarity metrics. Finally, train and build clustering models using mini-batch k-means and fuzzy k-means clustering algorithms. And the last was to test the model using evaluation metrics. The main aim of this work was to cluster Amharic comments to their belonging topics.

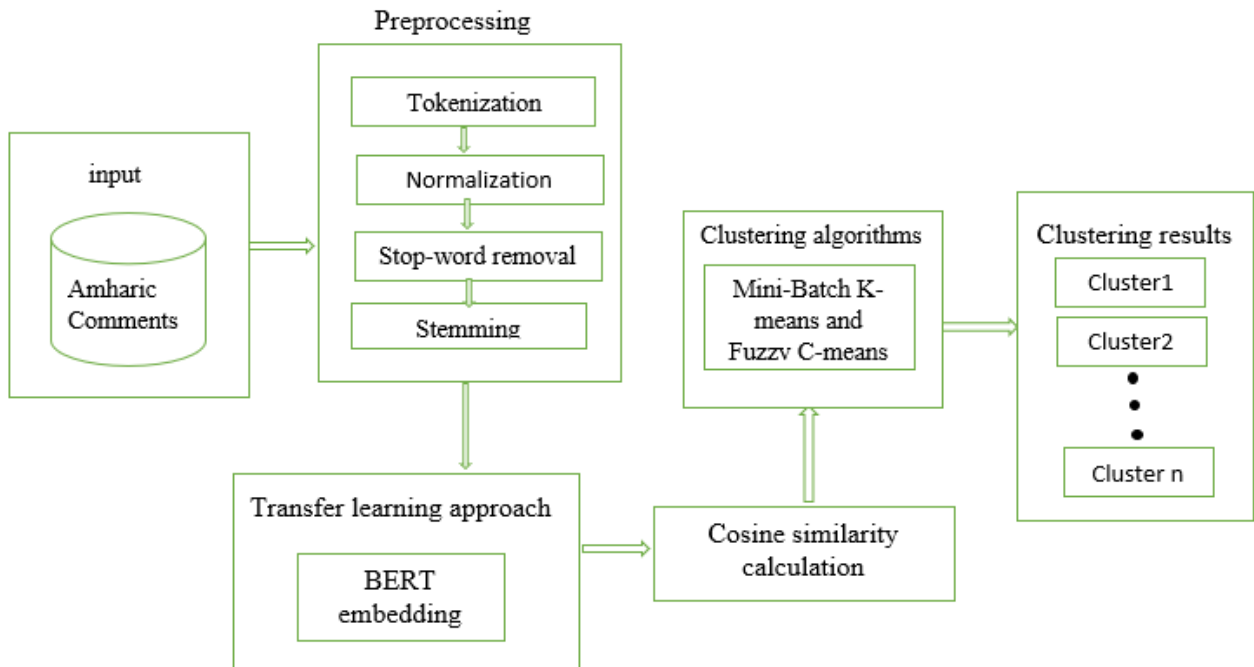


Figure 2 Architecture of Amharic Comments Topic clustering

3.4.1 Text Preprocessing

The preprocessing starts by tokenizing the sentence into separate tokens. After the individual tokens/terms are identified, stop-word removal will be done to remove non-concept bearing terms

from the sentence. Once non-concept bearing terms are removed morphological analysis will be used to extract the stem of a word that will have prefix, suffix, and infix derivations.

Tokenization

The first step of preprocessing is tokenization. In Amharic, a sentence separated is with another sentence by space (ክፍትባታ) and ends with a አራት ነጥብ (:). Splitting the text into a set of tokens is referred to as tokenization (usually words). This procedure determines where a written text's borders are. The Amharic language uses several punctuation marks which demarcate words in a stream of characters which include (፤ጽርብሰረዝ) deribsereze' (፣ነጠላሰረዝ), netelaserez" (!ቃልአጋኖ), exclamation mark and (?ጥያቄምልክት) question mark.

Algorithms for Tokenization

Input: Amharic comments dataset

output: list of terms/words

start

Read file name

Read list of punctuation marks and numbers

for each file name in corpus:

 Split into word stored in list

 for Each element of the list do:

 if there any punctuation or/and number in the list:

 Discard the punctuation or/and number

 else

 Return list.

 Endfor

Endfor

Figure 3: Tokenization algorithm

Normalization

There are homophone characters in the Amharic writing system, which means that letters with the same sound have different symbols for example; commonly, the characters ስ and ሥ are used interchangeably as ስራ and ሥራ to mean “work”. This increases the number of features that will be extracted without providing any benefits. Another example is the use of the characters, where there is no set rule on whether to use or and people frequently do. In these pre-processing subtasks, a character normalization script was developed by designing a common representation character list. All ‘ha’ sound varieties have changed to ‘ሀ’ type representation, ‘se’ varieties have changed to ‘ሠ’ type representation, ‘we’ varieties have changed to ‘ወ’ type representation, ‘a’ varieties have changed to ‘አ’ type representation and ‘te’ varieties have changed to ‘ፀ’ type representation. The normalization algorithm normalizes Amharic characters with the same sound and different representations into a common representation. The normalization algorithm is presented in Figure 4.

Algorithms for Normalization

Input: list of words

Output: normalized list of terms

Start

Read list for Alphabet normalization

For lists in list for corpus:

 If Find character with double values:

 Replace with list for Alphabet normalization

 Else:

 Return list

Endfor

Figure 4 Normalization algorithm

Stopword Removal

Stop words are words that show up most frequently in sentences but don't contribute to categorizing feelings. The stop words in Amharic are unique, much like in other languages. The most frequently used stop-words in Amharic documents are “ሆ”, “ሁሉ”, “ነው”, and “ነበር”, and others like “እና”, “ ወይም”, “ ውስጥ”, “ ላይ”. There are many available packages like NLTK for High resourced languages to remove stop words without requiring a list of stopwords but in the case of low-resourced languages like Amharic, no model were developed yet. So, a list of stop words should be identified and listed. Each language possesses its list of stopwords that can be adjusted according to the problem and having these we have prepared an Amharic stopword list by compiling from different sources like papers done on Amharic language (MULUGETA, 2021). The algorithm is presented in Figure 5.

```
Algorithms for Stopword removal  
Input: list of words  
Output: non-stopword list of terms  
Start  
Read list of stopword file  
For list of words in corpus:  
    If list is in stopword file:  
        Eliminate list  
    Else  
        Return list  
Endfor
```

Figure 5: Stop-word removal Algorithm

Stemming

The pre-processing subtask deals with the handling (elimination) of affixes from a word to extract the stem of a word are stemming. Stemming is a crucial preprocessing subtask that gets rid of suffixes and prefixes of a word while keeping its root, also called the "stem". In this work, we have constructed a list of Amharic language suffixes and prefixes by collecting from different sources like Amharic books and research papers done on this language (MULUGETA, 2021).

Stemming eliminates all word variations by using only the word's single stem. Among the variations that will be stemmed are plurals, ing-forms, third-person suffixes, past-tense suffixes, etc. For Amharic languages stemming is removing suffix, infix, and prefixes, such as “ዎች”፣ “አችን”፣ “አች”፣ “ከ”፣ “ን”፣ “በ”፣ “ስለ”፣ “ከ”... Amharic is a morphologically rich language which requires good work for handling its morphologies.

Algorithms for prefix removal

```
Input: list of terms
Output: list of affix free terms
Start
Read list of prefix-list file
For list of terms in corpus:
    If list starts with prefix-list:
        Eliminate prefix-list
    Else
        Return list
Endfor
```

Figure 6 prefix Removal algorithm

Algorithms for suffix removal

```
Input: list of terms
Output: list of affix free terms
Start
Read list of suffix-list file
For list of terms in corpus:
    If list ends with suffix-list:
        Eliminate suffix -list
    Else
        Return list
Endfor
```

Figure 7 suffix removal algorithm

3.4.2 Semantic Word Representation

Due to the sparse, high-dimensional, and noisy characteristics of a short text corpus, effective representation learning is essential for short text clustering. Words with a common context (semantic similarity) are mapped next to one another in the latent space with similar embeddings, which depicts the spatial position of the word along with the word vector (Waiyamai, 2020).

There are many word embedding techniques like bag-of-words, Term frequency-inverse document frequency, Word2Vec, Glove, FastText, and the like. Bag-of-words and Term frequency-inverse document frequency cannot capture semantics or the relation of words therefore they are not appropriate for semantic word representation. For each word, the Word2Vec embedding approach only offers a single, independent embedding vector. Since the representation is only kept as a single vector per word. This restricts the ability to understand a word's meaning across two contexts. GloVe utilizes the standard word-like tokens. It prevents out-of-vocabulary tokens but splits

completely unfamiliar words into characters. Glove and Word2vec are word-based models, which means that they use words as input and produce word embeddings. Additionally, they are unable to encode words that are unfamiliar or uncommon. Because word embedding models do not consider the sequence in which words appear, the syntactic and semantic meaning of the sentence is lost.

Recently, there has been a lot of discussion on the Bidirectional Encoder Representations from Transformers (BERT) paradigm in the fields of machine learning and natural language processing. The model architecture of BERT is substantially different from the methods mentioned above and cannot be compared. Based on the placement of a word in a phrase or sentence, BERT creates a vector for that word. With the help of a transformer, an attention mechanism that can understand the relationships between words in context, it learns how to represent text (or sub-words). The context will determine which word embeddings BERT can understand. Particularly, word embedding differs if the word has several meanings in various phrases (Kenton, 2019; Yang, 2019). The current input sentence affects the contextual word vectors that BERT produces. A word could theoretically have an endless number of vectors because of the words that are present around it in the input text. The weights available from one or more of the 12 levels can be used by the BERT model to derive not only word representations but also representations of sentences (Ankiit, 2020).

