NATURENGS
E-ISSN 2717-8013
https://dergipark.org.tr/en/pub/naturengs

Text Clustering with Pre-Trained Models: BERT, RoBERTa, ALBERT and MPNet

Oğuzhan ALAGÖZ [a], Taner UÇKAN [b,*]

[a] Van Yüzüncü Yıl University, Department of Artificial Intelligence and Robotics, Van Türkiye – 65080
[b] Van Yüzüncü Yıl University, Department of Computer Engineering, Van Türkiye - 65080

*Corresponding author

ARTICLE INFO

Received 01.11.2024
Accepted 06.12.2024

Doi: 10.46572/naturengs.1577517

ABSTRACT

Text clustering, a fundamental challenge in natural language processing, involves grouping semantically related sentences from texts of varying lengths into coherent and meaningful categories. This process plays a crucial role in data analysis by enabling the extraction of valuable insights from unstructured textual data. Despite extensive research employing diverse methodologies and computational techniques to address its inherent complexity, ongoing advancements have further refined the field. This study conducts a systematic comparison of pre-trained transformer-based models, including BERT (Bidirectional Encoder Representations from Transformers), RoBERTa (Robustly Optimized BERT Pretraining Approach), ALBERT (A Lite BERT), and MPNet (Masked and Permuted Pre-Training for Language Understanding), against the traditional statistical feature extraction method, TF-IDF (Term Frequency-Inverse Document Frequency). To evaluate the text representations generated by these methods, clustering algorithms such as Agglomerative Clustering, Mini-batch K-means, K-means, and BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies) were employed. The findings demonstrate that pre-trained transformer-based models consistently outperform the widely used TF-IDF technique in terms of clustering accuracy and efficiency. These results highlight the advanced capabilities of transformer-based language models, establishing them as powerful and practical alternatives to conventional approaches for enhancing text clustering performance.

Keywords: Natural language processing, sentence transformers, deep neural networks, pre-trained models, text clustering, text representation

1. Introduction

The field of information technology has undergone significant advancements, profoundly influencing numerous aspects of our lives. These developments have notably enhanced the speed and efficacy of communication, enabling individuals to establish connections with the other individuals across the globe through the use of social media, email and video conferencing platforms. Furthermore, it has facilitated access to information in the digital age. As a result of these, text and content data occupy a pivotal position within the digital realm, serving as the basis form of communication and information sharing.

Artificial intelligence has considerably improved textual processing and analysis, allowing for the extraction of meaning and insights from large amounts of unstructured data. Using natural language processing (NLP) technology, artificial intelligence can analyze and identify the relationships and signification of textual data, it can even produce text that is indistinguishable from human-written content. In this context, clustering is one of artificial intelligence's, and thus machine learning's, familiar problems. Clustering denotes the technique of partitioning a collection of data entries into distinct groups based on their shared characteristics. In text clustering, data points are text files and similarity is typically measured by the help of phrases and words used in the text groups or documents. The main goal of text clustering to categorize documents that exhibit content similarities into cohesive groups. In the context of machine learning, clustering is classified as an unsupervised learning technique that operates on unlabeled data (Bishop, 2006). Textual data available on the internet typically lacks labels, and the process of labeling all such data manually would be excessively challenging and time-consuming. So, a machine learning method is better suited for figuring out groups in text data (Subakti et al., 2022).

Across many disciplines, text clustering is used extensively to organize and analyze large collections of text data. Text clustering has been implemented to

* Corresponding author. e-mail address: taneruckan@yyu.edu.tr
ORCID : 0000-0001-5385-6775

enhance the capabilities of search engines, particularly in the context of data extraction. (Zhang et al., 2002). By contributing to more accurate and reliable outcomes in this field, it has been utilized to improve the efficiency of text classification processes in research associated with NLP. (Dhillon et al., 2001). In the field of bioinformatics, it has been used to analyze gene expression data and protein interaction networks (Boult et al., 2003). In social sciences area, it has been used to study the structure of online communities (Ahmed et al., 2007). Text clustering is also used in healthcare to predict maternal health risks by analyzing unstructured clinical data, improving early diagnosis and intervention tactics (Mutlu et al., 2023). In marketing and customer behavior researches, it is used to do sentiment analysis on consumer reviews and feedback, allowing businesses to better understand their customers and further develop their products and services (Liu 2012). Within the legal sector, it makes it easier to organize and retrieve enormous numbers of legal papers, case laws, and statutes, allowing legal practitioners to access relevant information more efficiently (Ashley, 2017). In cybersecurity area, it is employed to detect spam and phishing emails by grouping similar dangerous material, hence improving security measures against possible attacks (Caruana and Li, 2012).

In the domain of NLP, text clustering stands as a fundamental task, offering a broad spectrum of applications, covering but not limited to information retrieval, text classification, text organization, text summarization, and topic modeling. This versatility underscores its significance in advancing NLP methodologies and enhancing the analysis of textual data. (Aggarwal & Zhai, 2012). A variety of unsupervised algorithms are frequently employed for text clustering, including BIRCH, k-means clustering, mini-batch k-means clustering and agglomerative clustering algorithm. These techniques play a crucial role in systematically organizing and analyzing extensive datasets of textual information. Other unsupervised algorithms that have been used for text clustering include latent semantic analysis (LSA) (Deerwester et al., 1990) and latent Dirichlet allocation (LDA) (Blei et al., 2003). In this study, the first four techniques for text clustering and benchmarking will be used.

