Ingénierie des Systèmes d'Information
Vol. 29, No. 1, February, 2024, pp. 235-246
Journal homepage: http://iieta.org/journals/isi

# Augmenting Document Classification Accuracy Through the Integration of Deep Contextual Embeddings

Rama Krishna Paladugu[1,2*], Gangadhara Rao Kancherla[1]

[1] Department of Computer Science and Engineering, Acharya Nagarjuna University, Guntur 522510, India
[2] Department of Computer Science and Engineering, R.V.R. & J.C. College of Engineering, Guntur 522019, India

Corresponding Author Email: mails4prk@gmail.com



https://doi.org/10.18280/isi.290123

**Received:** 22 May 2023
**Revised:** 25 August 2023
**Accepted:** 6 October 2023
**Available online:** 27 February 2024

***Keywords:***
*Deep Contextual Embedding Models, text clustering algorithms, document classification, natuaral language processing and machine learning*

**ABSTRACT**

Document classification, a fundamental process within the field of natural language processing, has benefitted from the recent advancements in deep learning, particularly in enhancing accuracy. Traditional text clustering methods, such as bag-of-words models, exhibit domain specificity and struggle to handle vast data volumes. They also face limitations in elucidating sophisticated patterns and intricate word and phrase relationships within textual data. These constraints may adversely affect the accuracy of text clustering, subsequently impacting downstream applications like information retrieval, document classification, and natural language processing. This paper proposes a novel text classification model, termed Deep Contextual Embeddings Model (DCEM), designed to improve document classification accuracy. The DCEM integrates pre-trained deep contextual embedding architectures (e.g., GPT-2) with text clustering algorithms (e.g., K-Means). It employs contextual embedding models to enhance document clustering accuracy by capturing context and semantic depth, improving data structure comprehension, and eliminating noise for more precise results. Experimental results, derived from the application of DCEM on AG News, Reuters-21578, and IMDB reviews datasets, indicate a significant improvement in document classification accuracy (81.09%), compared to traditional text clustering and document classification methods.

## 1. INTRODUCTION

The field of Natural Language Processing (NLP) has garnered substantial attention in recent research, largely due to the advent of deep learning techniques. A central task within NLP is document classification, defined as the process of assigning categories to documents based on their respective contents or characteristics. Traditional text clustering methods, such as bag-of-words models, have historically been prevalent in achieving this task [1, 2]. However, these conventional approaches face limitations. They often struggle to detect sophisticated patterns and interdependencies within textual data, and typically exhibit domain specificity and an inability to handle large data volumes [3].

The constraints of traditional text clustering models become evident in real-world scenarios, such as the processing of customer feedback. Consider a company inundated with diverse customer feedback in the form of product reviews, customer service inquiries, and social media posts. Here, traditional text clustering methods, including bag-of-words models [4, 5], may fail to accurately group similar feedback due to inherent deficiencies in capturing context and meaning.

For instance, when a customer review encapsulates both positive and negative aspects of a product, a bag-of-words model might categorize the feedback based on word frequency rather than capturing the holistic sentiment or meaning. This could potentially lead to misclassification, causing the company to overlook valuable insights. Recent advancements in deep learning have shown potential in surmounting these challenges by utilizing Deep Contextual Embedding Models (DCEMs) [6-9] for text representation. Through the integration of DCEMs, companies can more accurately discern the context and meaning of customer feedback, leading to improved categorization and, ultimately, superior understanding of customer sentiment and needs. These models generate embeddings that capture the contextual information of each word and phrase in a document, thereby enabling clustering algorithms to group similar documents with higher accuracy.

In recent years, deep learning methodologies have demonstrated considerable potential in enhancing the accuracy of natural language processing tasks, including text clustering and document classification [10-12]. Notably, the pre-training of deep bidirectional transformers [6, 8], as epitomized by the BERT model proposed by Devlin et al. [6], has made significant strides in language understanding. Further, Convolutional Neural Networks (CNNs) have been extensively utilized for text categorization, capitalizing on word order to enhance performance [3]. Attention-based models, such as the Transformer model proposed by Vaswani et al. [13], have also demonstrated their efficacy in capturing intricate patterns and relationships within textual data. Despite

these advancements, traditional text clustering methods, such as bag-of-words models, retain their prevalence across various domains, although their limitations in capturing contextual information and managing extensive data volumes are well documented [11].

The primary objective of this research is to devise an innovative deep contextual embedding model to address the constraints of existing contextual models and enhance the accuracy of deep text clustering and document classification. The proposed methodology integrates pre-trained contextual models with clustering algorithms, aimed at augmenting the quality of clusters and document classification accuracy. High-level semantic features are extracted from text documents using pre-trained models and subsequently employed as input for clustering algorithms, thus grouping similar documents. This integration improves the capacity to capture the context and semantics of words and phrases, in addition to unravelling the underlying structure and relationships within the text data. Furthermore, it aids in the interpretation and understanding of latent representations in clustering, simultaneously identifying and eliminating noise and outliers, thereby leading to enhanced clustering performance. The novelty of this research paper lies in its integration of pre-trained Deep Contextual Embedding Models (such as GPT-2) with traditional clustering algorithms (e.g., K-Means, DBSCAN), thereby improving text clustering and document classification accuracy and effectively addressing the limitations of conventional methods.

In order to substantiate the proposed methodology, a series of experiments were undertaken utilizing several publicly accessible datasets, including AG News, Reuters-21578, and IMDB reviews. The datasets were selected based on several criteria such as relevance, accessibility, volume, complexity, cleanliness, and their representative nature of real-world data.

The effectiveness of text clustering algorithms and document classification accuracy was assessed using a variety of performance metrics. These included the Silhouette score, Calinski-Harabasz index, Davies-Bouldin index, Homogeneity, Completeness, Normalized mutual information, Accuracy, Precision, Recall, and F1-Score. This assessment was conducted on several labeled datasets with ground truth labels to evaluate the performance of the clustering algorithm. Such experimental evaluation proved pivotal in determining the optimal combination of contextual embedding models for text clustering and document classification accuracy.

The proposed methodology has potential implications across several domains, including data analysis, information extraction, and target recommendation systems. By integrating Deep Contextual Embedding Models (DCEMs) with text clustering algorithms, it is anticipated that the efficiency and accuracy of downstream NLP tasks can be significantly enhanced. In summary, this research represents a significant progression in the field of document classification, highlighting the potential utility of employing DCEMs for a range of NLP applications.

## 2. LITERATURE SURVEY

This section undertakes a comprehensive survey of the existing literature, with a specific focus on identifying the limitations of traditional text clustering methods and discussing the advancements in Deep Contextual Embedding Models.

### 2.1 Traditional methods

An extensive survey of text clustering algorithms, inclusive of traditional and deep learning-based methodologies, is provided by Liang et al. [11]. Their study underscored the limitations of conventional methods, such as the inability to extract textual semantics and their suboptimal accuracy in classification tasks.

Meanwhile, Wei et al. [14] propose a novel model for text document clustering, leveraging term frequency-inverse document frequency (TF-IDF) weighting and cosine similarity [15]. While this approach was observed to surpass basic clustering methods, it failed to account for the semantic meaning of words.

A traditional text clustering algorithm integrating evidence accumulation clustering and concept generation is proposed by Wong et al. [16]. Despite its innovative approach, it was noted that the method required a large number of parameters and was heavily reliant on a fixed set of predefined concepts.

In a distinct approach, Mohammed et al. [17] propose a text document clustering methodology utilizing the firefly algorithm. Although it was found to outperform general clustering algorithms, the model necessitated the tuning of several parameters.

Karol et al. [18] presented a text clustering method that amalgamates fuzzy clustering with particle swarm optimization (PSO) [19]. Despite surpassing traditional clustering methods in terms of accuracy, stability, and robustness, this method was found to be computationally demanding and required the tuning of multiple parameters.