With more condensed, low-dimensional, and continuous elements added by BERT, the expressiveness of brief text representations has significantly increased. For the proposed work, BERT has been used as a word embedding tool. But no BERT is pre-train by Amharic languages. So, we have used a cross-lingual transfer learning approach for contextual sentence embedding of Amharic comments.

3.4.2.1 Cross-Lingual Contextual Word Embedding

Contextual word embeddings reflect a word utilizing its context, in contrast to word embeddings. It is possible to learn a model using supervised data in one language and utilize it for a range of tasks in another without any explicit cross-lingual signal. Surprisingly, the cross-lingual transfer

is successfully made possible by multilingual language models like XLM, mBERT, and XLM-R. Masked language modeling (MLM) and Translation Language Modeling (TLM) are used to train XLM. Keep in mind that TLM needs a dataset of parallel sentences, which may be challenging to obtain (Reimers, 2019).

RoBERTa is available in multiple languages as XLM-RoBERTa. Its pre-training data includes 2.5 TB of filtered Common-Crawl data with 100 languages. It was specifically pretrained using Masked Language Modeling (MLM). The model receives a sentence as input, randomly selects 15% of the words, and then processes the entire masked sentence to determine which words were masks. The model acquires an internal representation of 100 languages from which it can derive features useful for further tasks (Conneau, 2019; Ruder, 2019). XLM-RoBERTa requires additional GPU memory space than BERT models. In addition, researchers proved that mBERT showed very impressive performance for cross-lingual transfer learning in different downstream tasks which are competitive performance with XLM-R. And also improving the inference speed taken by XLM-R for cross-lingual transfer learning without performance degradation (András, 2021; Ruder, 2019; Wu, 2022). So, we used mBERT for the cross-lingual contextual embedding of Amharic comments.

For the cross-lingual transfer learning, we have prepared translated dataset which contains both English and Amharic translations. This is used to train and test the BERT model. We have developed the cross-lingual contextual word embedding model by using sentence transformer SBERT as the teacher model and mBERT from the transformer as the student model. The reason behind combining those models was to take the advantage of each model. Directly Applying mBERT for the cross-lingual contextual embedding of Amharic comments were costly related to time and space usage than sentence transformer models. This is due to the mBERT having a tokenization step before embedding words to vector forms and this creates complexity when the number of datasets is high. On the other hand, directly using SBERT for the Amharic dataset was not possible because it trained only English languages and needs parallel data for cross-lingual transfer learning.

The solution for this was making mBERT directly encode the dataset by eliminating this step using SBERT as a teacher model. SBERT is a sentence transformer model which is developed for extracting features purposes only without any tokenization process. So, combining those models allow us to create a model which has cross-lingual contextual embedding ability and can directly extract features from the dataset. We used “bert-base-nli-stsb-mean-tokens” from the sentence transformer as a teacher model which is developed as a feature extraction model and pretrained on English languages only. It can be used for tasks like clustering or semantic search because it maps sentences and paragraphs to a 768-dimensional dense vector space (Reimers, 2019).

The student model was “bert-base-multilingual-uncased”. BERT multilingual base model (uncased) model that has been pre-trained using masked language modeling (MLM) targets the top 102 languages with the biggest Wikipedia. This model is uncased and it does not make a difference between capital and small letter of the same word. In the training set, the model learns an internal representation of the languages that can be used to extract features for subsequent tasks. It was pretrained with two goals, to be more precise. Masked language modeling (MLM): the model must predict the words that are hidden by randomly masking 15% of the input words in a sentence before running the complete sentence through the model. Next sentence prediction (NSP): During pretraining, the model concatenates two masked sentences as inputs. They occasionally match sentences that were next to one another in the original text, and sometimes they do not. The model must then determine whether the two sentences followed one another (Kenton, 2019; Lee, 2018).

The concept was we initially used sentence transformer BERT to generate sentence embeddings in English sentences, which we refer to as the Teacher model. Then, for the Amharic language, we have developed a second model called Student, which attempts to emulate the Teacher model. To put it another way, the original English sentence was trained in the Student model to produce a vector that is identical to that of the Teacher model. Following training, the Student model was capable of encoding sentences in both English and the Amharic language.

3.4.3 Text similarity calculation

The goal of the short text similarity (STS) measurement is to assess how similar two pairs of short texts are to one another. The similarity should be studied not only from a lexical standpoint, which only looks at character sequences but also from a semantic standpoint. For the proposed, work we have used String-based similarity calculation. BERT is the basis for choosing this approach. In word embedding, BERT can learn the contextual relationships between words (or sub-words). As a result, the contextual issue was resolved, and words are embedded based on their context rather than cooccurrence. It also enables the use of String-based similarity calculation, which compares sentences based on the character or string sequence used to construct them. This method's primary benefit is that it does not primarily depend on a language and does not use an external semantic net or corpus to calculate similarity. The semantic similarities are usually language and domain-dependent, so they do not apply to all languages for different country languages (Elton, 2022). In chapter two, the formula for calculating the cosine similarity between two vectors was described.

3.4.4 Clustering Algorithms

As discussed in chapter two different short text topic clustering techniques are available, such as density-based clustering, hierarchical clustering, partitioning clustering, topic modeling, and others. The density-Based Clustering method has a high running time because of its non-linear temporal complexity (Waiyamai, 2020). The drawback of density-based Clustering is that it is susceptible to the selection of epsilon and minimum points, especially if clusters have various densities. Sparser clusters will be noise if eps are set to too low. Denser clusters may be combined if eps are too large. This suggests that a single eps value might not be enough if there are clusters with variable local densities. This method may identify any cluster shape, whether it be spherical or arbitrary. However, certain data points will be identified as noise and not assigned to any clusters to ensure cluster completeness (Lasek, 2019). Regardless of whether a cluster is spherical or random in shape, this approach may identify it. To guarantee cluster completeness, some data points will be labeled as noise and not allocated to any clusters (Amandeep Kaur Mann, 2013). Topic modeling requires very appropriate preprocessing to perform best and the additional

contextual document is required after terms are selected by topic modeling techniques like LDA for short text clustering (Pascual, 2019).

Partitioning-based clustering is more desirable because it has lower time complexity than the other approaches. The main drawback of this clustering algorithm it requires prior specification of many clusters. A prominent and commonly used distance-based partitioning clustering method is K-means clustering, particularly for text document clustering (Niraj N Kasliwal, 2012). In terms of cluster shape, partitioning clustering produces spherical shapes of clusters while density-based and hierarchical clustering produces all cluster shapes. Density-based clustering can identify the noise in a data collection, which excludes some documents from any cluster in terms of cluster completeness. All documents that are part of a cluster are determined using a partitioning clustering technique since it does not identify any noise. In terms of time complexity, partitioning clustering is linearly faster than density- and hierarchically-based clustering.

For the proposed work partitioning clustering approach specifically, mini-batch k-means and fuzzy k-means clustering algorithm (also called fuzzy c-means) was tested and the one performing well was selected. K-means is very good for cluster completeness, ensuring that every text document is assigned to a cluster (Tapas Kanungo, 2022). In addition, k-means require lower complexity time, it is simple to use, and have less-complicated parameter settings (Waiyamai, 2020). K-means computation time grows with the size of the datasets being researched because it needs to store the complete dataset in the main memory. Consequently, many methods for decreasing the algorithm's time and space cost have been put forth. The Mini batch K-means algorithm is another method. When utilizing fuzzy clustering, sometimes referred to as soft clustering or soft k-means, each data point might belong to multiple groups. Through the use of similarity metrics, clusters are found. There are three similarity metrics: proximity, connectedness, and intensity (Subas, 2022). The main factors considered for choosing clustering algorithms in this study are the ability to handle large-dimension data, cluster shape, cluster completeness, and low time complexity.

Mini Batch K-means algorithm

The Mini-batch K-means clustering algorithm is a machine learning variation of the traditional K-means technique. It gathers a random sample of the data to update the clusters after each iteration and maintains the data in memory in brief, random, fixed-size batches. When working with huge datasets, it performs better than the standard K-means algorithm because it does not cycle through the entire dataset. On each cycle, it creates random batches of data to be stored in memory before gathering a random batch to update the clusters. The Mini-batch K-means algorithm's main advantage is that it reduces the computational cost of cluster finding (KHARWAL, 2021).

The main goal of the Mini Batch K-means method is to temporarily store small random batches of data. The clusters are updated using a fresh random sample from the dataset at the beginning of each iteration, and this procedure is continued until convergence. Each mini-batch uses a convex combination of prototype values and data to update the clusters, with the learning rate decreasing as the number of iterations rises. The learning rate is equal to the inverse of the number of data assigned to a cluster during the procedure. Convergence can be seen when there are no changes in the clusters for several iterations in a row since the impact of incoming data decreases as the number of iterations grows. The technique uses tiny, randomly chosen batches of the dataset for each iteration. Each piece of data in the batch is categorized into one of the clusters based on the centroids' prior placements. The cluster centroids' coordinates are then changed using the fresh points from the batch. The update is made via gradient descent, which is much faster than a batch K-Means update (Geeksforgeeks, 2021b).

Fuzzy K-Means Clustering

The Fuzzy K-means clustering algorithm is another partition clustering algorithm that was proposed for this study. Fuzzy K-Means clustering, also known as Fuzzy C-Means clustering, is a variant of K-Means clustering. The Fuzzy K-means clustering algorithm uses several points that are exclusively part of one cluster. The algorithm Fuzzy K-Means is identical to K-means, a popular simple clustering algorithm. The only difference is that instead of assigning a point to one

cluster exclusively, it may have fuzziness or overlap between two or more clusters (Edureka, 2019). The following are the salient features of fuzzy k-means:

- One point in a soft cluster may be a member of several clusters, each of which has a different affinity value.
- The affinity is proportional to the separation between that point and the cluster centroid.
- Select the number of clusters.
- Assign each point's coefficient at random to create the first k clusters.
- Continue doing this until the algorithm converges.

