algorithms

Article

Learning Distributed Representations and Deep Embedded Clustering of Texts

Shuang Wang [1,2,*], Amin Beheshti [1,*], Yufei Wang [1], Jianchao Lu [1], Quan Z. Sheng [1], Stephen Elbourn [1,3] and Hamid Alinejad-Rokny [1,4]

1 School of Computing, Macquarie University, Sydney, NSW 2109, Australia
2 School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
3 ITIC Pty Ltd., Sydney, NSW 2000, Australia
4 The Graduate School of Biomedical Engineering, UNSW, Sydney, NSW 2052, Australia
* Correspondence: shuangwang@seu.edu.cn (S.W.); amin.beheshti@mq.edu.au (A.B.)

Abstract: Instructors face significant time and effort constraints when grading students' assessments on a large scale. Clustering similar assessments is a unique and effective technique that has the potential to significantly reduce the workload of instructors in online and large-scale learning environments. By grouping together similar assessments, marking one assessment in a cluster can be scaled to other similar assessments, allowing for a more efficient and streamlined grading process. To address this issue, this paper focuses on text assessments and proposes a method for reducing the workload of instructors by clustering similar assessments. The proposed method involves the use of distributed representation to transform texts into vectors, and contrastive learning to improve the representation that distinguishes the differences among similar texts. The paper presents a general framework for clustering similar texts that includes label representation, K-means, and self-organization map algorithms, with the objective of improving clustering performance using Accuracy (ACC) and Normalized Mutual Information (NMI) metrics. The proposed framework is evaluated experimentally using two real datasets. The results show that self-organization maps and K-means algorithms with Pre-trained language models outperform label representation algorithms for different datasets.

Keywords: distributed representation; deep clustering; data augmentation; contrastive learning; artificial intelligence

check for updates

Citation: Wang, S.; Beheshti, A.; Wang, Y.; Lu, J.; Sheng, Q.Z.; Elbourn, S.; Alinejad-Rokny, H. Learning Distributed Representations and Deep Embedded Clustering of Texts. *Algorithms* **2023**, *16*, 158. https://doi.org/10.3390/a16030158

Academic Editors: Szymon Łukasik and Piotr A. Kowalski

Received: 27 January 2023
Revised: 23 February 2023
Accepted: 8 March 2023
Published: 13 March 2023

1. Introduction

Over the past years, different solutions have been proposed to reduce instructors' workload and facilitate the interaction between teachers and students [1]. An early example of such a technique is one-to-one interaction models, which are well suited for courses with a small number of students. However, assessment clustering becomes challenging when large classes with hundreds of students, or massive online open online courses, are involved. Accordingly, reducing instructors' workloads in the above situations becomes important in educational systems. To address this challenge, Artificial Intelligence (AI) has been used in intelligent tutoring systems to cluster similar texts. Examples of such systems can be seen in computer science courses, programming courses, or detecting plagiarism [2,3].

Unsupervised clustering is an important research topic in data science, machine learning and artificial intelligence. It has been used in a wide range of applications, ranging from document analysis [4] to image retrieval [5], for various systems. During the past few decades, a lot of clustering algorithms have been proposed, such as Kmeans [6], Gaussian mixture model [5], and Self Organising Map (SOM) [7]. Although unsupervised clustering algorithms have been studied widely, the performance of these traditional clustering methods generally deteriorate when the data dimensionality is high, due to unreliable similarity metrics. The assumptions underlying the dimensionality reduction techniques are generally independent of the assumptions of the clustering techniques, which is known as the curse of dimensionality. To solve this problem, a common way is to transform data from high dimensionality to lower dimensionality by applying dimension reduction techniques, like principle component analysis (PCA) [8] or feature selection methods [9]. Then, clustering is performed in the lower dimensional feature space. However, this scheme ignores the interconnection between feature learning and clustering. To address this issue, with the development of deep learning, such feature transformation can be achieved by using Deep Neural Networks (DNNs), referred to as deep clustering. There are two different stages for deep clustering, representation and clustering, which can be combined by two strategies. On the one hand, the clustering stage can be adopted after the representation stage [10]. On the other hand, the representation and clustering stages can be jointly combined [11].