### 2.2 Deep contextual embedding methods

The literature review underscores a critical gap, namely, the disregard for semantic meaning of words [2-5] in several of the methods reviewed. Recent scholarly endeavors have concentrated on deep learning-based strategies [11, 12] and alternative techniques such as topic modeling and graph-based clustering [20]. Deep Contextual Embedding Models [6-9] have emerged as a promising avenue for text clustering, delivering state-of-the-art results.

An enhanced text clustering algorithm utilizing Deep Contextual Embedding Models with attention mechanisms is introduced by Wei et al. [21]. Notably, this approach contributes to the improvement of cluster quality and stability. Subsequently, Zhang et al. [22] presented a novel text clustering technique rooted in Deep Contextual Embedding Models, employing a self-attention mechanism to enhance clustering accuracy.

A fresh deep contextual embedding model for text clustering is put forth by Tan et al. [23]. This model, leveraging a sparse auto-encoder and a skip-gram model, displays improved clustering accuracy compared to conventional methodologies. Mehta et al. [24] proposed a unique text clustering approach using Deep Contextual Embedding Models and word mover's distance (WMD), demonstrating superior clustering accuracy than traditional clustering algorithms. By incorporating contextual information and a feature selection algorithm, Ravi and Kulkarni [25] succeeded in enhancing text clustering performance using Deep Contextual Embedding Models. They utilized a regularization technique to tackle the challenge of high dimensionality inherent in text data.

In 2019, a novel language model titled Conditional

Transformer Language Model (CTRL) was introduced by Keskar et al. [7]. CTRL's distinct ability to generate text based on user-defined attributes, offering nuanced control over style, tone, and content, renders it a suitable model for various NLP tasks like text classification, summarization, and query-response. This stands in contrast to previous language models that were limited to text generation. In the same year, Devlin et al. [6] presented BERT, a pre-trained deep bidirectional transformer model. The study exhibited BERT's superior performance across multiple NLP tasks, establishing it as a state-of-the-art model.

ROBERTa, introduced by Liu et al. [8] in 2019, is trained on a substantial volume of input data using an approach similar to BERT. However, the use of larger batch sizes and more training data than BERT resulted in a higher-performing model. ELECTRA, a pre-training method for NLP developed by Google researchers [9], employs a generator-discriminator approach to pre-training text encoders, with the generator being a masked language model trained to predict masked tokens, similar to BERT [6].

This literature review underscores advancements in deep learning-based techniques for NLP tasks [6-9], such as text clustering and document classification. Although these models have advanced NLP tasks, they still exhibit limitations in capturing contextual information, handling dimensionality, domain dependency, and interpretability. The identification of these research gaps motivates the present study to improve the prominence of semantic meaning in text clustering algorithms and suggests potential avenues for future research.

## 3. PROPOSED METHOD

The novelty of our integrated approach lies in combining pre-trained Deep Contextual Embedding Models (e.g., GPT-2) with clustering algorithms (e.g., K-Means, DBSCAN) to effectively captures contextual information, improves clustering performance, and addresses the limitations of traditional methods, resulting in more accurate and context-aware document analysis.

Traditional text clustering methods, such as model based text clustering [26] and feature based text clustering [27], which struggle to accurately categorize and group similar feedback due to their limitations in extracting the context and meaning of words and phrases. To address the limitations of conventional text clustering methods in the realm of deep text clustering, pre-trained deep contextual models, such as BERT [6], GPT-2 [10], and RoBERTa [8] were introduced and they have shown remarkable success in deep text clustering. These models learn contextualized representations of words, which capture the meaning and context of words in a sentence or document. However, the standalone contextual models (i.e. BERT, GPT-2 and RoBERTa etc.) are having certain limitations when it comes to deep text clustering. Some of these limitations include: Lack of clustering and interpretability abilities, Ambiguity in capturing document similarities, Limited domain knowledge and ability to handle noise and outliers.

In general, the standalone contextual models are designed for NLP tasks like text generation, classification, and sentiment analysis but not for deep text clustering on their own. Moreover the contextual model may produce latent representations in clustering, which are difficult to interpret and understand [28, 29]. As contextual models are trained on large amounts of generic data and they may not have specific domain knowledge relevant to the text being clustered, which leads to the suboptimal clustering results. The contextual models are optimized for generating coherent and fluent sentences and paragraphs in language modeling [30]. Hence they are unable to effectively handle the noise and outliers (irrelevant data points) in text data clustering. Due to the lack of a direct mechanism for modeling the document-level similarity [31], the contextual models may not be able to effectively capture the similarities between documents.

To overcome the limitations of standalone contextual models, this paper proposes an integrated deep contextual embedding model, in which the pre-trained contextual models are integrated with clustering algorithms, to increase the cluster quality and document classification accuracy. In this method, pre-trained models [32] are employed to extract high-level semantic features from text documentsand these extracted features are then used as input to the clustering algorithms [33] to group similar documents. Integration of clustering algorithms with pre-training contextual models will enhance the ability to capture the context and meaning of words and phrases along with the underlying structure and relationships in the text data. This integration will also help to interpret and understand the latent representations in clustering along with identifying and removing noise and outliers, leading to better clustering performance. The proposed model provides a more robust and generalizable approach to text clustering, reduces the need for manual feature engineering, and offers superior performance compared to traditional and standalone contextual text clustering models.

### 3.1 Deep contextual text clustering model

In the context of deep text clustering and document classification, the proposed model (Figure 1) for integrating pre-trained contextual models [6-8] with clustering algorithms is designed in several steps are: Dataset selection, Data pre-processing, word embedding's, Positional encoding, Semantic Features Extraction, Feature vector dimensionality reduction, Text clustering and Document classification.

#### 3.1.1 Data selection

As part of the "Integration of pre-trained contextual models with clustering algorithms" project for deep text clustering and document classification, we selected the popular text datasets for experimentation and evaluation.

Each dataset ' $D$ ' comprises multiple documents $\{d_1, d_2, d_3 \dots d_n\}$ spanning different categories $\{c_1, c_2, c_3 \dots c_n\}$, making them widely used benchmarks in NLP research for text classification and clustering tasks. These datasets are particularly well-suited for evaluating the performance of text clustering and classification algorithms due to their extensive size and diverse range of topics.

#### 3.1.2 Pre-processing

After dataset selection process, the next step involves in preprocessing the text data with respective techniques [34], which is crucial to increase the cluster quality and document classification accuracy in further steps. As part of this, the unwanted characters or symbols from the text data (i.e. punctuation marks, emojis, special characters, and HTML tags etc.) are removed using the regular expressions. To make simplify the clustering process, the input text data $d_i$ is

converted into the input token (i.e., individual words or phrases) sequences. Frequently occurring stop words (i.e. “the”, “and”, “is”, etc.) are eliminated and the unique words are stemmed from the input token sequences to reduce the processing complexity.