2. Materials and Methods

Because most machine learning algorithms, including clustering algorithms, can not work with raw text data, the text must be represented numerically prior to clustering. This step is called “text vectorization” or “text representation” and it is a crucial step in text processing. In shortest terms, text vectorization is the methodological approach of converting individual words, word fragments, clauses, or larger units such as sentences into vector representations. These representations are commonly known as vector embeddings, enabling effective computational analysis and manipulation of textual data. The implementation of text vectorization techniques has the potential to substantially augment the efficacy of machine learning models, thereby optimizing their ability to interpret and process data for improved predictive accuracy and informed decision-making (Le at al., 2014). Text vectorization not only enables the synthesis of textual data with diverse data modalities, like numerical datasets, but also significantly enhances the analytical framework for intricate tasks, including topic classification and sentiment analysis. This integration fosters a multidimensional approach to data analysis, thereby enriching the interpretative capacity and methodological rigor of empirical investigations. (Zhang et al., 2015). During the formative period of information retrieval, researchers sought to develop ways for expressing textual content in numerical form, thereby laying the foundational groundwork for contemporary data processing techniques. This initiative aimed at enhancing the functionality of search engines and various other applications reflects the early history of text vectorization methodologies.

Early methods such as term frequency-inverse document frequency (TF-IDF) emerged in the 1950s (Salton & Buckley, 1988). Conceived and refined during the 1970s and 1980s, the prominent TF-IDF method was built upon these foundational methodologies, which remain extensively utilized in contemporary text analysis. During the onset of the 2010s, word or text embedding techniques like GloVe (Global vectors for word representation) and word2vec (word to vector) became popular for text vectorization (Mikolov et al., 2013; Pennington et al., 2014). These techniques represented textual pieces as non-discrete vector expressions within a reduced-dimensional enviroment, which allowed them to uncover the conceptual relationships among data points, which denotes text pieces in text clustering. During the contemporary period models founded on transformer architecture, including generative pre-trained transformers (GPT) (Radford et al., 2018), bidirectional encoder representations from transformers (BERT) (Devlin et al., 2018), a lite BERT (ALBERT) (Lan et al., 2020), Robustly optimized BERT pre-training approach (RoBERTa) (Liu et al., 2019) and masked and permuted pre-training for language understanding (MPNet) (Song et al. 2020) have emerged as powerful text vectorization methods, achieving exceptional outcomes in a broad spectrum of NLP applications.

To conduct text clustering, this study employs pre-trained language models, namely BERT, ALBERT, RoBERTa, and MPNet, in conjunction with the traditional method of TF-IDF. As clustering methods BIRCH algorithm, k-means, agglomerative clustering and mini-batch k-means are used. The performances of pre-trained models are evaluated using clustering accuracy (ACC), normalized mutual information (NMI), fowlkes mallows score (FMS) and adjusted rand index (ARI).

2.1. Term Frequency-Inverse Document Frequency (TF-IDF)

The term frequency-inverse document frequency is a quantitative assessment employed to evaluate the significance of a text piece in a text file or set of

documents. This method is commonly utilized in tasks related to NLP and knowledge extraction like summary generation, content-based suggestive models or document classification (Sidorov, 2014). The basic idea behind TF-IDF is that text pieces which exhibit higher frequency in a particular document while being infrequent in others are deemed more significant and, consequently, should be assigned greater weight. This is due to the fact that such text pieces are greater chances of relevance to the document and provide useful information.

The determination of TF (term frequency) for a specific word is achieved by first calculating the word's occurrence frequency throughout the content and then normalizing this frequency against the total word count of that document. Computed by applying the logarithmic function, the IDF (inverse document frequency) quantifies the rarity of a term across the dataset by taking the ratio of the overall document count in the collection to the quantity of documents that include the specific word. The product of a word's term frequency (TF) and its inverse document frequency (IDF) points constitutes the overall term frequency-inverse document frequency (TF-IDF) computation for that word within a document. As seen in Formula-1, the mathematical formula used to calculate TF-IDF is illustrated below:

$$TF - IDF = (TF)(IDF) \quad (1)$$

The term frequency-inverse document frequency calculation can be generalized as seen in Formula-2:

$$w_{t,d} = t_{t,d} \log\left(\frac{n}{df_t}\right) \quad (2)$$

In this formulation, in document d the frequency of t words is represented by $tf_{t,d}$, the overall count of documents is represented by N, df_t shows documents' frequency including t number of words (Subakti et al., 2022).

2.2. Sentence Transformers

Sentence Transformers is a library developed by The Alan Turing Institute and DFKI (German Research Center for AI) and that provides pre-trained deep neural network models for creating sentence-level embeddings. The underlying architecture of these models is grounded in transformer networks, which have gained significant prominence in natural language processing owing to their capacity to effectively discern and model the contextual interdependencies among text pieces within a given sentence. (Vaswani et al., 2017). This library is designed to be easy to use and to allow fine-tuning on new tasks using transfer learning, a technique that allows the models to leverage their pre-learned knowledge on a large collection of linguistic documents to improve performance on specific NLP tasks.

Sentence embeddings are high-dimensional representations of sentences within a connected multidimensional space, which capture the contextual interpretation of sentences in a compact and transferable format (Conneau et al., 2017). The models in Sentence Transformers leverage a substantial amount of textual resources for training, like a public domain English-language reference platform (Wikipedia), to infer patterns how to produce superior sentence embeddings. Subsequently, specific layers tailored to particular natural language processing tasks can be incorporated atop the pre-trained embeddings, allowing for the fine-tuning of the models to address these specialized tasks effectively. (Howard & Ruder, 2018). This adaptation procedure enables models to suit to the particular requirements of a given task by the help of information acquired from preliminary model development process.

Sentence Transformers' effectiveness has been demonstrated across various NLP operations. For instance, in SemEval-STS benchmark with semantic text similarity task, Sentence Transformers models have accomplished top-tier performance (Wang et al., 2020). This library has also been used for response generation for queries, content labeling and entity identification tasks, demonstrating its versatility and effectiveness (Reimers & Gurevych, 2019).

Some of the most popular models included in Sentence Transformers are BERT, ALBERT, ROBERTA and MPNET.

BERT is a model grounded in transformer architecture that effectively seizes the context-specific associations among words of a sentence. A substantial corpus of textual data served as the basis for training BERT, which was subsequently refined for a diverse array of natural language processing operations, demonstrating strong performance and versatility (Devlin et al., 2018).

ALBERT represents a more efficient and streamlined variant of BERT that uses cross-layer parameter sharing and factorized embedding parameters to minimize the quantity of variables within the model. ALBERT outperforms BERT across diverse natural language processing applications while utilizing significantly fewer parameters (Lan et al., 2020).