3.4.5 Clustering Performance Evaluation

The final stage of this research was assessing the developed model's performance. Using the Adjusted Rand Index, homogeneity-score, Completeness, V-measure, Silhouette Coefficient, and Adjusted Mutual Information, we evaluated the clustering quality of the proposed work. If only data points from the same class are present in each cluster of clustering, the clustering is homogenous. The clustering algorithm's ability to satisfy a critical requirement—that only samples from one class should be in a cluster—can be assessed using this score. Low numbers between 0 and 1 suggest low homogeneity. Completeness is the primary idea, and we check to see if each class label's data point is in the same cluster. The completeness score ranges from 0.0 to 1.0, with 1 denoting entirely comprehensive labeling. The second metric for evaluation is perfect clustering would receive a score of 1, while poor clustering or independent clustering would receive a score of 0 or lower, according to the adjusted Rand Index. The silhouette coefficient varies from -1 to 1, with 1 indicating distinct clusters that are widely separated apart. Zero indicates that clusters are unrelated, or that the distance between clusters is not significant, and -1 indicates that clusters were incorrectly assigned (Geeksforgeeks, 2019; Tutorialspoint, 2019).

CHAPTER FOUR

EXPERIMENTATION, RESULT, AND DISCUSSION

4.1 Introduction

In this chapter, we have presented the dataset used, the algorithm we have experimented with, the experimentation setups, and the results of the experiments done. We have collected a total of 704 KB Dataset. Among these datasets, 37.5 KB was Translated Dataset and 664 KB was Amharic comments. Bidirectional Encoder Representations from Transformers (BERT) from hugging face are used for contextual word embeddings with a selected hyper-parameter (train-batch-size, max-seq-length, learning rate, epoch, etc.). Mini-batch K-means clustering with a selected hyper-parameter (init, number of clusters, random-state, maximum-iteration, etc.) and fuzzy c-means used as a clustering algorithm.

4.2 Experimentation Setup

4.2.1 Dataset Description and Distribution

The dataset we used for model development contains a total of 704 KB size. Among those datasets, 37.5 KB are Translated Dataset which contains pairs of English and Amharic sentences for transfer learning approach and 664 KB are Amharic comments to news for topic clustering. There is typically no separation of training and test data in cluster analysis. We cannot "train" when performing cluster analysis without labels. Train-test splitting, a machine learning technique, is used to prevent overfitting in training. However, there is no overfit if the algorithms are not learning labels.

4.2.2 Environment and hyper-parameter Setups

We have done our experimentation by setting environmental and hyper-parameter setups. We have used an HP laptop with Intel(R) Core (TM) i5-4300M CPU@ 2.60GHz 2601 MHz processor and 4GB RAM specification. For evaluating BERT, we have used the hyper-parameter shown in

Table 1. for mini-batch k-means we have used the hyper-parameter shown in Table 2. The result of each experiment was documented in the appendix.

Table 1 Hyper-parameter setups for BERT embedding.

No.	Hyper-parameter	Setup 1 (Experiment 1)
1	maximum sequence _length	512
2	Batch-size	16
3	epochs	5
4	Learning-rate	3e-5

The above hyper-parameters are selected for experimentation after trying many setups and the best setup which performs well was selected as our model hyper-parameter. We have developed the models using the hyper-parameter setups and the result that we are presented is based on these setups. During testing, we discovered that changing the hyperparameter values improves the model's performance.

The maximum sequence length determines the input's maximum token length. The input tokens are padded or shortened depending on their value. we can decide its value based on the end task. We know that BERT has a max length limit of tokens is 512, so we use this max length limit for our transfer learning model because the translation dataset contains Amharic comments with their translation in English makes some lines of comments to have long-sized tokens.

The number of samples processed before the model is updated is referred to as the train batch size. The batch size is always a multiple of two, for example, 16, 32, 64,128, 512, 1024, or 2048. Because there are 12 layers in the BERT and each word is encoded into a floating-point vector with a size of 768, we must take maximum-sequence-length into account when determining the train batch size. The data might not fit into GPU RAM with batch size 32 if the maximum 512 lengths are utilized. Next, a decrease to 16 is mandatory. When the maximum length is 128 or 256, 32 is a decent choice (Devlin, 2019). So, we used batch size 16 because we used maximum-

sequence-length 512 and this intermediate batch size 16 makes the proposed transfer learning model perform well.

The epoch specifies how many times we want to pass the entire dataset. Its value should be more than one. Too small epochs make the proposed model not learn the training dataset enough and too large epochs make the model overlearn. Both cases reduced the accuracy of the model. The BERT paper suggests a few heuristics for fine-tuning BERT and they suggest that Epochs from 2 to 5 are enough for BERT embedding (McCormick, 2019; Swatimeena, 2020). We set epochs to 5 to allow the BERT model to learn the training dataset accurately.

The BERT model pre-training itself used a higher learning rate. So, we should avoid using a learning rate that is either too high or too low. The literature demonstrates that for BERT to overcome the catastrophic forgetting issue, a lower learning rate, such as $2e-5$, is required. The training set is unable to converge with an aggressive learning rate of $4e-4$ (Devlin, 2019)(Devlin, 2019). The tendency of an artificial neural network to entirely and abruptly lose previously learned information upon acquiring new information is known as catastrophic interference, sometimes known as catastrophic forgetting (Zuhua Dai, 2020). We used learning rate $3e-5$ which is an intermediate learning rate between lower and aggressive learning rates.

Table 2: Hyper-parameter setups for mini-batch K-Means

No.	Hyper-parameter	Setup 1 (Experiment 1)
1	Number of Clusters	5
2	Random-State	64
3	Batch-Size	64
4	Maximum-Iteration	300
5	Reassignment-Ratio	0.01
6	Max-no improvement	10

The first parameter for mini-batch k-means is the number of clusters. We have considered two methods to determine the cluster numbers. The first was manually analyze our dataset and we

decide that the number of clusters is 5. Secondly, we experimented with the Silhouette coefficient value of the model by using different cluster numbers (2,3,4,5,6,7, and 8) and the optimal Silhouette value is obtained when $K = 5$. So, our model's cluster number is five. The results of the experiment were documented in Appendix D.

The other hyperparameter Random state is useful if we want to reproduce exact clusters repeatedly. We can set it to any number and we want to see the changes. We decide to use 32 because we conducted experiments on three different values (0, 32, 64) of random-state and the model's performance is high at value 32. The results of the experiment were documented in appendix D. We used the Maximum number of iterations of the mini-batch k-means algorithm for a single run of 300 because we conducted experiments on values (100, 300, and 1000) of a maximum number of iterations and the results showed that there is no any variation on the results of those three experiments. And this is an intermediate value that is not a too low or high maximum number of iterations.

Table 3: Hyper-parameter setups for Fuzzy C-Means

No.	Hyper-parameter	Setup 1 (Experiment 1)
1	Number of Clusters	5
2	Random-State	32
3	Fuzzifier	2
4	Maximum-Iteration	300
5	Metric	Cosine similarity

We conducted experiments on different batch-size values (32,64,128 and 256) and we observed in practice that when using a small and a larger batch there is no difference in the quality of the model instead there is a slight difference in the execution time. The small batch size has less execution time than the large batch size. So, we take batch-size 64 for the model. Reassignment-Ratio controls the percentage of the total counts that may be used to reassign a center. The model will take longer to converge but should converge in a better clustering if the value is bigger because it makes low count centers easier to reassign. Convergence, however, could result from an excessive

value. Therefore, we used the default value of reassignment-ratio 0.01, which improves the performance of the model. Early termination under the maximum-no improvement control is based on the number of mini-batches that have been run in a row without improving the smoothed inertia. We used the default value 10 because of the mini-batch size was set to 64.

We set the same number of clusters and Random-state value for Fuzzy C-means with mini-batch k-means because we conducted experiments on a different number of clusters (2,3,4,5,6 and 7) for determining an optimal number of clusters. The Maximum-Iteration set to 300 because we conducted experiments on 100, 300, and 1000 maximum iteration values. The results were low at 100 and show the same value starting from 300 with a high score. So, we can set any value starting from 300 and greater.

We have two parameters, μ_{ij} , and c_i , as well as one hyperparameter, m , in the Fuzzy c-means (FCM) clustering algorithm. The membership value, or μ_{ij} , is the likelihood that the j th data point is a member of the i th cluster, and it is confined to the condition that the total of μ_{ij} over C cluster centers is 1 for every data point j . (the same dimension as X). In addition, the fuzzifier, abbreviated as m , determines how fuzzy the cluster boundary should be. Fuzzifier is a parameter that has a significant impact on the performance of the FCM. It is a scalar parameter used to define the degree of fuzziness in the fuzzy algorithm. If $m = 1$, the objective function, which sums the distances between the data points and the cluster centers probability-weighted, is. It denotes that closer data points near cluster centers will have larger weights (Yufeng, 2021). When we set fuzzifier value 1 it works like k-means which is a hard-clustering method that assigns data points to one cluster and when we set it to a higher number it leads to all data points belonging to all clusters numbers (Schwämmle, 2016; Zhou, 2019). So, we start our experiment by setting the values of the Fuzzifier starting from 1.1 up to 2. And our model achieves a high score value when the Fuzzifier value is set to 2 and the result were documented in Appendix E.