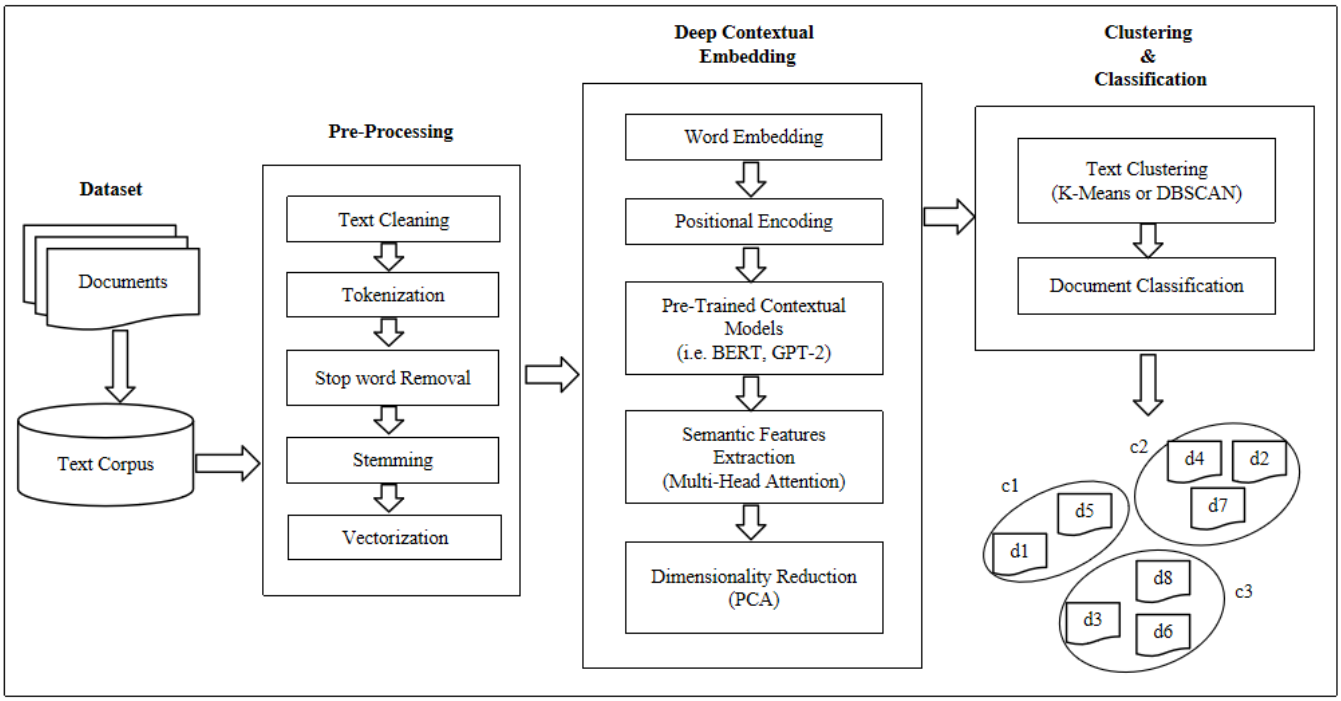


**Figure 1.** Block diagram of the Deep Contextual Embedding Models based text clustering

3.1.3 Word embedding

After pre-processing, different techniques are used to convert the text (tokens) into numerical representations for clustering. In case of our pre-trained contextual models, the input token sequences are converted into the “word embedding vectors” using a transformer-based neural network [35]. word embedding vectors is to represent words and phrases in a continuous, semantically meaningful space. These word embeddings are essential for capturing contextual information, understanding the relationships between words, and enhancing the accuracy of text clustering and document classification tasks. They enable the integration of pre-trained deep contextual models with clustering algorithms, providing a foundation for improved document analysis.

The word embedding’s [36] generated by the contextual models is used to extract high-level semantic features for text clustering and document classification.

Let's take a sequence of words, denoted as $W = \{w_1, w_2, w_3, \ldots w_n\}$, where ‘$n$’ represents the length of the sequence. Each word in the sequence is represented by a $d$ - dimensional vector, denoted as, where $d$ represents the dimension of the embedding space. The embedding for the $i^{th}$ word in the sequence is denoted as $e_i = \{e_{i1}, e_{i2}, e_{i3}, \ldots e_{id}\}$.

3.1.4 Positional encoding

In our approach, we incorporate the order of words in the input sequence by implementing positional encoding [37]. By employing this technique, we are able to encode details regarding the position of each token in the input text, which allows our model to capture the sequential information and contextual meaning of the words. The positional encoding is a vector that is added to the input embedding’s before they are fed into the multi-head attention mechanism. The formula for the positional encoding is:

$$PE_{i,2j} = \sin\left(\frac{i}{10000^{\left(\frac{2j}{d_{\text{model}}}\right)}}\right) \tag{1}$$

$$PE_{i,2j+1} = \cos\left(\frac{i}{10000^{\left(\frac{2j}{d_{\text{model}}}\right)}}\right) \tag{2}$$

where, $'i'$ is the position of the word in the input sequence, $'j'$ is the index of the dimension in the embedding vector, and $d_{\text{model}}$ is the dimension of the embedding vector. This equation is to incorporate the positional encoding $PE_{(i,j)}$ to the generated embedding’s for $i^{th}$ word and $j^{th}$ dimension is defined as:

$$PE_{(i,j)} = \begin{cases} \sin\left(\frac{i}{10000^{2j/d}}\right), & \text{if 'j' is even} \\ \cos\left(\frac{i}{10000^{2j/d}}\right), & \text{if 'j' is odd} \end{cases} \tag{3}$$

The positional encoding values are now added to the original embedding vector for each word to create the final embedding vectors, which has to be processed by the subsequent layers of the proposed model.

At this moment, the contextual model has to generate a hidden state for each token in the input text, which captures the contextualized meaning [35] of the token based on its context within the sentence. For each token of the position encoded final embedded vector, our pre-trained contextual

model generates a sequence of hidden states, $H = \{h_1, h_2, h_3 \ldots h_n\}$, where each $h_i$ is a $d$ -dimensional vector representing the hidden state at position'$i$'. The hidden states are generated by iteratively applying multi-head self-attention and feed-forward layers, with each layer utilizing the output of the previous layer to produce a more refined representation of the input text.

3.1.5 Semantic features extraction

When applying pre-trained contextual models [6, 9] in the context of deep text clustering with clustering algorithms [26], multi-head self-attention is utilized to capture the high-level semantic features of each word in the input text. The multi-head attention mechanism [38] is considered as a type of the self-attention mechanism where instead of using a single set of learned matrices to generate queries, keys, and values, multiple sets of these matrices are used. The attention computation is performed multiple times in parallel, each time using a different set of matrices. The ultimate output is achieved by concatenating the results of each attention head and then passing them through a linear transformation.

The pre-trained contextual model generates word embedding's along with positional encodings, which are then fed as input to the multi-head self-attention layer.The word embedding's $(E)$ in self-attention are fed through learnable weight matrices $W_Q$, $W_K$ *and* $W_V$ of the linear transformation to create the queries $(Q)$, keys $(K)$, and values $(V)$ vectors as:

$$Q = W_Q * E, \quad K = W_K * E \ \text{ and } V = W_V * E \tag{4}$$

These vectors are then split into multiple heads $(head_1, head_2, \ldots head_n)$, and each head computes the attention score between the queries and keys to obtain the attention weights. The output of each attention head is obtained by computing a weighted sum of the values using the attention weights. The hidden states produced by the multi-head self-attention mechanism in our model can be represented as:

$$M\_Head(Q,K,V) = Concat(head_1, \ldots, head_n)W^O \tag{5}$$

where, $head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$, and $W_i^Q \in \Re^{d_{\text{model}} * d_k}$, $W_i^K \in \Re^{d_{\text{model}} * d_k}$, $W_i^V \in \Re^{d_{model}{}^{*d_v}}$, $W^O \in \Re^{hd^* d_{model}}$.

In this equation, $Q, K, V \in \Re^{n_v{}^* d_{model}}$ represents the input query, key, and value matrices, where $d_k$ denotes the dimensions of the key and value vectors for each head, '$n$' represents the count of attention heads, and $d_{\text{model}}$ represents the dimensionality of the input embedding's. Multi-head attention model is to capture complex semantic features and relationships within the text data. Multi-head attention allows the model to focus on different parts of the input text simultaneously, enabling it to extract diverse and informative contextual information. This is crucial for improving the performance of text clustering and document classification, as it helps in understanding the nuances, nuances, and subtle patterns in textual data, ultimately leading to more accurate results. In Multi-Head attention model the attention for each head is evaluated as:

$$\text{Attention}(Q,K,V) = soft\_max\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

In this context, the dimensionality of the key vector is denoted as $d_k$, and a scaling factor $\sqrt{d_k}$ is applied to prevent the dot product from becoming too large, which can impact the soft-max function. The soft-max function is then applied to the dot product of the query $Q$ and key vectors $K^T$ along the second axis to obtain the attention weights. The resulting attention weights are then multiplied by the value matrix $V$ to obtain the final output.