Deep Embedded Clustering (DEC) [12] has been proposed to improve the deep feature representation results by simultaneously learning feature representations and cluster assignments. However, DEC does not make use of prior knowledge to guide the learning process. Semi-supervised Deep Embedded Clustering (SDEC) is proposed to overcome this limitation [13,14]. Some pioneering work proposes to simultaneously learn embedded features and perform clustering by explicitly defining a clustering-oriented loss. Though promising performance has been demonstrated in various applications, a vital ingredient has been overlooked, which is that the defined clustering loss may corrupt feature space. This, then, leads to non-representative meaningless features, and this, in turn, hurts clustering performance. To address this issue, an Improved Deep Embedded Clustering (IDEC) algorithm is proposed to take care of data structure preservation [15].

To cluster similar features, the Kmeans clustering algorithm was used in [12,13,15]. Although Kmeans is a simple and effective clustering algorithm, it is especially sensitive to the choice of K and the initial starting conditions [16]. With its natural property of the neural network, SOM can solve these problems. When combining the Kmeans algorithm with representation, the centres are determined which cannot adjust when training [12,13,15]. This kind of alternation learning method would suffer from the error accumulated during the alternation between the stages of representation learning and clustering, which results in sub-optimal clustering performance. Moreover, the aforementioned methods can only deal with offline tasks, i.e., the clustering is based on the whole dataset, which limits their application on large-scale online learning scenarios [17]. To conquer the aforementioned offline limitation, a one-stage online deep clustering method called Contrastive Clustering is proposed [17,18].

In this paper, we present a novel model by considering the contrastive learning loss and clustering loss to enable learning distributed representations of assessments with deeply embedded clustering to cluster similar assessments. Compared to [19,20], we propose using Pre-trained language models (PLMs) to represent assessment content as vectors because PLMs have been shown to be effective in improving natural language processing (NLP) tasks [21,22]. Specifically, we use BERT (Bidirectional Encoder Representation from Transformers) [23] for this purpose. In this context, Instance-wise Contrastive Learning (Instance-CL) has recently achieved remarkable success in self-supervised learning. Supervised Contrastive Learning (SCL) [24] term fine-tuned to the objective can significantly improve the performance on natural language understanding tasks from the GLUE benchmark [25]. Compared to [12,13,15], the effectiveness of SCL is studied in this paper. How to use SCL, however, can be challenging to cluster assessments. To address these challenges, we propose a general clustering framework that transforms assessments into vectors in terms of SCL loss and clustering assessments. The major contributions of the paper are summarized as follows:

- A deep embedding strategy is used jointly to cluster similar texts, which can improve the representation of texts.
- A loss, based on supervised contrastive learning and Kullback Leibler (KL) divergence [15], is used on clustering texts, which can not only cluster similar texts, but also distinguish the difference in a cluster.

- A general framework for clustering is proposed, which compares different clustering algorithms. SOM is first combined with deep representation. The results show the effectiveness of our proposed models in clustering texts which can significantly reduce instructor workload in marking assessments.

The rest of the paper is organized as follows. In Section 2, we discuss the related work, and in Section 3 we detail the problem descriptions and proposed models. The general framework is proposed in Section 4. Finally, the experimental results are reported in Section 5, followed by conclusions and future research directions in Section 6.

2. Related Work

To reduce instructors' workload, a series of text techniques such as text mining [26–29], text clustering [12,13,15], and text representation [23,30] were studied. Natural language processing (NLP) has been used widely in different applications. Neural models have become a dominant approach in NLP. For encoding texts, bi-directional LSTMs (BiLSTMs) [30] have been a dominant method among various neural networks, in terms of state-of-the-art studies in language modeling [31], machine translation [32], syntactic parsing [33] and question answering [34]. Sequential information flow for LSTM leads to relatively weaker power in capturing long-range dependencies, which results in lower performance during encoding longer sentences. To solve the problem, the sentence similar function was studied, based on word2vector similar elements [35], while word2vector only represented static word information. Sentence2vector considers information in a sentence while it ignores the information in assessments. Pre-trained BERT is easily fine-tuned with an additional output layer to create a state-of-the-art model for a wide range of tasks which suggests that BERT representations are potential universal text embedding. BERT [23] was applied to short answer grading. A python-based RESTful service, that utilizes the BERT model for text embedding, and Kmeans clustering, to identify sentences closest to the centroid for summary selection [36–38], provides students with a utility that could summarize lecture content, based on the desired number of sentences. A Sentence-BERT (SBERT), a modification of the pre-trained BERT network, that uses siamese and triplet network structures, was proposed to derive semantically meaningful sentence embedding that can be compared using cosine-similarity [39].