ROBERTA improves on BERT by training with a more extensive dataset, employing dynamic masking, also changing the training objective. Consequently, enhancements in performance are observed across a broad spectrum of natural language processing tasks, including text categorization and entity recognition (Liu et al., 2019).

MPNET is a multi-prediction deep network that learns sentence representations by combining unidirectional and bidirectional attention mechanisms. MPNET has generated state-of-the-art results across diverse natural language processing applications, containing text classification and semantic text similarity (Song et al. 2020).

The models in Sentence Transformers, such as BERT, ROBERTA, MPNET, and ALBERT, are all cutting-edge models for creating sentence-level embeddings. These models have undergone pre-training using extensive collections of textual data and have been subsequently fine-tuned across a diverse array of natural language

processing tasks, demonstrating their flexibility and effectiveness.

2.3. BERT (Bidirectional Encoder Representations from Transformers)

Developed by researchers at Google in 2018, BERT represents an innovative self-supervised model for language representation. Trained on an extensive dataset of textual information, BERT acquires the ability to anticipate omitted words within a given sentence, which is referred to as "masked language modeling." This helps BERT to effectively capture the semantic environment of a text piece derived from the words preceding and following it, as well as the overall structure of the sentence.

Alongside the masked language modeling approach, BERT is additionally trained to ascertain the likelihood that one sentence succeeds another, a process referred to as NSP (next sentence prediction). That feature enables the model to comprehend the relationship among longer texts and capture the overarching significance of a given text. A significant advancement introduced by BERT is the incorporation of attention mechanisms, facilitating the model's ability to concentrate on particular segments of the text input when processing it. This allows BERT to effectively accommodate long-distance connections and capture the meaning of a word in a much larger context. BERT's embedding dimension is typically set to 768, which means every text piece given with the input text is used as a 768-element numerical representation. This particular dimension was chosen because it has been found to work well in practice and has produced cutting-edge outcomes with a variety of NLP operations. (Devlin et al., 2018).

BERT framework's in-depth analysis indicates that it constitutes a model architecture grounded in Transformer technology, heralding a novel phase in the execution of natural language processing operations. As seen in Figure 1, a transformer represents a specific kind of encoder-decoder architecture that also employs positional encoding, residual connection and different types of attention mechanisms. The BERT model exclusively employs the encoder component of the Transformer architecture, leaving out the Decoder. BERT, like the Transformer, employs positional encoding, self-attention, multi head attention, and Residual Connection. It employs the exact same encoder architecture as the Transformer.

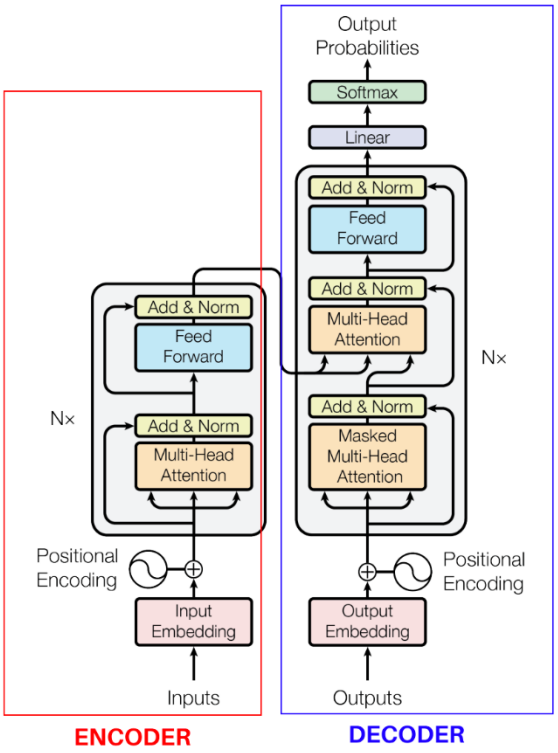

Figure 1. BERT using only the encoder part with transformer based model architecture (Fahim, 2021).

2.4. RoBERTa (Robustly Optimized BERT Pretraining Approach)

RoBERTa represents an adaptation of the widely recognized BERT language representation framework, which was developed in 2019 by researchers at Facebook AI (Liu et al., 2019). RoBERTa was created to address some of BERT's shortcomings and improve its efficacy across a diverse range of NLP operations.

The training data and process differ significantly between RoBERTa and BERT. RoBERTa undergoes training on a considerably more extensive and more varied dataset than that utilized for BERT, which includes both books and internet articles. RoBERTa is also trained with a more efficient training method that makes better use of the data, resulting in a larger and more accurate model than BERT.

In addition to having larger and more diverse training data, RoBERTa differs from BERT in data preprocessing and training procedure. The task of predicting the subsequent sentence has been eliminated from the training process of BERT, and the model handles the masked language modeling task differently. These changes make RoBERTa's training process more efficient and effective.

2.5. ALBERT (A lite BERT)

ALBERT represents a variation of the widely utilized BERT language representation framework created by Google researchers in 2020. (Lan et al., 2020). ALBERT was created to address some of BERT's efficiency limitations and improve its performance on resource-constrained devices. The model architecture is one significant difference between ALBERT and BERT. By initiating with cross-layer parameter sharing, ALBERT significantly cuts downs the parameter load by ensuring

that the same parameters are applied across different layers of the model. This allows ALBERT to be much smaller and more efficient than BERT while maintaining its performance.

"Sentence-order prediction" an innovatively presented self-supervised loss function in ALBERT, serves as a core feature that enhances the model's ability to capture relationships between sentences. Complementing this is a parameter-reduction technique, which improves computational efficiency by sharing parameters across layers. Together, these innovations not only optimize the model's performance in natural language understanding tasks but also reduce the computational cost without compromising on effectiveness.