3.1.6 Feature vector dimensionality reduction

The output of the contextual model may result in high-dimensional feature vectors. To enhance computational efficiency and reduce the dimensionality of these vectors, dimensionality reduction techniques like Principal Component Analysis (PCA) [39] or t-Distributed Stochastic Neighbor Embedding (t-SNE) [40] can be employed. The choice of dimensionality reduction technique should depend on the data characteristics and analysis objectives. In this study, we have opted for the linear technique, and PCA is used for implementing the dimensionality reduction process.

PCA is a widely used linear method for reducing dimensionality. It converts high-dimensional data into a lower-dimensional space while preserving the maximum possible variance in the data. This dimensionality reduction technique ensures that the clustering algorithms can operate more efficiently and effectively. The reduced feature vectors obtained through PCA can be used for further analysis such as clustering or classification. Let '$X$' be the original high-dimensional feature vectors of shape ($n$_samples, $n$_features). The initial process in PCA is to center the data '$X_C$' by subtracting the mean from each feature:

$$X_C = (X - \bar{X}) \tag{7}$$

After centering the data, the covariance matrix $S$ can be calculated with number of samples $n$ and mean of the centereddata $X_C$ as:

$$S = \frac{1}{n-1}\left(X_C\right)^T * \left(X_C\right) \tag{8}$$

The eigenvectors $(v)$ and eigenvalues $(\lambda)$ of the covariance matrix can be computed as:

$$Sv = \lambda v \tag{9}$$

The top $k$ eigenvectors can be selected based on the highest $k$ eigenvalues. The data can then be projected onto the new basis formed by these top $k$ eigenvectors as shown in below:

$$X_R = XW \tag{10}$$

where, $X$ is the original data, $W$ is the matrix of top $k$ eigenvectors, and $X_R$ is the transformed data with reduced dimensionality.

3.1.7 Text clustering

Following dimensionality reduction, clustering algorithms such as K-means, Hierarchical Clustering, or DBSCAN can be applied to group similar documents based on the extracted high-level features [33]. K-means clustering [41] is a widely used unsupervised clustering algorithm that partitions data points into K clusters based on their similarity. In the context

of our contextual-based text clustering, the feature vectors obtained from the pre-trained contextual models after PCA reduction are utilized as input data points for clustering. The k-means algorithm can be defined as follows:

(1) *Initialization*: Randomly select K cluster centroids from the data points.

(2) *Assignment*: Allocate the data points with the closest centroid.

(3) *Update*: Update all centroids by computing the data points mean of them.

(4) *Repeat*: Continue with steps 2-3 until convergence is achieved.

The Elbow Method [41] involves running K-Means for a range of K values and calculating the sum of squared distances (inertia) of data points to their assigned centroids for each K. A plot of K against inertia is created, and the "elbow point" where the inertia starts to level off is considered as the optimal K. Silhouette analysis measures how similar each data point in one cluster is to the data points in the same cluster compared to the nearest neighboring cluster. For each K value, a silhouette score is calculated, and the K that results in the highest silhouette score is chosen as the optimal number of clusters. In the case of k-means clustering, the distance between two data points is frequently defined using the Euclidean distance metric [41], which can be expressed as:

$$d\left(x_i, x_j\right) = \sqrt{\sum_{k=1}^{n}\left(x_{i,k} - x_{j,k}\right)^2} \tag{11}$$

In above context the $x_i$and $x_j$ are two data points, $n$ is the dimensionality of the feature vectors ($X_R$), $x_{i,k}$ and $x_{j,k}$ are the corresponding '$k$' components of the obtained feature vectors for $x_i$ and $x_j$ , respectively. The objective of k-means clustering is to minimize the total sum of squared distances between data points and the centroids to which they are assigned, as expressed by the following equation:

$$\sum_{i=1}^{N}\sum_{j=1}^{K} w_{ij} d\left(x_i, c_j\right)^2 \tag{12}$$

where,'$N$' is considered asthe count of data points, $K$ is clusters count value, $c_j$ is the centroid of the $j^{th}$ cluster, and $w_{ij} = 1$ , if data point $x_i$ is assigned to cluster $j$ , or 0 otherwise.The k-means algorithm converges when the cluster assignments no longer change, i.e., the centroids no longer move. The ideal number of clusters, denoted as K, can be determined through techniques such as the elbow method or silhouette analysis.

3.1.8 Document classification

After the k-means clustering [41] is performed on the lower-dimensional feature vectors, the resulting clusters can be treated as different classes or categories for document classification. At this moment, we perform the document classification process on each cluster by assigning a label to the entire cluster. The best way to do this is by computing the centroid [42] of each cluster and using it as a representative vector for that cluster. Then, for each document in the cluster, we can compute its similarity to the centroid vector using a similarity metric such as cosine similarity. The similarity score can be used to assign the document to the appropriate class. The equation for computing the centroid vector for a cluster $C$ is:

$$\text{centroid}(C) = \frac{1}{|c|}\sum_{i \in C} X_i \tag{13}$$

where, $C$ is considered as the documents count in the cluster, and $X_i$ represents the feature vector of document $i$. Once the centroid vector is computed, the comparison score amongst the document $X_i$ and its centroid vector $\boldsymbol{c}_C$ can be calculated using cosine similarity:

$$\text{similarity}\left(x_i, \mathbf{c}_\mathbf{C}\right) = \frac{x_i \cdot \mathbf{c}_\mathbf{C}}{\left|x_i\right|\left|\mathbf{c}_\mathbf{C}\right|} \tag{14}$$

where, '.' represents the dot product amongst the two vectors, and '|.|' represents the Euclidean model of a vector. Based on the obtained similarity scores, we can assign each document in the cluster to a class with the highest score. In this way the total documents are classified based on the contained cluster similarity scores.

## 4. EXPERIMENTAL ANALYSIS

Experimental evaluation plays a vital role in Deep Contextual Embedding Models-based text clustering to assess algorithm performance in accurately clustering text data. The primary aim of the experimental evaluation is to identify the optimal combination of contextual embedding models for text clustering and document classification using the selected datasets. The evaluation process is typically carried out on various labeled datasets where the ground truth labels are utilized to measure the clustering algorithm's performance. This section provides a step-wise description of the experimental evaluation process.

### 4.1 Dataset selection

In this paper, the datasets used for text clustering are generally collections of documents, which can be categorized into various domains and topics. These datasets are utilized as a benchmark to evaluate and compare the performance of the proposed deep contextual models with their counterparts in terms of clustering accuracy, efficiency, and reliability. Several publicly available datasets are used in this text clustering research are the AG News dataset [43], Reuters-21578 [44] dataset and the IMDB reviews [45] dataset. While selecting these datasets we have considered the research problem relevance, accessibility, volume and complexity. Additionally, the quality of the data, including its cleanliness and representativeness of the real-world data, should be taken into account when selecting a dataset.

### 4.2 Clustering and classification metrics

Evaluation metrics are crucial in measuring the effectiveness of the clustering algorithm in accurately grouping the text data and improving the overall classification accuracy. In our experiments, multiple metrics are employed to assess the performance of text clustering algorithms, including: Silhouette score [46], Calinski-Harabasz index [47], Davies-Bouldin index [48], Homogeneity [49], Completeness [50] and Normalized mutual information [51].