4.2.3 Experimental Setup

The experiments were carried out by comparing the two models in terms of clustering results and time complexity to cluster an equal number of datasets. And, we have conducted experiments by using different hyperparameter values for both clustering algorithms. We conducted experiments on different clusters, batch-size, and maximum iteration hyperparameters to select the best fit hyperparameter for mini-batch K-means. Different values of fuzzifier, number of clusters, and maximum iteration were the experimentation variable for fuzzy c-means. We used an equal number of datasets for both algorithms. The experimental results were compared against running time complexity and clustering quality. The experimental results were evaluated in terms of Homogeneity-score, Completeness-score, V-measure, Adjusted Rand Index, Adjusted-mutual information, and Silhouette Coefficient.

4.3. Experimentation Result of Amharic Comments Clustering Model

In this sub-section, we have presented an experimental evaluation of our two models. The first model was using BERT embedding and mini-batch K-means clustering algorithms. The second was BERT embedding and fuzzy c means clustering algorithms. We have evaluated the two models using the same hyper-parameter discussed in the experimentation setup section and the performance of each model is presented below.

Experimental Results of BERT Embedding with Mini-Batch K-Means Clustering Algorithms

We have conducted a total of 18 experiments on the first model by using different hyperparameters values of min-batch k-means. The first experiment was conducted by using different values of cluster-number or K-value (2,3,4,5,6,7) and the experiment result shows that the model performs best at cluster number 5. For the second experiment, we used different batch-size values and the model performs best at batch-size 64. The third experiment was conducted by considering different maximum iteration numbers and we observed that maximum iteration numbers didn't influence the results of the model. The fourth experiment was by using different random-state values and the model performs best at random state value 32. Based on the experimental results of those

hyperparameters we select setups that are listed in Table 2 for conducting experiments on the first model (BERT embedding with mini-batch k-means algorithms). Table 4 below shows the performance of the model in this hyperparameter setup for clustering Amharic comments to news. We have calculated the Homogeneity-score, Completeness, V-measure, Adjusted Rand Index, Adjusted Mutual Information and Silhouette Coefficient of the model are presented in table 4.

Table 4: Experimentation Results of BERT embedding and Mini-Batch K-Means Model

Evaluation metrics	Experiment 1
Homogeneity-score	1.0
Completeness-score	1.0
V-measure	1.0
Adjusted Rand Index	1.0
Adjusted-mutual information	1.0
Silhouette Coefficient	0.998

As we can see in Table 4, we can observe that the clustering evaluation of the models reaches a score value 1.0 of for Homogeneity-score, Completeness-score, V-measure, Adjusted Rand Index, and Adjusted-mutual information. Hyperparameter configurations from Table 2 were used in the experiment. Because the homogeneity score is confined between 0 and 1, high values imply strong homogeneity and low values that are close to 0 suggest less homogeneity, models with homogeneity scores that reach 1.0 indicate that a cluster contains only data from a single class. Completeness is the primary idea, and we check to see if the data points for each class label are in the same cluster. The completeness score ranges from 0.0 to 1.0, with 1 denoting entirely comprehensive labeling. So, the model shows perfect completeness labeling on the experiment because its value reaches 1. The other evaluation metric Adjusted Rand Index indicates perfect clustering would be scored 1 and bad clustering or independent clustering is scored 0 or negative. So, the model's ARI values reach one during the experiment, which shows the model perfectly clusters to their group. The value of the silhouette coefficient ranges from -1 to 1. One indicates that clusters are distinct and spaced widely apart. Assigning clusters incorrectly results in a score of -1, while zero indicates that clusters are indifferent or that the distance between clusters is not

relevant. So, the clustering models have a value of Silhouette Coefficient 0.997 which indicates clusters are well apart from each other in the conducted experiment.

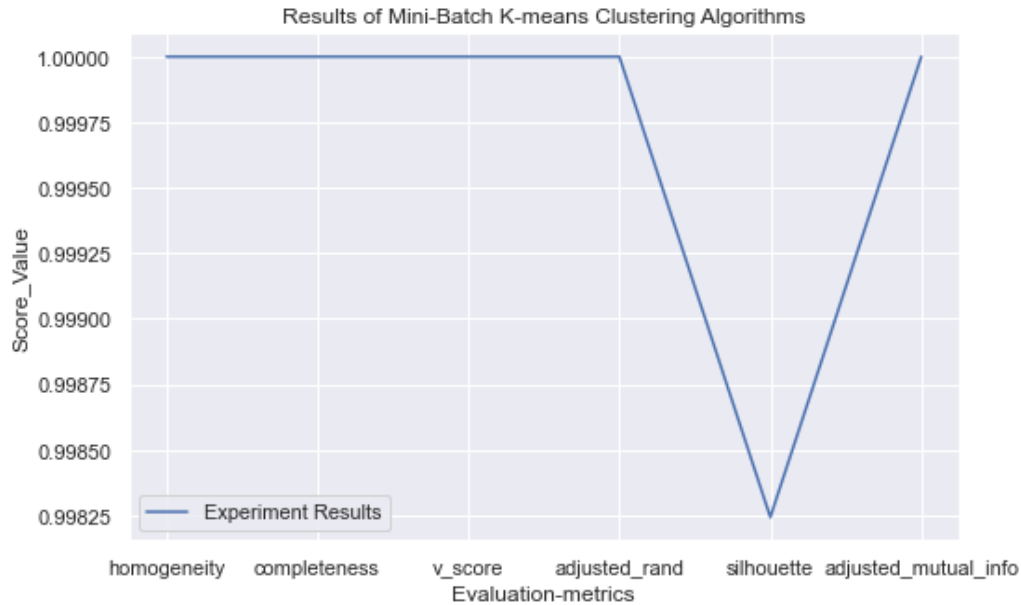


Figure 8: Evaluation Results of Mini-Batch K-Means model

Figure 10 shows the experimental results of the experiments on different evaluation metrics. We can observe the model reaches the top score value on those evaluation metrics and there is not that much variation in the results of those metrics this indicates that the hyperparameter setup for the experiment is the best match to the proposed problem and the dataset used. So, we can conclude that the experimentation setup is the best match to the proposed problem and the dataset used.

We also conducted experiments on different hyperparameter setups. Table 5 below shows the evaluation of the mini-batch K-Means model for different numbers of clusters (k-value). The experiment results show the perfect cluster number for the given dataset is 5 because the model has a high value of V-measure, Adjusted Rand Index, Adjusted-mutual information, and Silhouette Coefficient at k value 5 than the other cluster numbers.

Table 5 Mini-Batch K-Means Model Evaluation Results for Different Cluster Numbers

Evaluation metrics	K= 2	K= 3	K=4	K=5	K=6	K=7
V-measure	0.592	0.823	0.947	1.0	0.936	0.896
Adjusted Rand Index	0.4.03	0.717	0.922	1.0	0.883	0.810
Adjusted-mutual information	0.592	0.823	0.947	1.0	0.936	0.895
Silhouette Coefficient	0.705	0.761	0.934	0.998	0.863	0.774

As we can see from Table 5, we can observe that the model's performance is very low at cluster number 2, very high at cluster number 5, and start decreasing from cluster number 6. Adjusted Rand Index indicates perfect clustering would be scored 1 and bad clustering or independent clustering is scored 0 or negative. So, the model's ARI value reaches 1.0 on cluster number 5 and this shows the model perfectly clusters at K value five than other cluster numbers. The value of the silhouette coefficient ranges from -1 to 1. And 1 indicates that clusters are distinct and spaced widely apart. Clusters are either indifferent or the distance between them is not relevant if the value is 0. Clusters are incorrectly assigned if the value is -1. The silhouette coefficient reaches 0.998 at cluster number five, which indicates clusters are well apart from each other because of this number near number 1. Figure 11 below shows the plot of results of mini-batch k-means with a different number of clusters.

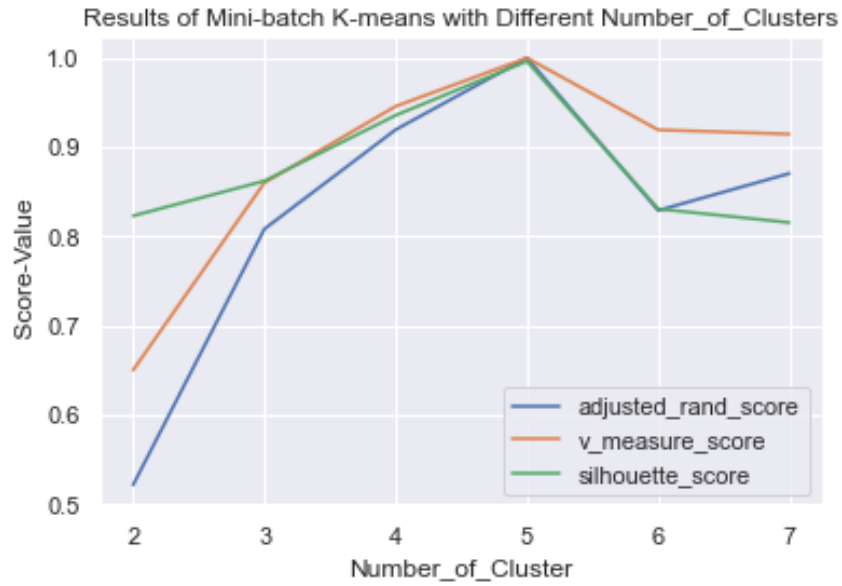


Figure 9 Results of Mini-Batch K-means with Different Number of Clusters.