2.6. MPNet (Masked and Permuted Pre-training for Language Understanding)

MPNet, an advanced large language model developed by Microsoft, is structured to enhance natural language comprehension tasks. (Song et al., 2020). Distinct from earlier transformer-based models, such as BERT and RoBERTa, MPNet integrates features of both permuted language modeling and masked language modeling approaches. This hybridization enables MPNet to exceed its predecessors in contextual understanding, combining the strengths of each modeling approach to achieve superior performance in linguistic representation and predictive accuracy. MPNet employs a permutation-based token ordering strategy during its training phase, which enables it to capture bidirectional context with greater efficacy than models restricted to masked language modeling alone. This approach significantly enhances the model's capacity to discern inter-word relationships, thereby producing more nuanced and robust contextual embeddings.

Regarding its performance metrics, MPNet demonstrates superior efficacy relative to numerous transformer-based models across widely recognized NLP benchmarks. This model thus represents a highly suitable option for applications demanding both rapid processing and high precision in textual analysis.

2.7. K-means Clustering Algorithm

The process of clustering using the K-means approach involves an algorithm of machine learning which operates in an unsupervised manner, effectively segmenting a dataset into various groups. (Jain, 2010). Partitioning samples from a dataset into distinct clusters is the primary objective of K-means clustering, ensuring that the similarity among points within the same cluster exceeds that of samples in other groups. The process initiates with the establishment of k centroids, representing the central positions of the clusters. Using a distance measure such as Euclidean distance, samples are then delegated to the group for which the centroid is nearest (Xu et al., 2015). Subsequently, the centroids are adjusted to reflect the mean of their respective clusters, after which the processes of assignment and update are reiterated.

The process continues until either the cluster assignments remain unchanged or a predetermined count of cycles is attained, involving the repeated adjustment of centroids and the reassignment of data points to their closest centroid. (Elkan, 2003). The process has now converged, and the final clusters and centroids are returned.

The K-means optimization function is represented as outlined in Formula-3:

$$W(S,C) = \sum_{k=1}^{K} \sum_{i \in S_k} |y_i - c_k|^2 \quad (3)$$

Where S is a K-cluster partition of the entity set represented by vectors y_i in the M-dimensional feature space, made up of non-empty non-overlapping clusters S_k, each with its own centroid c_k (k=1,2,...K).

Let S represent a partition of the entity set into K clusters, where each entity, denoted by the vector y_i resides in an multidimensional feature plane with M attributes. The clusters S_k (with k = 1, 2,…, K) are filled and mutually exclusive groups, each associated with its respective centroid c_k.

Steps in the k-means approach are as follows:

a. To begin, designate k points within the space that encapsulates the objects to be grouped. These points serve as the initial centroids of the clusters.

b. Reassign each entity to the cluster associated with the nearest centroid based on their proximity.

c. After all entities have been allocated to their respective groups, the positions of the k centroids are recalculated.

d. Continue iterating through Steps 2 and 3 until there is no further movement of the centroids.

(Kodinariya & Makwana, 2013)

2.8. BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)

The BIRCH approach represents a data grouping method designed to handle large data sets efficiently. It works by constructing a CF (Clustering Feature) tree, which is a tree-based data structure that stores data hierarchically. The algorithm begins by scanning the data set and producing a set of clusters based on a similarity metric. It then iteratively merges the groups up to a termination criterion is reached. BIRCH algorithm's ability to perform effectively with extensive data collections is one of its key features, making it an appealing choice for many applications. It has found widespread application in numerous contexts, containing study of gene expression patterns, data mining and image analysis.

In a recent publication within the Journal of Data Mining and Knowledge Discovery, it was revealed that the BIRCH algorithm outperforms other commonly employed clustering methods, such as K-means and DBSCAN, exhibiting enhanced quality of results and greater computational efficiency. (Zhang et al., 1996).

Another study published in the IEEE Transactions on Knowledge and Data Engineering discovered that when employed for the analysis of gene expression data, BIRCH "outperformed former leading clustering algorithms" (Guan et al., 2020).

In order to operate effectively, the BIRCH algorithm necessitates the specification of parameters such as the branching factor (Br), the threshold value (T), and the number of clusters (k). As values are processed through the BIRCH algorithm, a hierarchical structure known as the cluster features (CF) tree is incrementally constructed to represent the clusters. Calculations are made as follows, where C_i is the cluster center and R_i is the cluster radius as seen in Formula-4 and Formula-5:

$$C_i = \frac{1}{n_i} + \sum_j^n \ x_{ij}, where \ \{x_{ij}\}_{j=1}^n \ components \quad (4)$$

$$for \ i-th \ cluster$$

$$R_i = \sqrt{\frac{1}{n_i} + \sum_j^n \ (x_{ij} - C_i)^2}, for \ each \ cluster \quad (5)$$

2.9. Agglomerative Clustering (AC) Algorithm

Agglomerative clustering, a method within hierarchical or pyramidal grouping, is frequently employed for the clustering of textual data. Utilizing a gradual aggregation methodology, this approach initiates with individual data points represented as separate clusters, subsequently merging the nearest clusters in a series of iterations until the stopping criterion is met (Jain, 2010). This process is analogous to how a tree grows, with many small branches that eventually merge to form larger branches.

A significant advantage of agglomerative clustering is the method's capability to manage any amount size of data. This makes it a versatile choice for text clustering tasks involving large amounts of data with complex relationships. Furthermore, because agglomerative clustering is simple to implement and fully comprehend, it is a popular option among practitioners. The distance measure and linkage function used to merge clusters are critical decisions in agglomerative clustering. The distance measure specifies how related two data points are, while the linkage function determines how to calculate the distance between two clusters (Zhao et al., 2019). Distance measures for text data that are widely used include cosine similarity, Jaccard distance and Euclidean distance. (Zhao et al., 2019). Complete linkage, average linkage, and single linkage are all linkage functions (Jain, 2010). One restriction of agglomerative clustering is that it can be affected by the distance measure and linkage function used (Zhao et al., 2019). The improper pairing can result in poor cluster performance and inaccurate consequences. Practitioners must closely examine these options and verify their findings using various indicators.