**Silhouette score** (SIL) [46] is a widely used metric for evaluating the quality of clustering algorithms. A higher

Silhouette score indicates better clustering performance. The Silhouette score '$Sil(m)$' is calculated as variance among the mean proximity to the closest cluster and the mean proximity to all other clusters divided by the maximum of these two values. The formula to evaluate the '$Sil(m)$' is:

$$\text{Sil}(m) = \frac{bw(m) - \text{avg}(m)}{\max(\text{avg}(m), bw(m))} \tag{15}$$

In above equation, '$m$' represents a sample, denotes the average distance $avg(m)$ between '$m$' and the remaining points of the cluster, and represents the average distance between '$m$' and the remaining points in the next closest cluster. $Sil(m)$ can be vary from -1 to 1, with a score of -1 denoting erroneous clustering, a score of 0 denoting overlapping clusters, and a score of 1 denoting dense, well-separated clusters. Better clustering performance is thought to be indicated by a score that is higher and closer to 1.

**Calinski-Harabasz Index** (CHI) [47] or Variance Ratio Criterion, as a measure of clustering quality, is used in our evaluation. For all clusters, it measures the proportion of between-cluster variation to within-cluster variance, with a higher value indicating improved clustering. The mathematical definition is as follows:

$$CHI(n) = \frac{T(bw_n)}{T(in_n)} X \frac{m-n}{n-1} \tag{16}$$

where, '$m$' is the number of data points, '$n$' is the total number of clusters, $T(bw_n)$ is the trace off between-cluster scatter matrix, and $T(\text{in}_n)$ is the trace off within-cluster scatter matrix. The '$\text{CHI}(n)$' has a range of 0 to ∞, with a bigger value indicating more successful clustering.

**Davies-Bouldin Index (DBI)** [48] is a popular tool for assessing the effectiveness of clustering. It calculates the average degree of similarity between each cluster and its closest neighbor, taking the cluster's size into account. It is defined as:

$$DBI = \left(\frac{1}{N}\right) * \text{tot}_m \left(\frac{bw_m + bw_n}{G_m n}\right) \tag{17}$$

where, n is the total number of clusters, $bw_m$ is the average distance between data points inside a cluster $m$ and its centroid, and $G_m n$ is the separation between cluster centroids of $m$ and $n$. The range of DBI scores is in '0' to '∞', where'0' indicating perfect clustering, and higher values indicating poorer clustering.

**Homogeneity**(HG) [49] is the degree to which measures the data points in each cluster accurately correspond to a single class. It is expressed as a score between 0.0 and 1.0, with a higher score indicating more homogeneous labeling of clusters. The mathematical formula to compute homogeneity score is as follows:

$$HG = \left[1 - \left(\frac{H(C|K)}{H(C)}\right)\right] \tag{18}$$

In the given equation, H(C|K) represents the conditional entropy of the class labels given the cluster assignments, while H(C) denotes the entropy of the class labels.

**Completeness** (COMP) [50] is a text clustering statistic that measures how evenly all members of a class are distributed among clusters. The equation for completeness is:

$$COMP = \frac{\sum\left(Q_{mn}^2\right)}{\sum\left(Q_m^2\right)} \tag{19}$$

where, $Q_{mn}$ denotes the count of points in class '$m$' are also appears in cluster $n$, and $Q_m$ is the count of points in class '$m$'. The completeness resulting score is always between 0 and 1, a higher completeness score indicates improved clustering performance.

**Normalized Mutual Information** (NMI) [51] is another often used clustering assessment metric is normalized mutual information, which calculates the mutual information between the true class labels and the predicted clusters, standardized by the value of the entropy of the class labels and the value of entropy of the predicted clusters is defined as:

$$NMI = \left[\frac{MI}{\left(H_{\text{true}} + H_{\text{pred}}\right)/2}\right] \tag{20}$$

where, $MI$ is the mutual information between the true class labels and the predicted clusters, $H_{\text{true}}$ is the entropy of the true class labels, and $H_{\text{pred}}$ is the entropy of the predicted clusters. Apart from this text clustering metrics, we used the Accuracy, Precision, Recall and F1-Score as the classification metrics to evaluate the performance of document.

### 4.3 Experimental setup

To test the selected datasets (AG News dataset, Reuters-21578 dataset and the IMDB reviews dataset) using Python prototypes, an optimal hardware setup with at least 32 GB RAM, a quad-core processor, and a GPU with at least 16 GB VRAM is configure to run the Deep Contextual Embedding Models. Python 3.6 or higher, PyTorch 1.6 or higher, Transformers library, NumPy, pandas, and scikit-learn are selected to design the proposed model.The experiments are evaluated from a local machine connected to the cloud-based platforms such as Google Colab or AWS EC2 instance with similar hardware specifications. Additionally, to ensure accurate results, it is preferred to fit the deep models on high volume data and tune the hyper-parameters using a validation set.

### 4.4 Comparative analysis

Recently, deep learning models have shown promising results in deep text clustering with syntax and semantics. In this comparative analysis, we will evaluate the performance of different text clustering algorithms such as Agglomerative Hierarchical Clustering (AHC) [52], Density-Based Spatial Clustering (DBSC) [53], Mean Shift Clustering (MSC) [54], Spectral Clustering(SC) [55], DBSCAN and KM, where each usinga different deep learning models such as Language Models (LM) [56], LatentDirichlet allocation (LDA) [57], Recurrent Neural Network (RNN) [58] and CNNs [45].

Specifically, the results of various deep contextual models such as LM+AHC, LDA+DBSCAN, CNN+KM, RNN+MSC, BERT+SC, DCEM (GPT2)+DBSCAN, and DCEM (GPT2)+KM are planned to evaluate and compare. The main objective of this comparative analysis is to detect which combination produces the high performance under various metrics such as SIL, CHI, DBI, HG, COMP and NMI. We selected the AG News, IMDB Reviews and Reuters-21578 benchmark datasets, to conduct the experiments with selected deep learning models and metrics. Tables 1-3 are presenting the obtained results from various deep text clustering models on three selected datasets.

For the AG News dataset, DCEM (GPT2) with KM performs the best, with a SIL score of 0.82, CHI of 37656, DBI of 0.17, HG of 0.78, COMP of 0.79, and NMI of 0.79.On the IMDB Reviews dataset, the best-performing model is DCEM (GPT2)+DBSCAN, with a SIL score of 0.63, CHI of 86921, DBI of 0.22, HG of 0.72, COMP of 0.68, and NMI of 0.54.On the Reuters-21578 dataset, DCEM (GPT2) with KM again performs the best, with a SIL score of 0.82, CHI of 19957, DBI of 0.18, HG of 0.72, COMP of 0.64, and NMI of 0.69. Finally the deep text clustering models that incorporate pre-trained language models such as GPT2 and BERT generally outperform traditional machine learning models like CNN and RNN, as well as unsupervised clustering algorithms like DBSCAN and KM, on all three datasets.

In order to showcase the overall clustering performance of various models, the averages are evaluated at each metric is shown in Table 4 and the same visualized in Figure 2.

The averaged results show that DCEM (GPT2)+KM algorithm has the highest performance for all metrics. Both DCEM (GPT2)+DBSCAN and DCEM (GPT2)+KM, which are based on GPT2, achieve the highest performance in terms of all metrics compared to other models. BERT+SC and LM+AHC also showed good performance for most of the metrics. However, LDA+DBSC, CNN+KM, and RNN+MSC showed lower performance compared to other models. Finally, the results obtained from the deep learning-based clustering algorithms, such as DCEM (GPT2)+KM and DCEM (GPT2)+DBSCAN, are promising methods for clustering tasks. These results are assuring the higher accuracy in clustering and documentation using the deep embedding contextual models.

We evaluated that why the DCEM(GPT2)+KM generated lower metric values on IMDB dataset compared to the Reuters and AG News datasets can be attributed to several valid reasons: i) The IMDB dataset consists of movie reviews, which may be more diverse and complex in terms of language, sentiment, and topics compared to the news articles in Reuters and AG News datasets, making it harder to cluster accurately. ii) Movie-related terms, names, and phrases used in IMDB reviews may not be as well-suited for general-purpose word embeddings like GPT-2, leading to difficulties in capturing contextual information effectively.