Figure 9 shows the plot of the results of the first model with a different number of clusters. The experiment results show the perfect cluster number for the given dataset is 5 because the model has a high value of V-measure, Adjusted Rand Index, Adjusted-mutual information, and Silhouette Coefficient at k value 5 than the other cluster numbers. As we observed in figure 13 the clustering result starts increasing from cluster number up to cluster number 5, and the models have a high score value at cluster number 5. But starting from cluster number 6 it drops to 0.88 and 0.86 scores of Adjusted Rand Index and Silhouette Coefficient value. And, at cluster number 7 the evaluation score value drops highly. So, we can conclude that the optimal cluster number for the given dataset and Fuzzy C-Means Model is cluster number 5. The five clusters predicted by mini-batch K-means algorithms contain different topic categories. The first cluster number “0” includes comments about entertainment, cluster number “1” includes comments about health, cluster number “2” includes comments about politics, cluster number “3” includes comments about science and technology, and cluster number “4” includes comments about sport. The screenshots of those results are documented in Appendix C.

Experimental Results of BERT with Fuzzy C-Means Clustering Algorithms

Table 6 below shows the overall performance of the BERT and Fuzzy C-Means model to cluster a total of 7000 Amharic comments into five different clusters. We have calculated the Homogeneity-score, Completeness, V-measure, Adjusted Rand Index, Adjusted Mutual Information and Silhouette Coefficient of the model are presented in the table below.

Table 6 BERT and fuzzy C-Means model testing performance for Amharic comments clustering

Evaluation metrics	Experiment 1
Homogeneity score	1.0
Completeness	1.0
V-measure	1.0
Adjusted Rand Index	1.0
Adjusted mutual information	1.0
Silhouette Coefficient	0.996

As we can see in Table 6, we can observe that the clustering evaluation of the models reaches a score value 1.0 of for Homogeneity-score, Completeness-score, V-measure, Adjusted Rand Index, and Adjusted-mutual information. The experiment was conducted by using hyperparameter setups in Table 3. Models' homogeneity score reaches 1.0 which indicates a cluster contains only samples belonging to a single class because the homogeneity score is bounded between 0 and 1, with high values indicating high homogeneity and low value which is near 0 indicating less homogeneity. Checking if the data points for each class label are in the same cluster is the basic concept of completeness for each class label. The completeness score ranges from 0.0 to 1.0, with 1 denoting entirely comprehensive labeling. So, the model shows perfect completeness labeling on the experiment because its value reaches 1. The other evaluation metric Adjusted Rand Index indicates perfect clustering would be scored 1 and bad clustering or independent clustering is scored 0 or negative. So, the model's ARI values reach one during the experiment, which shows the model perfectly clusters to their group. The value of the silhouette coefficient ranges from -1 to 1. One indicates that clusters are distinct and spaced widely apart. Zero indicates that clusters are unrelated, or that the distance between clusters is not significant, and -1 indicates that the clusters were incorrectly assigned. So, the clustering models have a value of Silhouette Coefficient 0.996 which indicates clusters are well apart from each other in the conducted experiment.

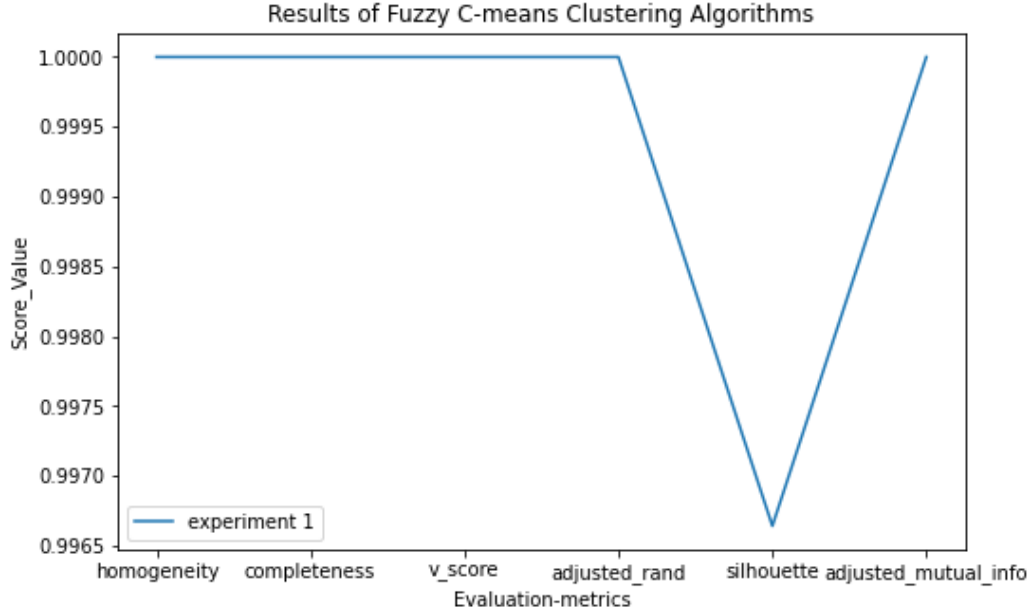


Figure 10: Fuzzy C-means Model Evaluation Result for Three Experiments

Figure 10 shows the experimental results of the experiments on different evaluation metrics. We can observe the model reaches the top score value on those evaluation metrics and there is not that much variation in the results of those metrics this indicates that the hyperparameter setup for the experiment is the best match to the proposed problem and the dataset used. So, we can conclude that the experimentation setup is best to match the proposed problem and the dataset used.

We also conducted experiments on different hyperparameter setups. Table 7 below shows the evaluation of the Fuzzy C-Means model for different numbers of clusters. The experiment results show the perfect cluster number for the given dataset is 5 because the model has a high value of V-measure, Adjusted Rand Index, Adjusted-mutual information, and Silhouette Coefficient at cluster number 5.

Table 7 Fuzzy C-Means Model Evaluation Results for Different Cluster Numbers

Evaluation metrics	K= 2	K= 3	K=4	K=5	K=6	K=7
V-measure	0.435	0.578	0.66	1.0	0.678	0.532
Adjusted Rand Index	0.302	0.457	0.567	1.0	0.611	0.380
Adjusted-mutual information	0.435	0.578	0.66	1.0	0.67	0.531
Silhouette Coefficient	0.415	0.152	0.279	0.996	0.476	0.041

As we can see from Table 7, we can observe that the second model's performance is very low at cluster number 2, very high at cluster number 5, and start decreasing from cluster number 6. Adjusted Rand Index indicates perfect clustering would be scored 1 and bad clustering or independent clustering is scored 0 or negative. So, the second model's ARI value reaches 1.0 on cluster number 5 and this shows the second model perfectly clusters at K value five more than other cluster numbers. The value of the silhouette coefficient ranges from -1 to 1. One indicates that clusters are distinct and spaced widely apart. Zero indicates that clusters are unrelated, or that the distance between clusters is not significant, and -1 indicates that the clusters were incorrectly assigned. The silhouette coefficient reaches 0.996 at cluster number five, which indicates clusters are well apart from each other because of this number near number 1.

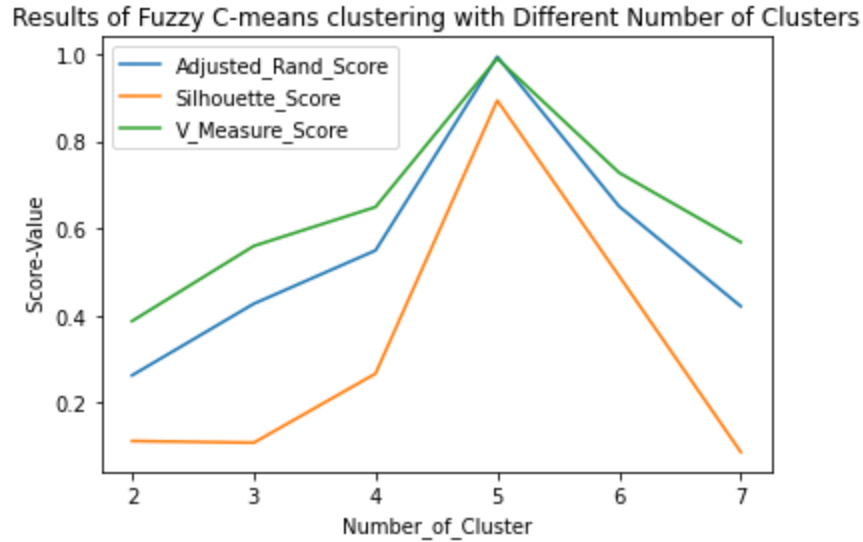


Figure 11 Results of Fuzzy C-Means Model with Different Number of Clusters.

Figure 11 shows the plot of the results of the second model with a different number of clusters. The experiment results show the perfect cluster number for the given dataset is 5 because the model has a high value of V-measure, Adjusted Rand Index, Adjusted-mutual information, and Silhouette Coefficient at k value 5 than the other cluster numbers. As we observed in figure 13 the clustering result starts increasing from cluster number up to cluster number 5, and the models have a high score value at cluster number 5. But starting from cluster number 6 it drops to 0.6 and 0.4 scores of Adjusted Rand Index and Silhouette Coefficient value. And, at cluster number 7 the evaluation score value drops highly. So, we can conclude that the optimal cluster number for the given dataset and Fuzzy C-Means Model is cluster number 5.

4.4. Comparison of the two models for Amharic Comment Clustering

As shown in the above section, the result of each model has been studied in terms of the following evaluation metrics:

- Homogeneity-score
- Completeness
- V-measure
- Adjusted Rand Index
- Adjusted Mutual Information

➤ Silhouette Coefficient

We compare the two models using the score value of those evaluation metrics and run-time complexity for clustering an equal number of datasets. Figure 14 shows comparison results of the BERT embedding with mini-batch K-means and BERT embedding with fuzzy C-means in terms of the above-listed evaluation metrics.