2.10. Mini-batch K-means (MBKM) Clustering Algorithm

A variant of the conventional k-means technique, mini-batch k-means (MBKM) is frequently employed to partition a data collection into separate clusters, relying on the correlation among individual data points. The primary distinction between the two methods is that MBKM clusters the data in small subsets called "mini-batches," rather than the entire dataset at once (Sculley, 2010). This approach enables MBKM to effortlessly expand to large datasets and function properly with data streams.

One of MBKM's key benefits is its capability to process high-dimensional data. The computational cost of traditional k-means expands exponentially with the amount of dimensions, making it impractical for datasets with too many features (Vincent et al., 2008). Because of its mini-batch approach, MBKM could manage high-dimensional sets of data quite effectively. This allows it to be a useful mechanism for application domains such as data mining and image compression, where the data may include a variety of characteristics. Some other benefit of MBKM is its power to converge more quickly than conventional k-means, particularly for large sets of data with many samples. This is due to the mini-batch technique provides the method to refresh the cluster centers more frequently, resulting in faster convergence (Sculley, 2010). As a result, MBKM is suitable for implementations require fast processing of enormous sets of data.

Despite its advantages, MBKM has some drawbacks. Because of the random selection of mini-batches, it may yield to slightly different results than traditional k-means (Vincent et al., 2008). This can be reduced by raising the amount of mini-batches or by executing the algorithm several times and averaging the results.

2.11. Methodology Overview

As illustrated in below, the research process initiates with the aggregation of data and concludes examining the findings.

a. Data collection and data cleaning

b. Data representation with TF-IDF, BERT, RoBERTa, ALBERT or MPNet

c. Clustering with K-means, BIRCH, agglomerative clustering or mini-batch K-means

d. Clustering performance evaluations

e. Results and analysis

After data collection, text data preprocessed with lowercasing, elimination of numerical values, punctuation removal, trimming of white spaces, exclusion of stopwords and lemmatization steps. Preprocessed text data represented with traditional TF-IDF method, and with pre-trained models BERT, RoBERTa, ALBERT and MPNet. Then, four different text clustering methods, K-means, BIRCH, agglomerative clustering and mini-batch k-means applied. After clustering simulation, results are evaluated using clustering accuracy, normalized mutual information,

fowlkes mallows score and adjusted rand index clustering performance metrics.

2.12. Text Data

In this research, popular datasets AG news and emotions are used. AG news dataset contains exceeding one million articles about news which was collected from approximately 2.000 resources using an academic news search engine named ComeToMyHead. This data collection process took more than 1 year (Gulli, 2015). As seen in Table 1, 20.000 articles for AG news and 16.000 tweets are used for text clustering in this research.

Table 1. Short description of AG news and emotion datasets.

Dataset	Number of classes	Number of data
AG news	4	20000
Emotion	6	16000

2.13. Text Representations

TF-IDF vectorization is conducted with scikit-learn library's TfidfVectorizer using maximum 158503 features. This vectorizer uses L2 normalization by default, meaning that cosine similarity between two vectors is calculated by their dot product. On the other hand for BERT, RoBERTa, ALBERT and MPNet, different pre-trained models are used from Sentence-Transformers library as seen in Table 2 and Table 3.

Table 2. Pre-trained models from Sentence-Transformers with arithmetic mean average performance of sentence embeddings and semantic search embeddings.

Pre-trained Sentence Transformer Models	Average performance
multi-qa-distilbert-dot-v1	59.59
all-roberta-large-v1	61.64
paraphrase-albert-small-v2	52.25
all-mpnet-base-v2	63.30

Table 3. Pre-trained models from Sentence-Transformers with speed and model size.

Pre-trained Sentence Transformer Models	Speed (encoding speed, sentence/sec, on a V100 GPU)	Model size (MB)
multi-qa-distilbert-dot-v1	4000	250
all-roberta-large-v1	800	1360
paraphrase-albert-small-v2	5000	43
all-mpnet-base-v2	2800	420

3. Results And Discussions

3.1. AG News Dataset

The clustering results benchmarking metrics used in the study are applied using the ground-truth labels in the dataset. As seen in Table 4, K-means clustering method with MPNet pre-trained model give the best performance with %82 accuracy using AG news dataset. However, in total evaluation, ROBERTA pre-trained model outperforms the rest pre-trained models and TF-IDF text representation.

Table 4. Text clustering performance evaluation using AG news dataset.

Clustering and text representation method	ARI	NMI	FMS	ACC
Kmeans+BERT	0.37	0.41	0.53	0.70
Kmeans+ROBERTA	0.57	0.54	0.67	0.80
Kmeans+ALBERT	0.46	0.45	0.60	0.74
Kmeans+MPNET	0.60	0.56	0.70	0.82
BIRCH+BERT	0.47	0.46	0.60	0.75
BIRCH +ROBERTA	0.55	0.52	0.66	0.80
BIRCH +ALBERT	0.37	0.41	0.56	0.68
BIRCH +MPNET	0.53	0.53	0.65	0.78
AGC+BERT	0.46	0.46	0.60	0.75
AGC +ROBERTA	0.53	0.53	0.65	0.78
AGC +ALBERT	0.38	0.39	0.54	0.70
AGC +MPNET	0.50	0.51	0.63	0.77
MB-Kmeans+BERT	0.37	0.41	0.53	0.69
MB-Kmeans +ROBERTA	0.57	0.54	0.67	0.80
MB-Kmeans +ALBERT	0.26	0.31	0.45	0.50
MB-Kmeans+MPNET	0.37	0.42	0.53	0.59
Kmeans+TF-IDF	0.05	0.23	0.42	0.45
Birch +TF-IDF	0.10	0.24	0.43	0.50
AGC +TF-IDF	0.03	0.15	0.43	0.39
MB-Kmeans +TF-IDF	0.24	0.32	0.46	0.62

In order to provide a better evaluation, the clustering distributions and confusion matrices of the methods which performed minimum %75 accuracy or higher in Table 3 are presented in Figure 3. As predicted labels and ground-truth labels does not match naturally, confusion matrices are relabelled after evaluated intuitively. Class distribution for Kmeans+ROBERTA , Kmeans+MPNET , BIRCH+ROBERTA, BIRCH+MPNET, AGC+BERT, AGC+ROBERTA , AGC+MPNET and MB-Kmeans+ROBERTA are given in figure 2-9 respectively.