Deep Embedded Clustering. To improve the representation performance for clustering, there are two categories of deep clustering algorithms in existing studies: (i) clustering applied after having learned a representation, and (ii) approaches jointly optimized by feature learning and clustering. The first category of algorithms takes advantage of existing unsupervised deep learning frameworks and techniques. For example, autoencoder is used to learn low dimensional features of an original graph [40], and then runs the Kmeans algorithm to get clustering results [41]. A clustering loss is defined in the second category of algorithms, which simulates classification error in supervised deep learning. DEC [12] learns a mapping from the observed space to a low-dimensional latent space with deep neural networks, which can obtain feature representations and cluster assignments simultaneously and starts with pre-training an autoencoder and then removing the decoder. Existing deep embedded clustering algorithms only consider Kmeans algorithms and only work for offline tasks [17]. To solve this problem, the SOM algorithm was used in this paper, which has never before been considered for text representation in existing studies.

SOM Algorithms. SOM has the special property of effectively creating spatially organized internal representations of various features of input signals and their abstractions [42]. SOM is an automatic data analysis method that is widely applied to clustering problems and data exploration in industry, finance, natural sciences, and linguistics [43,44]. A SOM-similar network is designed to simultaneously implement encoding and clustering purposes on data samples, which are jointly trained with a Generative Adversarial Network (GAN) to optimize a newly defined clustering loss [45]. Principal Component Analysis (PCA) was combined with a SOM to determine the pull-off adhesion between concrete layers [46]. SOM algorithms have been used widely to cluster similar tasks [19,20,47].

However, there are no existing studies that combine SOM with BERT in text representation and clustering.

Contrastive Learning. Contrastive learning has recently become a dominant component in self-supervised learning for natural language processing. It aims to embed augmented versions of the same sample close to each other, while trying to distinguish embedding from different samples [48,49]. Therefore, different instances are isolated in the learned embedding space with local invariance, which is preserved for each instance. Contrastive learning at a particular instance, considers the level [48,49] while performing, whereas the method simultaneously conducts contrastive learning at both the instance- and cluster-level following the observation of label as representation [17]. Existing works aimed to learn a general representation, which is off-the-shelf for the downstream tasks. On the contrary, the method is specifically designed for clustering, which could be the first successful attempt at task-specified contrastive learning. In this paper, contrastive learning loss was first used on clustering texts with SOM, which has never before been considered in existing studies [23,50–54].

In this paper, we first combine text representation and contrastive learning with SOM algorithms in deep-embedded clustering, which has never been considered in the existing studies.

3. Problem Description and the Models

In this section, we first describe problem definitions and model building blocks. After that, contrastive learning is used to improve the performance of deep clustering.

3.1. Problem Description

The task in this paper is to cluster texts automatically. Formally, there are different texts, and our model should output the clustering result for it. To cluster similar assessments, BERT was studied to transform texts into vectors. To improve deep feature representations, a deeply embedded clustering technique was adopted. To improve the performance on clustering, Kullback Leibler (KL) divergence [15] and contrastive learning loss were calculated simultaneously.

3.2. The Framework

To cluster similar texts, the overall framework of our proposed model is shown in Figure 1. Firstly, the texts are represented as vectors PLMs. After that, the clustering technique is used to cluster vectors without the pre-defined number of clusters. To improve the representation performance, contrastive instances for texts are generated by vector representation models. For example, assume that there is assessment evaluation for the topic “Environmental protection” in a middle school. There are hundreds of assessments to mark which is a huge workload for instructors. To solve the problem, these assessments are first processed by the data augmentation technique, which can increase the diversity of an assessment. The contrastive learning loss is calculated either after vector representation or clustering, which improves the representation in future.

In this paper, we used pre-trained BERT [55] to represent words and sentences in the given answers. We also explored the usage of the BiLSTM model [30] for learning word-level and sentence-level information when we had few resources to fine-tune the whole BERT model. The procedure of text clustering, including representation and clustering, is shown in Figure 2.