As discussed in previous section, to classify the documents after clustering, centroid of each cluster can be computed and used as a representative vector for the cluster. Similarity between the centroid vector [42] and each document in the cluster can be measured using a similarity metric such as cosine similarity. The document can be assigned to the relevant class based on the highest similarity score. There are several widely used metrics for evaluating the performance of document classification, including Accuracy, Precision, Recall, and F1-Score. After classifying the three datasets, the unified metrics obtained from each contextual model are presented in Table 5. The results from Table 5 show that, the performance of different deep contextual models for document classification using various metrics. The DCEM (GPT2) based contextual model outperformed other models (Figures 2-3) with an accuracy of 81.09% for DCEM (GPT2)+KM and 76.47% for DCEM (GPT2)+DBSCAN. These models also showed higher precision and F1 scores compared to other models. BERT+SC also performed well with an accuracy of 72.55%. However, LM+AHC showed lower accuracy than other models, but it still achieved an accuracy of 68.22%. The results suggest that the use of Deep Contextual Embedding Models can improve the accuracy of document classification using text clustering algorithms.A limitation we find from the results evaluation is the need for ample labeled data for effective deep contextual model training, which is challenging in domains with limited labeled data. Evaluation assumes ground truth labels, not always present in real-world scenarios, affecting performance assessment.

**Table 1.** Experimental results of various deep text clustering models on AG News dataset

| | AG News | | | | | |
|---|---|---|---|---|---|---|
| | SIL | CHI | DBI | HG | COMP | NMI |
| LM+AHC | 0.33 | 23040 | 0.46 | 0.67 | 0.51 | 0.47 |
| LDA+DBSC | 0.42 | 27612 | 0.34 | 0.51 | 0.49 | 0.34 |
| CNN+KM | 0.49 | 28626 | 0.49 | 0.57 | 0.43 | 0.43 |
| RNN+MSC | 0.61 | 24856 | 0.34 | 0.62 | 0.59 | 0.59 |
| BERT+SC | 0.65 | 35109 | 0.24 | 0.74 | 0.66 | 0.66 |
| DCEM (GPT2)+DBSCAN | 0.71 | 37792 | 0.22 | 0.81 | 0.69 | 0.69 |
| DCEM (GPT2)+KM | 0.82 | 37656 | 0.17 | 0.78 | 0.79 | 0.79 |

**Table 2.** Experimental results of various deep text clustering models on IMDB Reviews dataset

| | IMDB Reviews | | | | | |
|---|---|---|---|---|---|---|
| | SIL | CHI | DBI | HG | COMP | NMI |
| LM+AHC | 0.54 | 72992 | 0.59 | 0.42 | 0.49 | 0.38 |
| LDA+DBSC | 0.57 | 73507 | 0.31 | 0.44 | 0.56 | 0.49 |
| CNN+KM | 0.49 | 65839 | 0.37 | 0.57 | 0.62 | 0.53 |
| RNN+MSC | 0.56 | 57168 | 0.34 | 0.51 | 0.58 | 0.43 |
| BERT+SC | 0.61 | 80750 | 0.32 | 0.65 | 0.61 | 0.49 |
| DCEM (GPT2)+DBSCAN | 0.63 | 86921 | 0.22 | 0.72 | 0.68 | 0.54 |
| DCEM (GPT2)+KM | 0.62 | 86608 | 0.24 | 0.74 | 0.64 | 0.56 |

**Table 3.** Experimental results of various deep text clustering models on Reuters dataset

| | Reuters-21578 | | | | | |
|---|---|---|---|---|---|---|
| | **SIL** | **CHI** | **DBI** | **HG** | **COMP** | **NMI** |
| LM+AHC | 0.56 | 12211 | 0.47 | 0.32 | 0.45 | 0.32 |
| LDA+DBSC | 0.73 | 14634 | 0.43 | 0.66 | 0.52 | 0.64 |
| CNN+KM | 0.51 | 15171 | 0.51 | 0.41 | 0.51 | 0.41 |
| RNN+MSC | 0.64 | 13173 | 0.43 | 0.33 | 0.29 | 0.38 |
| BERT+SC | 0.69 | 18607 | 0.24 | 0.65 | 0.59 | 0.62 |
| DCEM (GPT2)+DBSCAN | 0.78 | 20029 | 0.21 | 0.69 | 0.57 | 0.58 |
| DCEM (GPT2)+KM | 0.82 | 19957 | 0.18 | 0.72 | 0.64 | 0.69 |

**Table 4.** Average performance metrics of various deep text clustering models on three datasets

| | **SIL** | **DBI** | **HG** | **COMP** | **NMI** |
|---|---|---|---|---|---|
| LM+AHC | 0.48±0.13 | 0.51±0.07 | 0.47±0.15 | 0.48±0.06 | 0.39±0.16 |
| LDA+DBSC | 0.57±0.15 | 0.36±0.06 | 0.54±0.12 | 0.52±0.05 | 0.49±0.15 |
| CNN+KM | 0.50±0.02 | 0.46±0.09 | 0.52±0.08 | 0.52±0.09 | 0.46±0.06 |
| RNN+MSC | 0.60±0.06 | 0.37±0.06 | 0.49±0.16 | 0.49±0.14 | 0.47±0.10 |
| BERT+SC | 0.65±0.03 | 0.27±0.06 | 0.68±0.05 | 0.62±0.04 | 0.59±0.10 |
| DCEM(GPT2)+DBSCAN | 0.71±0.08 | 0.24±0.02 | 0.72±0.07 | 0.65±0.07 | 0.60±0.14 |
| DCEM (GPT2)+KM | 0.75±0.09 | 0.20±0.03 | 0.75±0.05 | 0.69±0.08 | 0.68±0.10 |

**Table 5.** Performance Metrics of document classification models

| | **Accuracy** | **Precision** | **Recall** | **F1_Score** |
|---|---|---|---|---|
| LM+AHC | 68.22% | 71.15% | 68.52% | 69.81% |
| LDA+DBSC | 59.17% | 60.00% | 63.46% | 61.68% |
| CNN+KM | 63.02% | 69.23% | 63.16% | 66.06% |
| RNN+MSC | 57.74% | 64.81% | 59.32% | 61.95% |
| BERT+SC | 72.55% | 73.91% | 68.02% | 70.83% |
| DCEM (GPT2)+DBSCAN | 76.47% | 78.72% | 72.55% | 75.51% |
| DCEM (GPT2)+KM | 81.09% | 84.12% | 79.15% | 81.55% |

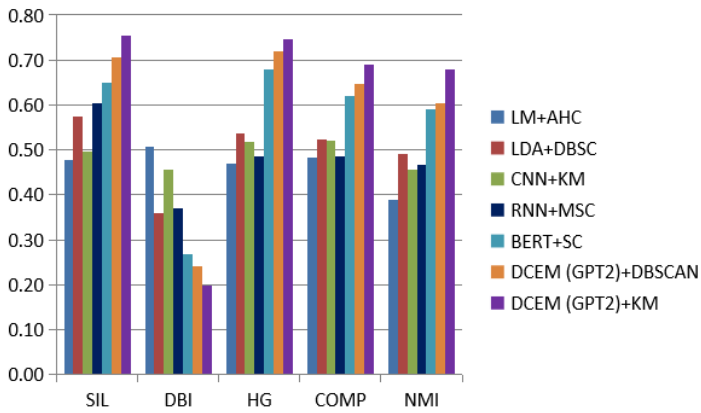


**Figure 2.** Visualization of the clustering performance of various deep text clustering models

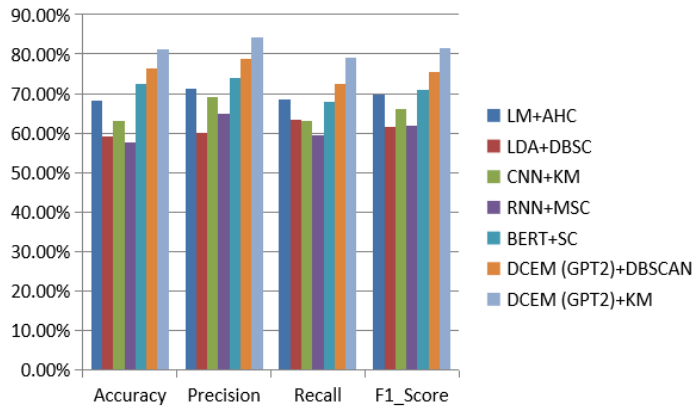


**Figure 3.** Visualization of the classification performance of various deep contextual models

## 5. CONCLUSION

The key objectives of this study were to improve text clustering and document classification accuracy by integrating pre-trained Deep Contextual Embedding Models (DCEMs) like GPT-2 with clustering algorithms. The methods involved generating contextual embeddings, applying multi-head self-attention, and dimensionality reduction. The overall findings demonstrated that the integrated approach outperformed traditional methods, achieving higher accuracy in text clustering and document classification on datasets like AG News, Reuters-21578, and IMDB reviews.