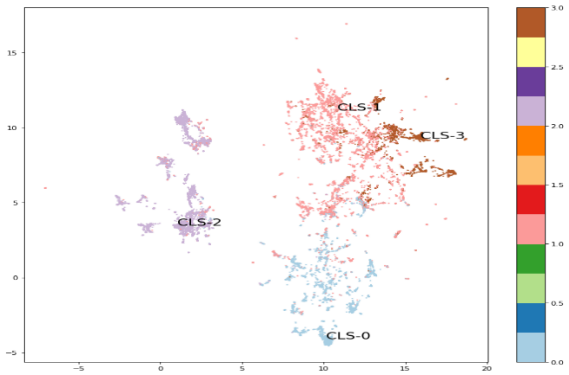

Figure 2. Class distribution for Kmeans+ROBERTA methodology

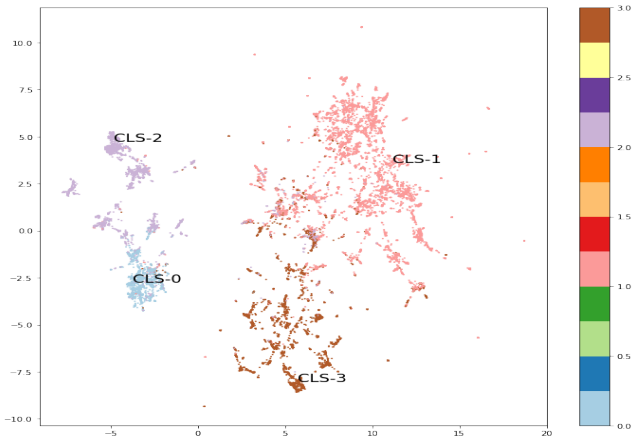

Figure 3. Class distribution for Kmeans+MPNET methodology

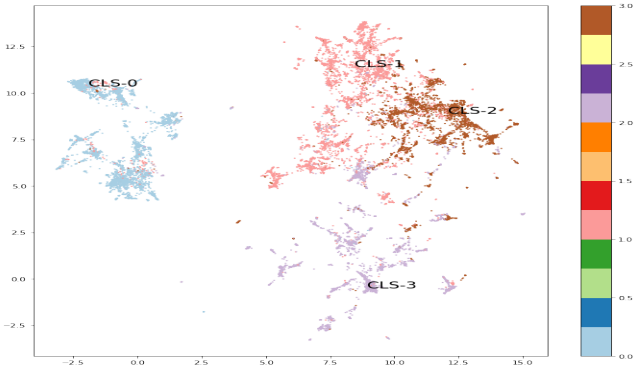

Figure 4. Class distribution for BIRCH+ROBERTA methodology

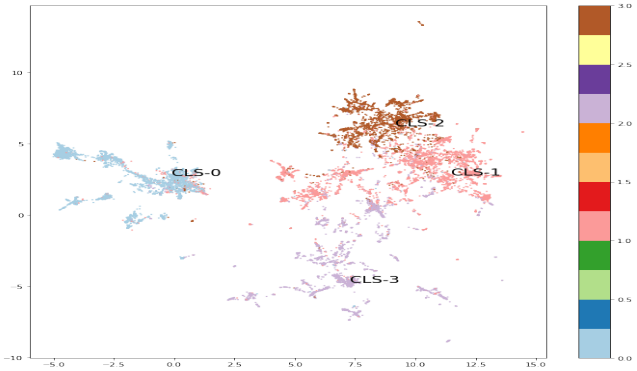

Figure 5. Class distribution for BIRCH+MPNET methodology.