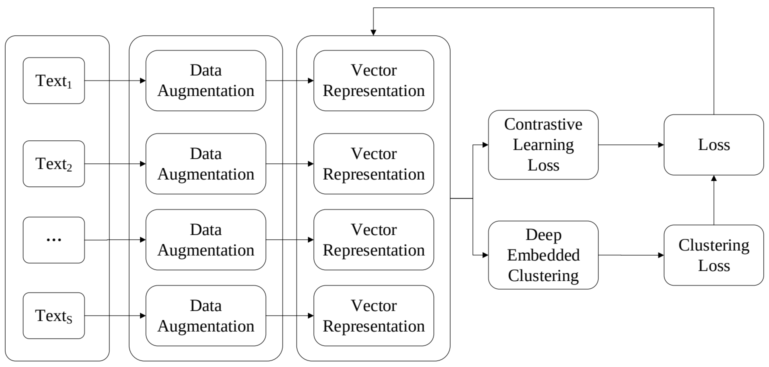

Figure 1. The proposed overall framework.

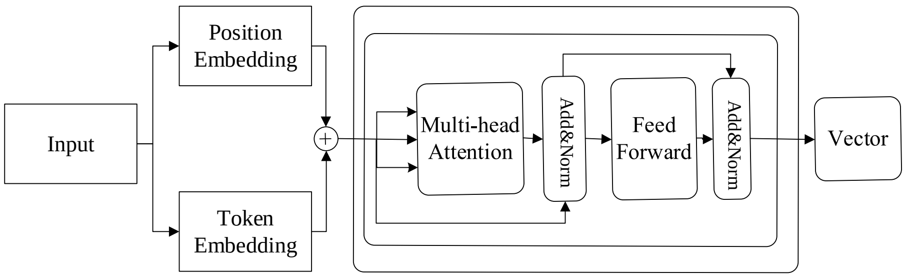

Figure 2. The procedure of text clustering.

Vector Representation

BERT. In the input, BERT adds a special leading token [cls] at the beginning of the input text (e.g., [cls] Artificial Intelligence is useful in education). The [cls] is used for sentence-level classification tasks during its pre-training stage, and its corresponding hidden vector is used as the sentence-level representation. All input tokens are represented as the additive combination of word embedding and positional embedding. Its core module is multi-head self-attention [56], where input tokens are represented as queries ($Q_i = XW_i^Q$), keys ($K_i = XW_i^K$) and values ($V_i = XW_i^V$) in the ith head. We have:

$$SelfAttention(Q_i, K_i, V_i) = softmax(\frac{Q_i K_i^T}{\sqrt{d_i^k}})V_i \tag{1}$$

$$MultiHead(Q, K, V) = cat(SelfAttention(Q_i, K_i, V_i))W^C \tag{2}$$

where $W_i^Q \in \Re^{d\times d_i^k}, W_i^K \in \Re^{d\times d_i^k}, W_i^V \in \Re^{d\times d_i^v}$ $W^C \in \Re^{\Sigma_{i=1}^{N_h} d_i^v \times d}$ are the trainable parameters. Finally, BERT has residual connection followed by layer-normalization [57]. A fully connected feed-forward network is applied to the matrix Z_1 as follows

$$FF(Z_1) = max(0, Z_1 W_1^F + b_1^F)W_2^F + b_2^F \tag{3}$$

where $W_1^F, b_1^F, W_2^F, b_2^F$ are parameter matrices. After feed-forward, Add&Norm is reused in the layer.

BERT performs well in transforming texts to vectors, although it consumes lots of server resources. To solve the problem, DistilBERT was adopted in this paper, which has the same general architecture as BERT [39]. The token-type embedding and the pooler are removed, while the number of layers is reduced by a factor of 2. Most of the operations used in the Transformer architecture (linear layer and layer normalization) are highly optimized in modern linear algebra frameworks and our investigations showed that variations on the last dimension of the tensor (hidden size dimension) had a smaller impact on computation efficiency (for a fixed parameter budget) than variations on other factors, like the number of layers.

3.3. Contrastive Learning

Contrastive learning aims to maximize the similarities of positive pair instances, and it can maximize the difference of negative pair ones. The characteristics of pair instances are defined by various rules. For example, when the pairs are within the same class, it is a positive pair. Otherwise, it is a negative pair. In this paper, the positive and negative pairs were constructed at the instance-level by data augmentations and embedding. In addition, the positive pairs consisted of samples either augmented or embedded from the same instance, while the negative pairs were from different instances.