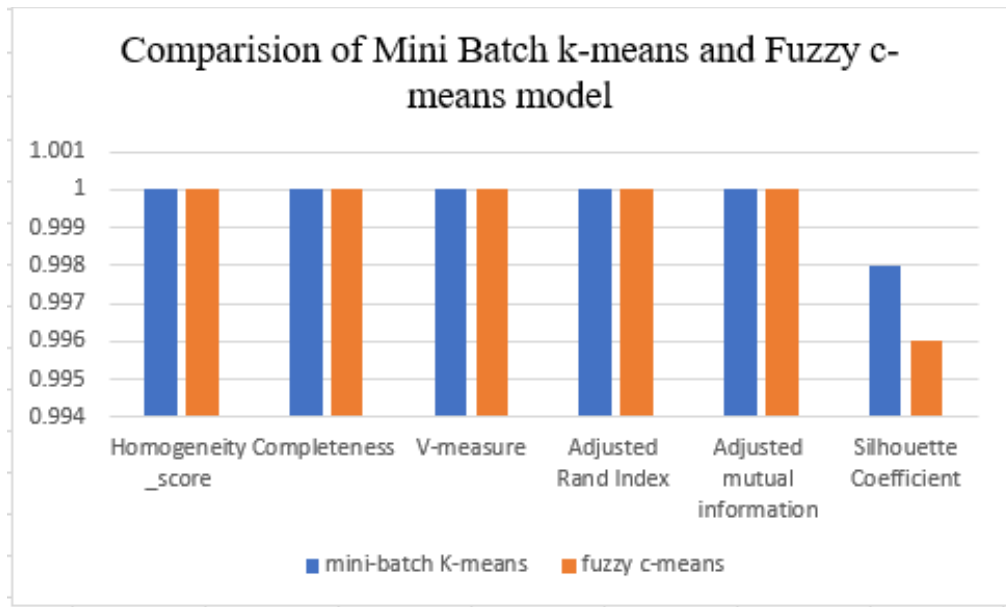


Figure 12 Comparison of the two models for Anharic Comment Clustering

The evaluation result shows that BERT embedding with mini-batch K-means algorithms shows better performance than BERT embedding with fuzzy C-means in our experiments. Literature shows that the Adjusted Rand Index is considered clustering accuracy. And the ARI value of the first model were 100% and 100% for the second model. So, the two models are equal in terms of accuracy but in terms of Silhouette Coefficient The first model has a high value of 0.998 and the second model have 0.996. The run-time complexity of the fuzzy c-means model is higher than the mini-batch K-means for clustering the given dataset. So, we can conclude that BERT embedding with mini-batch K-means algorithms is more accurate than the fuzzy c-means model.

4.5. DISCUSSION

For this research, we propose a Topic clustering model for Amharic comments on the news by experimenting with BERT embedding and two different clustering algorithms. We conducted experiments on the two models by selecting different hyperparameter setups to analyze the effect of the hyperparameter setups for selecting the best-performed model. We used the v-measure score, adjusted rand index, and Silhouette scores for measuring the performance of the model. The results of stopword elimination, normalization, and stemming have also been investigated. First, we conducted experiments without doing stopword elimination, normalization, and stemming separately and together. Then, we tried out stopword removal, normalization, and stemming. And we recognize that using the text preparation activities improves the effectiveness of the suggested work.

We have experimented on BERT embedding using a cross-lingual transfer learning approach to improve the clustering accuracy of short Amharic texts. For cross-lingual transfer learning, we use a combination of SBERT and MBERT and the model can accurately embed Amharic comments contextually. The experiment result showed that contextualized embeddings can improve clustering accuracy. We proved this by comparing the experiment results with works of literature that used non-contextual word embedding techniques. For instance, our work improves literature work (Assefa, 2020). This work archived an accuracy of 90% to cluster short Amharic texts using the word2vec embedding technique and k-means clustering algorithms. We achieved 100% accuracy to cluster Amharic comments using BERT embedding and mini-batch k-means. We understand that contextualized word embedding can improve the clustering accuracy of Amharic short texts more than non-contextualized word embedding because the nature of short texts is very concise.

We have done two experiments for the two models. The first used BERT embedding with mini-batch k-means and the second model was using BERT embedding with fuzzy c-means. The first model shows better performance than the second model in terms of time and Silhouette score value. As we understand from our experimentation the value of the number of clusters has a high impact on the experimentation results of mini-batch k-means. But the value of batch size and

maximum iteration number does not have any impact on clustering results. For fuzzy c-means, the value of fuzzifier, maximum iteration number, and the number of clusters have a high impact on clustering results. When the value of the fuzzifier equals one, the fuzzy c-means works like a hard-clustering method, and when its value gets too high all data points belong to all clusters. So, we understand that the fuzzifier value must be selected by considering the number of clusters and dataset size. And higher maximum iteration number improves the performance of fuzzy c-means.

The fuzzy c-means-based models take much more time than mini-batch k-means to cluster Amharic comments to news. This is because it performs more work than K means, fuzzy-C means will typically run more slowly. With each cluster, each point is reviewed, and each evaluation involves more processes. The mini-batch k-means have less computation time than k-means because it keeps data in memory in short, random, fixed-size batches, and then collects a random sample of the data to update the clusters with each iteration. It does not cycle over the complete dataset, it performs better than the usual K-means algorithm when working with large datasets.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In this thesis work, we have undertaken the clustering of Amharic comments to the news using the partitioning method specifically mini-batch K-Means and fuzzy c-means. For contextual sentence embedding, BERT was implemented. The main activity of this thesis work was to develop a model for Amharic comments clustering using BERT-based embedding and mini-batch K-Means and fuzzy C-means clustering. To accomplish the study activities like corpus preparation, preprocessing, design, implementation, and evaluation were done. The dataset preparation activity was performed crawl different local news websites and collecting different Amharic comments under different categories. After collecting the comments, the preprocessing module performs tokenization, stemming, normalization, stop word removal, and lemmatization on the corpus for better machine learning. For our experiment, we have identified an optimal number of cluster k to be five using different experiments. We have conducted the training experiment on a total of 7000 short Amharic texts on different topics.

We have evaluated the performance of two Amharic comments clustering models using the same evaluation metrics. i.e., V-measure, ARI, Adjusted Mutual Information, and Silhouette Coefficient. The evaluation result showed that BERT embedding with mini-batch K-means algorithms shows better performance than BERT embedding with fuzzy C-means in the experiments. Literature shows that the Adjusted Rand Index is considered clustering accuracy. And the ARI value for both models reaches 100% but the Silhouette Coefficient result of the first model was 99.8 % and 99.6 % for the second model. So, the first model is more accurate than the second model.

5.2 Contribution

In this work, we have developed a Topic clustering model for Amharic comments on the news using BERT-based embeddings and partitioning clustering algorithms. This study's key contribution is:

- We have prepared an Amharic comment dataset which can be an initial point for other researchers to study further. We have prepared 665KB Amharic comments and a 37.5KB Translation Dataset from English to Amharic.
- We have developed a model that can cluster short Amharic texts into a different group based on the topic. Due to grammar, semantic, and morphology difference, the models proposed for another language cannot be used in the Amharic language and no prior work was conducted on clustering short Amharic texts, so we can conclude that the proposed model is our contribution.
- This thesis work showed how effective are BERT models in contextually embedding Amharic comments.
- This thesis work showed that a better conceptual feature extraction technique can improve the clustering accuracy.

5.3 Recommendation

BERT is a model from Hugging face that improves the clustering of texts by considering the context for sentence embedding. This work tried to address the clustering of short Amharic texts with BERT embeddings. But another research effort is needed to make unsupervised text clustering more accurate.

- Extending this work by preparing enough datasets for Short Amharic text clustering. We have collected datasets on topics like health, politics, sport, entertainment, and technology news. As a result, we strongly recommend researchers include other additional topics.
- Assessing BERT-based embedding with other types of clustering algorithms like Hierarchical, density-based, affinity propagation, and others to enhance the accuracy of the model.

- We recommend researchers extend the transfer learning we have developed to the number of different Ethiopian languages by preparing a new language dataset.