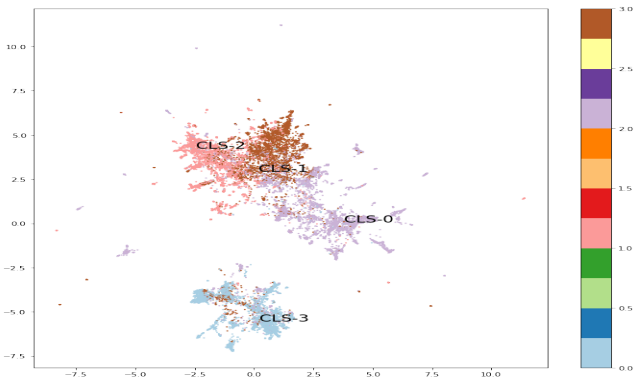

Figure 6. Class distribution for AGC+BERT methodology

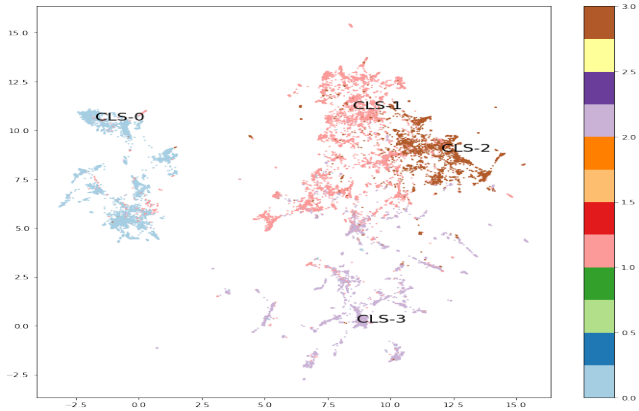

Figure 7. Class distribution for AGC+ROBERTA methodology

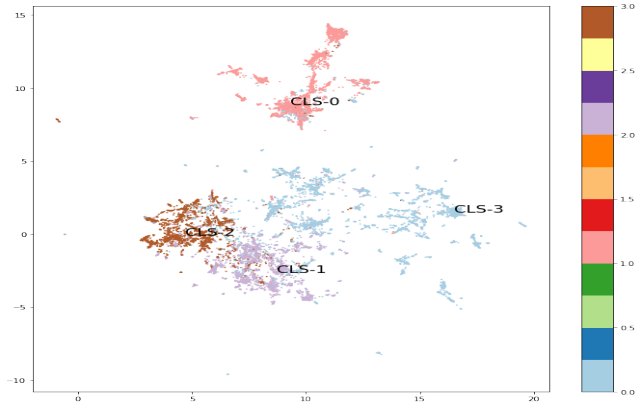

Figure 8. Class distribution for AGC+MPNET methodology

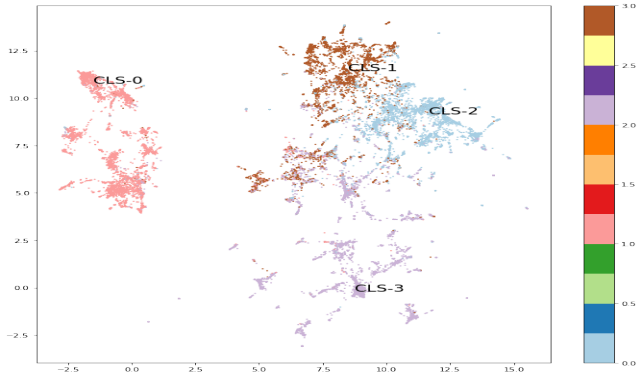

Figure 9. Class distribution for MB-Kmeans+ROBERTA methodology

3.2. Emotion Dataset

As seen in Table 5, BIRCH clustering method with MPNet pre-trained model give the best performance with %58 accuracy using emotion dataset. The primary factor contributing to the low success rate with the Emotion dataset is the brevity of the texts it contains. Since the texts are very short, all methods were insufficient to calculate the data representation capability of the words.

Table 5. Text clustering performance evaluation using AG news dataset.

Clustering and text representation method	ARI	NMI	FMS	ACC
Kmeans+BERT	0.03	0.18	0.45	0.40
Kmeans+ROBERTA	0.05	0.22	0.47	0.43
Kmeans+ALBERT	0.04	0.25	0.53	0.47

Kmeans+MPNET	0.10	0.28	0.45	0.57
BIRCH+BERT	0.05	0.20	0.47	0.43
BIRCH +ROBERTA	0.03	0.18	0.45	0.41
BIRCH +ALBERT	0.05	0.19	0.44	0.48
BIRCH +MPNET	0.11	0.30	0.46	0.58
AGC+BERT	0.03	0.17	0.46	0.41
AGC +ROBERTA	0.04	0.24	0.52	0.47
AGC +ALBERT	0.03	0.19	0.48	0.45
AGC +MPNET	0.05	0.21	0.46	0.49
MB-Kmeans+BERT	0.07	0.07	0.48	0.51
MB-Kmeans +ROBERTA	0.10	0.28	0.54	0.53
MB-Kmeans +ALBERT	0.04	0.17	0.45	0.41
MB-Kmeans+MPNET	0.04	0.18	0.47	0.42
Kmeans+TF-IDF	0.02	0.16	0.35	0.22
Birch +TF-IDF	0.02	0.23	0.31	0.19
AGC +TF-IDF	0.03	0.24	0.43	0.26
MB-Kmeans +TF-IDF	0.02	0.19	0.29	0.21

4. Conclusions

All of the pre-trained BERT, RoBERTa, ALBERT and MPNet models used in the study have a higher success rate than TF-IDF in all clustering methods. While the TF-IDF method provided the highest clustering performance with the Mini-batch K-means clustering method with an accuracy of 62%, the MPNet pre-trained model reached 82% accuracy with the K-means clustering algorithm.

Compared to the study of Subakti et al. in 2022 and the study of Guan et al. in 2020 which use the same AG news dataset, superior success for the first one and approximately the same success for the second one were achieved for in the ACC, ARI and NMI metrics, as can be seen in Table 6.

Table 6. Comparison of performance metrics with Subakti et al.'s study in 2022 and Guan et al.'s study in 2020 (I stands for identity normalization, MM stands for min-max normalization, LN stands for layer normalization and N stands for normalization).

Clustering and text representation method	ARI	NMI	ACC
This study			
Kmeans + ROBERTA	0.57	0.54	0.80
Kmeans + MPNET	**0.60**	**0.56**	**0.82**
BIRCH + ROBERTA	0.55	0.52	0.80
BIRCH + MPNET	0.53	0.53	0.78
AGC+ BERT	0.46	0.46	0.75
AGC + ROBERTA	0.53	0.53	0.78
AGC + MPNET	0.50	0.51	0.77
MB-Kmeans + ROBERTA	0.57	0.54	0.80
Subakti et al.'s study			
Kmeans + BERT + Max + I	**0.48**	**0.48**	**0.76**
Kmeans + BERT + Max + LN	**0.51**	**0.51**	**0.79**
Kmeans + BERT + Max + N	**0.51**	**0.51**	**0.78**
Kmeans + BERT + Max + MM	0.19	0.19	0.44
Kmeans + BERT + Mean + I	0.41	0.41	0.64
Guan et al.'s study			
Kmeans + BERT + Max + I	0.48	0.22	0.22
Kmeans + BERT + Max + LN	**0.82**	**0.56**	**0.56**
Kmeans + BERT + Max + N	**0.82**	**0.56**	**0.56**
Kmeans + BERT + Mean + LN	**0.80**	**0.53**	**0.53**
Kmeans + BERT + Mean + I	**0.80**	**0.54**	**0.54**
Kmeans + BERT + Mean + N	**0.80**	**0.53**	**0.53**
Kmeans + BERT + Last + LN	0.34	0.05	0.05
Kmeans + BERT + Last + I	0.35	0.05	0.05
Kmeans + BERT + Last + N	0.35	0.05	0.05

Our findings imply a paradigm shift in text clustering approaches by constructing pre-trained transformer models as a resilient and practical alternative to traditional methods. The use of these models can result in more intelligent data analysis and accurate information extraction from unstructured textual data. Future study should investigate the integration of these models with other clustering techniques, as well as their application across multiple domains and languages, in order to evaluate their effectiveness.