Considering a mini-batch S, the instance contrastive learning loss is defined in terms of the augmented pairs S^a, while for a mini-batch S^v, the instance contrastive learning loss is defined in terms of the representation vectors. $i \in \{1, \cdots, M\}$ is denoted as the index of an arbitrary instance in set S. $j \in \{1, \cdots, M\}$ is the index of the other instance in S^a augmented from the same instance in the original set S. For pairs (x_i, x_j^a), there are 2M pairs in total. For a specific sample $x_i \in S$, there are 2M-1 pairs in total where (x_i, x_i^a) is a positive pair and others are negative pairs. Similarly, $x_j^v \in \{1, \cdots, 2M\}$ is the embedding sample of the other instance in S^v embedded from the same instance in the original set S. To alleviate the information loss induced by contrastive loss, we did not directly conduct contrastive learning on the feature matrix. Let z_i and z_j be the corresponding outputs of the head g, i.e., $z_i = g(\varphi(x_i)), j = i_1, i_2$. Then, for x_{i_1}, we tried to separate x_{i_2} apart from all negative instances in S^a by minimizing the following applied. The pair-wise similarity was measured by cosine distance:

$$s(z_i, z_j) = \frac{z_i^T z_j}{||z_i||_2 ||z_j||_2} \tag{4}$$

To optimize pair-wise similarities, without loss of generality, the loss for a given sample x_i^a is described as follows [17]:

$$loss_i^{co} = -log \frac{exp(s(z_i, z_{i_1}) \cdot \tau^{-1})}{\sum_{j=1}^{N} 1_{j \neq i} \cdot exp(s(z_i, z_j) \cdot \tau^{-1})} \tag{5}$$

where τ is the temperature parameter which controls the scale of the difference. According to Equation (5) and the mini-batch S, the contrastive loss is calculated using:

$$loss^{co} = \sum_{i=1}^{S} \frac{loss_i^{co}}{S} \tag{6}$$

3.4. Clustering

To cluster similar instances together, there are different methods, such as Label as representation and Kmeans algorithm in existing studies for deeply embedded clustering. Label as representation is defined when projecting a data sample into a space where the dimensionality equals the number of clusters with the ith element interpreted as the probability of belonging to the ith cluster, and the feature vector denotes its soft label accordingly. Kmeans algorithms have been used in [12,17,36], while the SOM algorithm with BERT has never been used in existing studies.

A Self Organizing Map [42] is a classic unsupervised learning algorithm that follows the principle of topological mapping. The main aim of SOM is to convert input signals of arbitrary dimension into one-dimensional or two-dimensional discrete mapping and the spatial position of the output layer neurons in the topological mapping corresponds to a specific feature of input space. The original idea of SOM is to simulate the way that vision systems work in the human brain, which is used for the organization and visualization of complex data. In general, SOM can capture the topology map structure which distinguishes the distances among different clusters and distribution of the input data to provide a clustering analysis. In this paper, a $M \times N$ self-organizing map is constructed. The architecture of SOM includes two layers: the input layer and the Kohonen output layer.

To describe SOM clearly, x is denoted as the input vector, while $W_{ij}(i \in \{1, \cdots, M\}, j \in \{1, \cdots, N\})$ is one of the neurons in the self-organisation map which is a unit vector. In addition, the shape of weights neurons is the same as the input vector z_k. To calculate the distance from z_k to the weights neurons, z_k is reduced to as a unit vector where z_{kj} is described as follows:

$$z_{kj} = \frac{z_{kj}}{\sum_{j=1}^{d} z_{kj}} \tag{7}$$

For each sample z_k, the distance d_{ij} from a neurons W_{ij} is calculated by:

$$d_{ij} = ||z_k - W_{ij}||^2 \tag{8}$$

The corresponding neuron which has the minimum distance is called the winner. The rates of the modifications at different nodes depend on the mathematical form of the function $h_{ci}(t)$, which is described as follows [43]:

$$h_{ci}(t) = a(t)exp(\frac{-sqdist(c,i)}{2\delta(t)}) \tag{9}$$

where $\alpha(t)$ is the learning rate at time t, $-sqdist(c,i)$ is the square of the geometric distance between the nodes c and i in the grid, and $\delta(t)$ is a monotonically decreasing function of t which controls the number of neighborhood nodes.