References

- ANALYTICS, B. O. T. (2017). WHAT IS *TOPIC MODELING*? Provalisresearch.
<https://provalisresearch.com/blog/topic-modeling/>
- Anand, D. (2020). *Partitional Clustering*. Analytics Vidhya. <https://medium.com/analytics-vidhya/partitional-clustering-181d42049670#:~:text=What is Partitioning in Clustering%3F>
The most popular,clusters until a %28locally%29 optimal partition is attained.
- András. (2021). Subword pooling makes a difference. *EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, January*, 2284–2295. <https://doi.org/10.18653/v1/2021.eacl-main.194>
- Ankiit. (2020). *Word2vec vs BERT*. <https://medium.com/@ankiit/word2vec-vs-bert-d04ab3ade4c9#:~:text=Vectors%3A> Word2vec saves one vector representation of
a,classifier%2C which is a standard practice in application
- Assefa, K. (2020). *Short Amharic Text Clustering Using Topic Modeling* (Issue November) [Bahir Dar university]. <https://doi.org/10.13140/RG.2.2.17462.32326>
- Bao, T., Ren, N., Luo, R., Wang, B., Shen, G., & Guo, T. (2021). A BERT-Based Hybrid Short Text Classification Model Incorporating CNN and Attention-Based BiGRU. *Journal Of Organizational and End User Computing* 33(6), 1-21. <https://doi.org/10.4018/joeuc.294580>
- Berba, P. (2020). *Understanding HBDSCAN and Density-Based Clustering*.
<https://towardsdatascience.com/understanding-hdbscan-and-density-based-clustering-121dbee1320>
- Brownlee, J. (2019). *What Are Word Embeddings for Text?* Deep Learning for Natural Language Processing.
- Caiyan. (2018). Concept decompositions for short text clustering by identifying word communities. *Pattern Recognition*, 76, 691-703. <https://doi.org/10.1016/j.patcog.2017.09.045>
- Cannon, R. L., Dave, J. V., & Bezdek, J. C. (2012). Efficient Implementation of distinct the Fuzzy c-Means Clustering Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(2)*, 248–255. <https://doi.org/10.1109/TPAMI.1986.4767778>
- Chen, Z. (2019). *Short Text Embedding for Clustering based on Word and Topic Semantic Information*. 61–70. <https://doi.org/10.1109/DSSAA.2019.00020>

- Conneau, A, 2019. *XLM-RoBERTa (base-sized model)*. Hugging Face.
<https://huggingface.co/xlm-roberta-base>
- Dai. (2020). An unsupervised learning short text clustering method. *Journal of Physics: Conference Series*, 1650(3). <https://doi.org/10.1088/1742-6596/1650/3/032090>
- Devlin. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1(Mlm), 4171–4186.
- Donges, N. (2022). *What Is Transfer Learning? Exploring the Popular Deep Learning Approach*. <https://builtin.com/data-science/transfer-learning>
- Edureka. (2019). *Fuzzy K-Means Clustering*.
- Elton. (2022). *Cosine Similarity in Natural Language Processing*. Pythonwife.
- Ezra, K. (2022). *Word Embedding [Complete Guide]*. OpenGenus IQ.
- Geeksforgeeks. (2019). *V-Measure for Evaluating Clustering Performance*. Geeksforgeeks.
<https://www.geeksforgeeks.org/ml-v-measure-for-evaluating-clustering-performance/>
- Geeksforgeeks. (2020). *Partitioning Method (K-Mean) in Data Mining*.
<https://www.geeksforgeeks.org/partitioning-method-k-mean-in-data-mining/>
- Geeksforgeeks. (2021a). *Fundamentals of Software Architecture*. Geeksforgeeks
<https://www.geeksforgeeks.org/fundamentals-of-software-architecture>
- Geeksforgeeks. (2021b). *Mini Batch K-means clustering algorithm*. Geeksforgeeks.
<https://www.geeksforgeeks.org/ml-mini-batch-k-means-clustering-algorithm/>
- Getahun, S. (2021). *Customer Clustering for a Mobile Telecommunications Company Based on Call Detail Records*.
- Hadifar, A., Sterckx, L., Demeester, T., & Develder, C. (2019). A self-training approach for short text clustering. *ACL 2019 - 4th Workshop on Representation Learning for NLP, RepL4NLP, (2019), Proceedings of the Workshop, (2017)*. 194-199.
<https://doi.org/10.18653/v1/w19-4322>
- Hassan, A. (2020). Multi-source cross-lingual model transfer: Learning what to share. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, 3098–3112. <https://doi.org/10.18653/v1/p19-1299>
- Heidenreich, H. (2018). *Introduction to Word Embeddings*. Towards Data Science.

- Heu, J. U. (2018). Semantic-based K-means clustering for microblogs exploiting folksonomy. *Journal of Information processing System*, 14(6). 1438–1444.
<https://doi.org/10.3745/JIPS.04.0097>
- Interviewbit. (2022). *Top Applications of Python*.
<https://www.interviewbit.com/blog/applications-of-python>
- Jiaming. (2015). Short text clustering via convolutional neural networks. *1st Workshop on Vector Space Modeling for Natural Language Processing, VS 2015 at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2015*, 62–69. <https://doi.org/10.3115/v1/w15-1509>
- Kabaap. (2019). *Bag of words (BoW) model in NLP*. GeeksforGeeks.
- Karthikeyan. (2019). *Cross-Lingual Ability of Multilingual BERT: An Empirical Study*. 1–12.
<http://arxiv.org/abs/1912.07840>
- Kenton. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 4171–4186.
- KHARWAL, A. (2021). *Mini-batch K-means Clustering in Machine Learning*. Thecleverprogrammer. <https://thecleverprogrammer.com/2021/09/10/mini-batch-k-means-clustering-in-machine-learning/>
- Lasek, P. (2019). Density-based clustering with constraints. *Computer Science and Information Systems*, 16(2), 469–489. <https://doi.org/10.2298/CSIS180601007L>
- Lee, K. (2018). *BERT multilingual base model (uncased)*. Huggingface. BERT multilingual base model (uncased)
- Ma, E. (2018). *3 basic Distance Measurement in Text Minin*. Towardsdatascience.
- Malik, F. (2019). *Machine Learning Hard Vs Soft Clustering*. FinTechExplained.
- Mary, S. S., & Selvi, T. (2014). A Study of K-Means and Cure Clustering Algorithms. *International Journal of Engineering Research & Technology (IJERT)*, 3(2), 1985–1987.
- Maxion, R. (2009). *Experimental Methods for Computer Science Research*. 136–136.
<https://doi.org/10.1109/ladc.2009.29>
- McCormick, C. 2019. *BERT Fine-Tuning Tutorial with PyTorch*.
<https://mccormickml.com/2019/07/22/BERT-fine-tuning/>
- McGregor, M. (2020). *8 Clustering Algorithms in Machine Learning that All Data Scientists Should Know*. <https://www.freecodecamp.org/news/8-clustering-algorithms-in-machine->

learning-that-all-data-scientists-should-know/

Moberg, J. (2020). A deep dive into *multilingual NLP models*. Data-Science.

<https://peltarion.com/blog/data-science/a-deep-dive-into-multilingual-nlp-models>

Mohanty, A. (2019). *Understanding FastText: An Embedding To Look Forward To*. Medium.Com.

MULUGETA, A. G. (2021). . *july 2021 bahir dar, ethiopia* (Issue July). Bahir Dar university.

Nunzio. (2016). *Comments to News . Version : Accepted Version A Graph-based Approach to Topic Clustering for Online Comments to News*.

Ojha, V. (2020). *Understanding Euclidean Distance and Cosine_Similarity*. Analytics Vidhya.

Pascual, F. (2019). *Topic Modeling: An Introduction*.

Pedamkar, P. (2022). *Hierarchical Clustering Algorithm*. <https://www.educba.com/hierarchical-clustering-algorithm/>

Qiang, J., Qian, Z., Li, Y., Yuan, Y., & Wu, X. (2019). *Short Text Topic Modeling Techniques , Applications , and Performance : A Survey*. 14(8), 1–17.

Rakib. (2017). *Enhancement of Short Text Clustering by Iterative Classification*. 1–30.

Reimers, N. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 3982–3992. <https://doi.org/10.18653/v1/d19-1410>

Ruder. (2019). Unsupervised cross-lingual representation learning. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Tutorial Abstracts*, 31–38. <https://doi.org/10.18653/v1/p19-4007>

Ryan, C. M. and N. (2019). BERT word *Embeddings Tutorial*.

<https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/>

Saini, D. (2021). BERT *Embedding for Classification*. Analytics Vidhya.

<https://medium.com/analytics-vidhya/bert-embedding-for-classification-7c51aead26d9>

Schwämmle. (2016). A simple and fast method to determine the parameters for fuzzy c-means cluster analysis. *Bioinformatics*, 26(22), 2841–2848.

<https://doi.org/10.1093/bioinformatics/btq534>

Seid Muhie. (2019). *TETEYEQ : Amharic Question Answering For Factoid Questions*. 17–25.

Seldon. (2021). *Transfer Learning for Machine Learning*. <https://www.seldon.io/transfer-learning>

Shristikotaiah. (2020). *Word Embeddings in NLP*. GeeksforGeeks.

- Siddiqui, T., & Aalam, P. (2019). *Short Text Clustering ; Challenges & Solutions : A Literature Review*. June 2015.
- SIYAL, G. (2022). *10 Useful Tools for Python Developers*. [https://www.makeuseof.com/python-developer-tools/#:~:text= 10 Useful Tools for Python Developers ,spearheading Python ML and Deep Learning... More](https://www.makeuseof.com/python-developer-tools/#:~:text=10 Useful Tools for Python Developers ,spearheading Python ML and Deep Learning... More)
- Smalheiser. (2019). Unsupervised low-dimensional vector representations for words, phrases and text that are transparent, scalable, and produce similarity metrics that are not redundant with neural embeddings. *Journal of Biomedical Informatics*, 90(January), 103096. <https://doi.org/10.1016/j.jbi.2019.103096>
- Sourabh. (2022). *A tutorial on various clustering evaluation metrics*. DEVELOPERS CORNER. <https://analyticsindiamag.com/a-tutorial-on-various-clustering-evaluation-metrics/>
- Sukemi, J. P. E. (2019). Soft and Hard Clustering for Abstract Scientific Paper in Indonesian. *2019 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*.
- Suyal. (2016). Text Clustering Algorithms: A Review. *International Journal of Computer Applications*, 96(24), 36–40. <https://doi.org/10.5120/16946-7075>
- Swatimeena. (2020). *BERT Text Classification using Keras*. Data Scientist. <https://swatimeena989.medium.com/bert-text-classification-using-keras-903671e0207d>
- Systems, C. (2022). *Entropy, purity and V-measure*. WordPress and Bam. <https://complex-systems-ai.com/en/data-partitioning/entropy-purity-and-v-measure/>
- Team, G. L. (2020). *What is Hierarchical Clustering? An Introduction to Hierarchical Clustering*.
- Tutorialspoint. (2019). *Clustering Performance Evaluation*. Tutorialspoint. https://www.tutorialspoint.com/scikit_learn/scikit_learn_clustering_performance_evaluation.htm
- Waiyama, K. (2020). *Combining Distributed Word Representation and Document Distance for Short Text Document Clustering*. 16(2), 277–300.
- Wang. (2016). Clusters Merging Method for Short Texts Clustering. *Open Journal of Social Sciences*, 02(09), 186–192. <https://doi.org/10.4236/jss.2014.29032>
- Wibisono. (2021). Short text similarity measurement methods: a review. *Soft Computing*, 25(6), 4699–4723. <https://doi.org/10.1007/s00500-020-05479-2>
- Wu, S. (2022). *How Do Multilingual Encoders Learn Cross-lingual Representation?* <http://arxiv.org/abs/2207.05737>