References

[1] **Ahmed, A.,** Boyce, E., & Pfeffer, J. (2007). The structure of online discussion groups: A case study. Management Science (pp. 1432-1445).

[2] **Aggarwal CC** & Zhai C. (2012). A survey of text clustering algorithms in mining text data (pp. 77–128). New York, London: Springer.

[3] **Ashley, K. D.** (2017). Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age. Cambridge University Press. https://doi.org/10.1017/9781316761380

[4] **Blei, D. M.,** Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. Journal of Machine Learning Research (pp. 993-1022).

[5] **Bishop** C.M. (2006). Pattern Recognition and Machine Learning (pp. 128-129). Newyork, USA:Springer.

[6] **Boult, T.,** DeRose, T., Czerwinski, M., & Smith, B. (2003). A comparison of clustering algorithms for gene expression data. Pacific Symposium on Biocomputing (pp. 535-546).

[7] **Caruana, G.,** & Li, M. (2012). A survey of emerging approaches to spam filtering. ACM Computing Surveys, 44(2), 1–27. https://doi.org/10.1145/2089125.2089129

[8] **Conneau, A.,** Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. URL: https://ar5iv.labs.arxiv.org/html/1705.02364 (accessed date: March 23, 2023).

[9] **Deerwester, S.,** Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science (pp. 391-407).

[10] **Devlin, J.,** Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. URL: https://arxiv.org/abs/1810.04805 (accessed date: May 12, 2023).

[11] **Dhillon, I.,** Mallela, S., & Modha, D. (2001). Concept decompositions for large sparse text data using clustering. Machine Learning (pp. 143-175).

[12] **Elkan, C.** (2003). Using the triangle inequality to accelerate k-means. In Proceedings of the 20th international conference on Machine learning (ICML-03) (pp. 147-153).

[13] **Fahim, M.** (2021). BERT - In depth understanding. URL: https://www.kaggle.com/code/mdfahimreshm/bert-in-depth-understanding (accessed date: January 17, 2023).

[14] **Guan R,** Zhang H, Liang Y, Giunchiglia F, Huang L, Feng X. (2020). Deep feature-based text clustering and its explanation. IEEE Transactions on Knowledge and Data Engineering. URL: https://ieeexplore.ieee.org/document/9215004 (accessed date: November 10, 2023).

[15] **Gulli, A.** (2015). AG's corpus of news articles URL: http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html (accessed date: December 30, 2022).

[16] **Howard, J.,** & Ruder, S. (2018). Universal language model fine-tuning for text classification. URL: https://arxiv.org/abs/1801.06146 (accessed date: April 08, 2023).

[17] **Jain, A. K.** (2010). Data clustering: 50 years beyond K-means. Pattern recognition letters, 31(8), 651-666.

[18] **Kodinariya,** T. M., & Makwana, P. R. (2013). Review on determining number of Cluster in K-Means Clustering. International Journal, 1(6), 90-95.

[19] **Lan, Z.,** Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. URL: https://arxiv.org/abs/1909.11942 (accessed date: December 22, 2022).

[20] **Le, Q.,** & Mikolov, T. (2014). Distributed representations of sentences and documents. In International conference on machine learning (pp. 1188-1196).

[21] **Liu, Y.,** Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach.URL: https://arxiv.org/abs/1907.11692 (accessed date: January 17, 2023).

[22] **Liu, B.** (2012). Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers. https://doi.org/10.2200/S00416ED1V01Y201204HLT016

[23] **Mikolov, T.,** Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[24] **Mutlu, H. B.,** Durmaz, F., Yücel, N., Cengil, E., & Yıldırım, M. (2023). Prediction of maternal health risk with traditional machine learning methods. Naturengs, 4(1), 16-23.

[25] **Pennington, J.,** Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[26] **Radford, A.,** Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2018). Language models are unsupervised multitask learners. OpenAI. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed date: September 17, 2022).

[27] **Reimers, N.,** & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. URL: https://arxiv.org/abs/1908.10084 (accessed date: February 03, 2023).

[28] **Salton, G.,** & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information processing & management (pp. 513-523).

[29] **Sidorov, G.** (2014). Term frequency–inverse document frequency (TF-IDF) as a tool of content-based recommendation systems. International Journal of Computer Science and Information Security, 12(1), 44-51.

[30] **Song, K.,** Tan, X., Qin, T., Lu, J., & Liu, T. Y. (2020). Mpnet: Masked and permuted pre-training for language understanding.URL: https://arxiv.org/pdf/2004.09297 (accessed date: June 18, 2023).

[31] **Subakti, A.,** Murfi, H. & Hariadi, N. (2022). The performance of BERT as data representation of text clustering. URL: https://doi.org/10.1186/s40537-022-00564-9 (accessed date: March 25, 2023).

[32] **Vaswani, A.,** Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (accessed date: December 17, 2022).

[33] **Wang, A. L.,** Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2020). GLUE benchmark: Evaluating large-scale language understanding systems. URL: https://arxiv.org/abs/1804.07461 (accessed date: October 02, 2022).

[34] **Xu, R.,** Wunsch, D., & Hu, J. (2015). Survey of clustering algorithms. IEEE Transactions on Neural Networks and Learning Systems, 26(11), 2264-2281.

[35] **Zhang, T.,** Ramakrishnan, R., and Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. In Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp. 103-114, 293-304..

[36] **Zhang, Y.,** Zha, H., & Lai, J. (2002). Text clustering based on the latent Dirichlet allocation model. Advances in Neural Information Processing Systems (pp. 1049-1056).

[37] **Zhang, Y.,** Wallace, B. C., & Li, M. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. URL: https://arxiv.org/abs/1510.03820 (accessed date: February 19, 2023).

[38] **Zhao, Y.,** Chen, W., & Liu, X. (2019). A survey of clustering algorithms for text data. ACM Computing Surveys, 52(5), 1-45.