These findings from the experimental evaluation offer valuable insights into the effectiveness of various combinations of contextual embedding models for text clustering and document classification, which can be used to enhance the performance of text clustering algorithms. Experiments proven that, the ability of DCEM based clustering algorithms are effectively captured the semantic information in text data, coupled with the power of deep learning techniques, make them a potent tool for clustering and classification tasks in various domains.

A potential future research direction is to explore semi-supervised or unsupervised methods that reduce the reliance on large labeled datasets for deep contextual model training. Another direction is to investigate the potential benefits of incorporating domain-specific knowledge into Deep Contextual Embedding Models to improve their performance on domain-specific text data.

## REFERENCES


[1] Ceri, S., Bozzon, A., Brambilla, M., Della Valle, E., Fraternali, P., Quarteroni, S. (2013). An introduction to information retrieval. Web Information Retrieval, 3-11. https://doi.org/10.1007/978-3-642-39314-3_1

[2] Salton, G., Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5): 513-523. https://doi.org/10.1016/0306-4573(88)90021-0

[3] Aggarwal, C.C., Zhai, C. (2012). A survey of text clustering algorithms. In: Aggarwal, C., Zhai, C. (eds) Mining Text Data. Springer, Boston, MA. https://doi.org/10.1007/978-1-4614-3223-4_4

[4] Zhang, Y., Jin, R., Zhou, Z.H. (2010). Understanding bag-of-words model: A statistical framework. International Journal of Machine Learning and Cybernetics, 1: 43-52. https://doi.org/10.1007/s13042-010-0001-0

[5] Miles, S., Yao, L., Meng, W., Black, C.M., Miled, Z.B. (2022). Comparing PSO-based clustering over contextual vector embeddings to modern topic modeling. Information Processing & Management, 59(3): 102921. https://doi.org/10.1016/j.ipm.2022.102921

[6] Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv Preprint arXiv: 1810.04805. https://doi.org/10.48550/arXiv.1810.04805

[7] Keskar, N.S., McCann, B., Varshney, L.R., Xiong, C., Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. arXiv Preprint arXiv: 1909.05858. https://doi.org/10.48550/arXiv.1909.05858

[8] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv Preprint arXiv: 1907.11692. https://doi.org/10.48550/arXiv.1907.11692

[9] Clark, K., Luong, M.T., Le, Q.V., Manning, C.D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. arXiv Preprint arXiv: 2003.10555. https://doi.org/10.48550/arXiv.2003.10555

[10] Veyseh, A.P.B., Lai, V., Dernoncourt, F., Nguyen, T.H. (2021). Unleash GPT-2 power for event detection. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, 1: 6271-6282. https://doi.org/10.18653/v1/2021.acl-long.490.

[11] Liang, H., Sun, X., Sun, Y., Gao, Y. (2017). Text feature extraction based on deep learning: a review. EURASIP Journal on Wireless Communications and Networking, 2017(1): 1-12.. https://doi.org/10.1186/s13638-017-0993-1

[12] Seifollahi, S., Piccardi, M., Jolfaei, A. (2021). An embedding-based topic model for document classification. Transactions on Asian and Low-Resource Language Information Processing, 20(3): 1-13.. https://doi.org/10.1145/3431728

[13] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J. (2017). Attention is all you need in Advances in neural information processing systems. Search PubMed, 5998-6008. https://doi.org/10.48550/arXiv.1706.03762

[14] Wei, C.P., Yang, C.S., Hsiao, H.W. (2008). A collaborative filtering-based approach to personalized document clustering. Decision Support Systems, 45(3): 413-428. https://doi.org/10.1016/j.dss.2007.05.008

[15] Li, X., Roth, D. (2002). Learning question classifiers. COLING '02: Proceedings of the 19th international conference on Computational linguistics 1: 1-7. https://doi.org/10.3115/1072228.1072378

[16] Wong, W., Tsuchiya, N. (2020). Evidence accumulation clustering using combinations of features. MethodsX, 7: 100916. https://doi.org/10.1016/j.mex.2020.100916

[17] Mohammed, A.J., Yusof, Y., Husni, H. (2015). Document clustering based on firefly algorithm. Journal of Computer Science, 11(3): 453-465. https://doi.org/10.3844/jcssp.2015.453.465

[18] Karol, S., Mangat, V. (2013). Evaluation of text document clustering approach based on particle swarm optimization. Open Computer Science, 3(2): 69-90. https://doi.org/10.2478/s13537-013-0104-2

[19] Silva Filho, T.M., Pimentel, B.A., Souza, R.M., Oliveira, A.L. (2015). Hybrid methods for fuzzy clustering based on fuzzy c-means and improved particle swarm optimization. Expert Systems with Applications, 42(17-18): 6315-6328.https://doi.org/10.1016/j.eswa.2015.04.032

[20] Veremyev, A., Semenov, A., Pasiliao, E.L., Boginski, V. (2019). Graph-based exploration and clustering analysis of semantic spaces. *A*pplied Network Science, 4: 1-26. https://doi.org/10.1007/s41109-019-0228-y

[21] 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, 34(8): 3669-3680.. https://doi.org/10.1109/TKDE.2020.3028943.