- Yang. (2019). Discovering Topic Representative Terms for Short Text Clustering. *IEEE Access*, 7, 92037–92047. <https://doi.org/10.1109/ACCESS.2019.2927345>
- Yin, H., Song, X., Yang, S., Huang, G., & Li, J. (2021). Representation Learning for Short Text Clustering. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13081 LNCS, 321–335. https://doi.org/10.1007/978-3-030-91560-5_23
- Yin, J., & Wang, J. (2016). A Dirichlet multinomial mixture model-based approach for short text clustering. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 233–242. <https://doi.org/10.1145/2623330.2623715>
- Yufeng. (2021). *Fuzzy C-Means Clustering with Python*. Towards Data Science. <https://towardsdatascience.com/fuzzy-c-means-clustering-with-python-f4908c714081>
- ZACH. (2021). *What is the Rand Index? (Definition & Examples)*. ZACH. <https://www.statology.org/rand-index/>
- Zhou. (2019). Fuzzifier Selection in Fuzzy C-Means from Cluster Size Distribution Perspective. *Informatica*, 30(3), 613–628. <https://doi.org/10.15388/informatica.2019.221>

Appendix B

Some screenshots of preprocessing python codes

```
while index<=(num_dataset-1):#inside this loop tokenization, stop-word removal and stemming functions will
    sen=data[index][0]
    sen_list=sen.split()
    #print(sen_list)
    def remove_numbers():
        index_p=0
        size=len(sen_list)
        for word in sen_list:
            first_char=word[0]
            if first_char in amharic_numbers and index_p<=(size-1):
                temp=word[1:]
                sen_list[index_p]=temp
                #index_p=index_p+1
            elif index_p<=(size-1):
                sen_list[index_p]=word
                #index_p=index_p+1
            else:
                break
            index_p=index_p+1
        index_p=0
    for x in sen_list:# for removing punctuations from the end of each token.
        if len(x)>1 and x[-1] in amharic_numbers:
            temp=x[:-1]
            sen_list[index_p]=temp
            #index_p=index_p+1
        else:
            sen_list[index_p]=x
            #index_p=index_p+1
        index_p=index_p+1
    remove_numbers()
```

Tokenization

```
#print(sen_list2)
stopword=open("C:/Users/Mona Lisa/Desktop/resarch/New folder/stopword2.txt",encoding='utf-8-sig',mode='r')
stopwords=stopword.read()
stopword_list=stopwords.split()
sen_non_stopword=[]
for terms in sen_list2:
    if terms not in stopword_list:
        sen_non_stopword.append(terms)
```

Appendix C

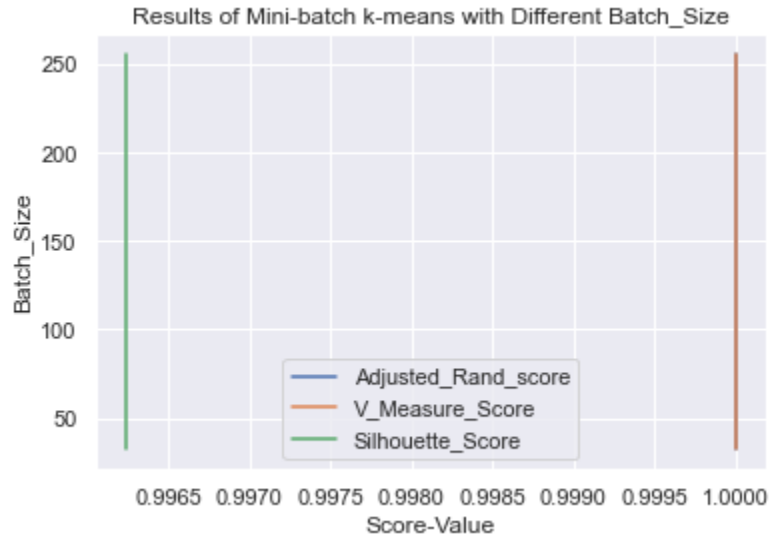
A sample snapshot of clustering result of BERT with mini-batch K-Means model

	Topic_com	Cluster
0	በጣም የሚገርመኝና የማይንቅለት አሁን ድረስ ለስፖርት ያለውን ጥንካሬና ...	4
51	ፋኖና አማራ ልዩ ኃይል ከውልቃይት ውጥቶ በመከላከያ ብቻ እንዲጠበቅ አድርገናል	4
52	ውንድ ከሆነ የትግራይን ምድር አይረግጣትም ታሪክ የሰራ መስሎት ታሪኩን ያ...	4
397	ለሪያል ማድሬድ ጩዋታ በፍጹም አይሆንም ተመልሶ ይመጣል	4
398	ባርሴሎና ሊቤ	4

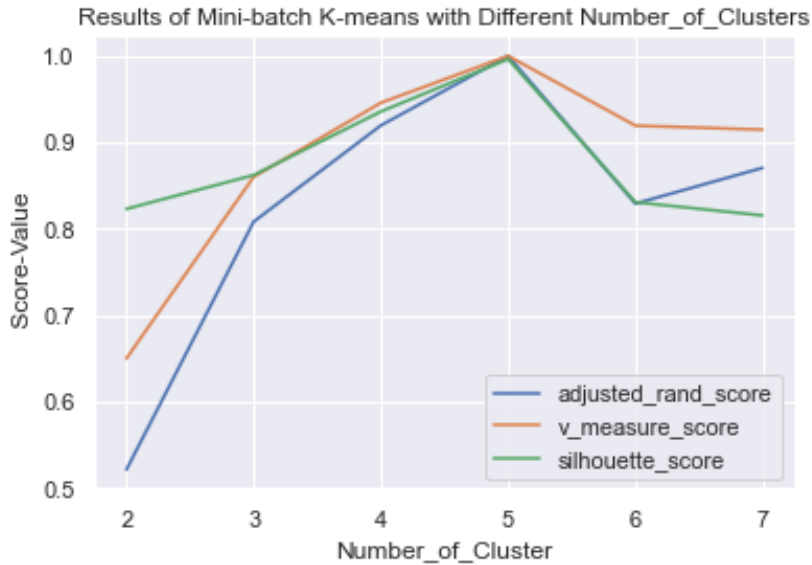
	Topic_com	Cluster
1	ኢትዮጵያ እየተደፈረች ያለችው በሠራዊቷ ችግር ሳይሆን ቁርጠኝነት በሌላቸው...	2
2	እደ አብይ አስተዳደር ኢትዮጵያ የተደፈረችበት ጊዜ የላትም አቤት ምላስ ...	2
3	የምትለው ሠራዊት እያለ እኮ ነው ሱዳኖች ያንን ያክል ኪ . ሜትሮች በውረ...	2
4	ቆሻሻች ከዚህ በፊት በህውሀት ስንደፈር ኢትዮጵያ መከላከያ ሰራዊት አልነበ...	2
5	ይህን የጅል ስብከት መስማት ሞኝነት እና ሰቆቃን ማራዘም ነው	2

Appendix D

Experimental Results of BERT with mini-batch K-Means model for different number of Batch size



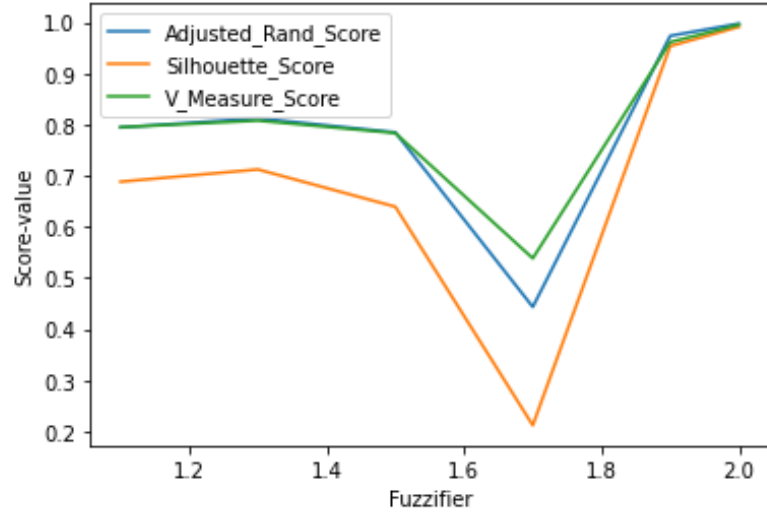
Experimental Results of BERT with mini-batch K-Means model for different numbers of Cluster



Appendix E

Experimental Results of BERT with fuzzy C-Means model with different Fuzzifier

Results of Fuzzy C-means Clustering with Different Number of Fuzzifier



Experimental Results of BERT with fuzzy C-Means model with different Maximum-iteration-number

Results of Fuzzy C-means Clustering with Different Number of Maximun_Itration