[22] Naseem, U., Razzak, I., Musial, K., Imran, M. (2020). Transformer based deep intelligent contextual embedding for twitter sentiment analysis. Future Generation Computer Systems, 113: 58-69. https://doi.org/10.1016/j.future.2020.06.050
[23] Tan, Z., Chen, J., Kang, Q., Zhou, M., Abusorrah, A., Sedraoui, K. (2021). Dynamic embedding projection-gated convolutional neural networks for text classification. IEEE Transactions on Neural Networks and Learning Systems, 33(3): 973-982. https://doi.org/10.1109/TNNLS.2020.3036192
[24] Mehta, V., Bawa, S., Singh, J. (2021). WEClustering: Word embeddings based text clustering technique for large datasets. Complex & Intelligent Systems, 7: 3211-3224. https://doi.org/10.1007/s40747-021-00512-9
[25] Ravi, J., Kulkarni, S. (2023). Text embedding techniques for efficient clustering of twitter data. Evolutionary Intelligence, 1-11. https://doi.org/10.1007/s12065-023-00825-3
[26] Zhang, X., Li, Y., Wang, B., Li, M. (2021). A survey on model-based text clustering algorithms. Applied Sciences, 11(5): 2402. https://doi.org/10.3390/app11052402
[27] Xu, D., Tian, Y. (2015). A comprehensive survey of clustering algorithms. Annals of Data Science, 2: 165-193. https://doi.org/10.1007/s40745-015-0040-1
[28] Zhang, X., Chen, X., Feng, Y., Liu, F. (2020). Deep learning based text clustering: A comprehensive review. ACM Transactions on Intelligent Systems and Technology (TIST), 11(2): 1-32. https://doi.org/10.1145/3384743
[29] Min, E., Guo, X., Liu, Q., Zhang, G., Cui, J., Long, J. (2018). A survey of clustering with deep learning: From the perspective of network architecture. IEEE Access, 6: 39501-39514. https://doi.org/10.1109/ACCESS.2018.2855437
[30] Mudassar, S., Muhammad, K., Mahmood, T., Ahmad, I., Afzal, H. (2021). Deep learning for text clustering: A comprehensive review. IEEE Access, 9: 5504-5544. https://doi.org/10.1109/ACCESS.2020.3045989
[31] Wu, J., Wang, Y., Wu, Z., Wang, Z., Veeraraghavan, A., Lin, Y. (2018). Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In International Conference on Machine Learning. PMLR, pp. 5363-5372. https://doi.org/10.48550/arXiv.1806.09228
[32] Shi, T., Liu, Z. (2014). Linking GloVe with word2vec. arXiv Preprint arXiv: 1411.5595. https://doi.org/10.48550/arXiv.1411.5595
[33] Aggarwal, C.C., Zhai, C. (2012). A survey of text clustering algorithms. Mining Text Data, 77-128. https://doi.org/10.1007/978-1-4614-3223-4_4
[34] Kadhim, A.I., Cheah, Y.N., Ahamed, N.H. (2014). Text document preprocessing and dimension reduction techniques for text document clustering. In 2014 4th International Conference on Artificial Intelligence With Applications in Engineering and Technology, IEEE, Kota Kinabalu, Malaysia, pp. 69-73. https://doi.org/10.1109/ICAIET.2014.21
[35] Shao, T., Guo, Y., Chen, H., Hao, Z. (2019). Transformer-based neural network for answer selection in question answering, IEEE Access, 7: 26146-26156. https://doi.org/10.1109/ACCESS.2019.2900753
[36] Selva Birunda, S., Kanniga Devi, R. (2021). A review on word embedding techniques for text classification. Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020: 267-281. https://doi.org/10.1007/978-981-15-9651-3_23
[37] Chen, P.C., Tsai, H., Bhojanapalli, S., Chung, H.W., Chang, Y.W., Ferng, C.S. (2021). A simple and effective positional encoding for transformers. arXiv Preprint arXiv: 2104.08698. https://doi.org/10.48550/arXiv.2104.08698
[38] Sharaf Al-deen, H.S., Zeng, Z., Al-sabri, R., Hekmat, A. (2021). An improved model for analyzing textual sentiment based on a deep neural network using multi-head attention mechanism. Applied System Innovation, 4(4): 85. https://doi.org/10.3390/asi4040085
[39] Migenda, N., Möller, R., Schenck, W. (2021). Adaptive dimensionality reduction for neural network-based online principal component analysis. PloS One, 16(3): e0248896. https://doi.org/10.1371/journal.pone.0248896
[40] Devassy, B.M., George, S. (2020). Dimensionality reduction and visualisation of hyperspectral ink data using t-SNE. Forensic Science International, 311: 110194. https://doi.org/10.1016/j.forsciint.2020.110194
[41] Cui, M. (2020). Introduction to the k-means clustering algorithm based on the elbow method. Accounting, Auditing and Finance, 1(1): 5-8. https://dx.doi.org/10.23977/accaf.2020.010102
[42] Abualigah, L., Gandomi, A.H., Elaziz, M.A., Hamad, H.A., Omari, M., Alshinwan, M., Khasawneh, A.M. (2021). Advances in meta-heuristic optimization algorithms in big data text clustering. Electronics, 10(2): 101. https://doi.org/10.3390/electronics10020101
[43] Zhang, X., Zhao, J., LeCun, Y. (2015). Character-level convolutional networks for text classification. Advances in Neural Information Processing Systems, 28. https://doi.org/10.48550/arXiv.1509.01626
[44] Lewis, D.D. (1997). Reuters-21578 text categorization test-collection. https://doi.org/10.24432/C52G6M
[45] Yenter, A., Verma, A. (2017). Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. In 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, NY, USA, pp. 540-546. https://doi.org/10.1109/UEMCON.2017.8249013
[46] Shahapure, K.R., Nicholas, C. (2020). Cluster quality analysis using silhouette score. In 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), IEEE, Sydney, NSW, Australia, pp. 747-748. https://doi.org/10.1109/DSAA49011.2020.00096
[47] Wang, X., Xu, Y. (2019). An improved index for clustering validation based on Silhouette index and Calinski-Harabasz index. In IOP Conference Series: Materials Science and Engineering. IOP Publishing, 569(5): 052024. https://doi.org/10.1088/1757-899X/569/5/052024
[48] Xiao, J., Lu, J., Li, X. (2017). Davies bouldin index based hierarchical initialization K-means. Intelligent Data Analysis, 21(6): 1327-1338. https://doi.org/10.3233/IDA-163129
[49] Dubes, R.C., Zeng, G. (1987). A test for spatial homogeneity in cluster analysis. Journal of Classification, 4: 33-56. https://doi.org/10.1007/BF01890074
[50] Yu, Y., Ren, J., Zhang, Q., Yang, W., Jiao, Z. (2020).

Research on tire marking point completeness evaluation based on k-means clustering image segmentation. Sensors, 20(17): 4687. https://doi.org/10.3390/s20174687

[51] Knops, Z.F., Maintz, J.A., Viergever, M.A., Pluim, J.P. (2006). Normalized mutual information based registration using k-means clustering and shading correction. Medical Image Analysis, 10(3): 432-439. https://doi.org/10.1016/j.media.2005.03.009

[52] Bouguettaya, A., Yu, Q., Liu, X., Zhou, X., Song, A. (2015). Efficient agglomerative hierarchical clustering. Expert Systems with Applications, 42(5): 2785-2797. https://doi.org/10.1016/j.eswa.2014.09.054

[53] Liu, Q., Deng, M., Shi, Y., Wang, J. (2012). A density-based spatial clustering algorithm considering both spatial proximity and attribute similarity. Computers & Geosciences, 46: 296-309. https://doi.org/10.1016/j.cageo.2011.12.017

[54] Carreira-Perpinán, M.A. (2015). A review of mean-shift algorithms for clustering. arXiv Preprint arXiv: 1503.00687. https://doi.org/10.48550/arXiv.1503.00687

[55] Mokshin, V., Yakupov, D., Yakhina, Z. (2021). Comparison of spectral clustering methods for graph models of pipeline systems. In 2021 International Russian Automation Conference (RusAutoCon), IEEE, Sochi, Russian Federation, pp. 841-846. https://doi.org/10.1109/RusAutoCon52004.2021.9537494

[56] Irie, K., Zeyer, A., Schlüter, R., Ney, H. (2019). Language modeling with deep transformers. arXiv Preprint arXiv: 1905.04226. https://doi.org/10.21437/Interspeech.2019-2225

[57] Blei, D.M., Ng, A.Y., Jordan, M.I. (2003). Latent dirichlet allocation. Journal of Machine Learning Research, 3(Jan): 993-1022. https://doi.org/10.1145/34170564

[58] Irsoy, O., Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 720-728. http://dx.doi.org/10.3115/v1/D14-1080

## NOMENCLATURE

| Abbreviation | Description |
|---|---|
| BERT | Bidirectional Encoder Representations from Transformers |
| GPT-2 | Generative Pre-trained Transformer 2 |
| DBSCAN | Density-based spatial clustering of applications with noise |
| AHC | Agglomerative Hierarchical Clustering |
| NLP | Natural Language Processing |
| DCEM | Deep Contextual Embedding Models |
| LM | Language Models |
| PCA | Principal Component Analysis |
| t-SNE | t-Distributed Stochastic Neighbor Embedding |
| CHI | Calinski-Harabasz Index |
| DBI | Davies-Bouldin Index |
| HG | Homogenity |
| COMP | Completeness |
| NMI | Normalized Mutual Information |