3.5. Clustering Loss

Suppose our data consists of $M \times N$ semantic categories and each category is characterized by its centroid in the $M \times N$ representation space, denoted as μ_{ij} $(i \in \{1,2,\cdots,M\}, j \in \{1,2,\cdots,N\})$. $e_j = g(x_j)$ is denoted as the representation of the instance x_j in the original set S^v. According to [58], the student's t-distribution is used to compute the probability q_{ij} of assigning x_j to one of the neurons W_{ij}:

$$q_{ji} = \frac{(1 + \frac{||e_j - \mu_i||_2^2}{\alpha})^{-\frac{\alpha+1}{2}}}{\sum_{k=1}^{M \times N}(1 + \frac{||e_j - \mu_k||_2^2}{\alpha})^{-\frac{\alpha+1}{2}}} \tag{10}$$

where α denotes the degree of freedom of the Student's t-distribution and $\alpha = 1$ by default [58]. According to SOM, the neurons are obtained. The neurons are refined by leveraging an auxiliary distribution proposed by [12]. p_{ji} is denoted as the auxiliary distribution which is described as follows:

$$p_{ji} = \frac{\frac{q_{ji}}{f_i}}{\sum_{i'} \frac{q_{ji'}}{f_{i'}}} \tag{11}$$

where $f_i = \sum_{i=1}^{M} q_{jk}, k \in \{1, \cdots, K\}$ is the interpreted as the soft cluster frequencies approximated within a mini-batch. The target distribution first sharpens. The soft-assignment probability q_{ji} is sharpened by raising it to the second power, and it is normalized by the associated cluster frequency. Learning is encouraged between high confidence cluster assignments and the bias which is simultaneously combated by imbalanced clusters.

To optimize the KL divergence between the cluster assignment probability and the target distribution, the cluster loss for each instance is described as follows:

$$loss_i^c = KL[p_j||q_j] = \sum_{i=1}^{K} p_{ji} log \frac{p_{ji}}{q_{ji}} \tag{12}$$

Since mini-batch instances were studied in this paper, the cluster loss for each mini-batch was obtained as follows, in terms of Equation (12):

$$loss^c = \sum_{i=1}^{M} \frac{loss_i^c}{M} \tag{13}$$

The overall loss function was obtained by Equations (6) and (13) which is described as follows:

$$l = loss^{co} + loss^c \tag{14}$$

4. Algorithms

To cluster similar assessments, Distill BERT was first used to transform texts into vectors with a text augmentation technique. After augmentation, contrastive learning and clustering techniques were adopted to cluster similar texts. Since for different clustering algorithms the efficiency is different. We propose three different clustering algorithms with the proposed model. The proposed algorithm framework is described in Algorithm 1. To calculate the efficiency of Algorithm 1, the adopt Accuracy (ACC) and Normalized Mutual Information (NMI) are measured. NMI is calculated by NMI$(Y, C) = \frac{2\times I(Y;C)}{H(Y)+H(C)}$ where Y is the real label, C is the clustering label, H(.) is the cross entropy, and I(.;.) is the mutual information.

To describe the algorithm framework clearly in Algorithm 1, the input data contains dataset S, training epochs E, temperature τ in contrastive learning and the structure of augmentation, while the output results are ACC and NMI. A mini-batch $\{x_i\}$ of texts is selected from dataset S. For each input sample x_i, two random augmentations are generated where $x_i^a = A_j^S(x_i)(j = 1, 2)$, which represents a different view of the text and contains some subset of the information in the original sample. x_i^1, x_i^2 $(i \in \{1, 2, \cdots, |S|\})$ are denoted as the augmented text of an assessment x_i. After text augmentation, the contrastive loss is calculated according to the temperature τ. Texts are transformed into vectors by DistilBERT. The predicted labels are obtained after the clustering algorithms. By comparing predicted labels to ground truth labels, the clustering loss is calculated. According to the instance loss and clustering loss, the overall loss is obtained. The ACC and NMI metrics are calculated.

The deeply embedded clustering framework is described in Algorithm 1. For vector clustering techniques, there are three different strategies: label as representation, Kmeans algorithm, and SOM algorithm. For label as representation, the dimension of vectors is projected to the number of clusters by multiple neuron networks. The index of the maximum values of the vector is represented as the label of clusters. After all of the vectors are projected to the fixed dimensions, Kmeans and SOM algorithms are used to cluster similar vectors together, respectively. To calculate the clustering centers for the label as representation algorithms, Kmeans and SOM algorithms are adopted, respectively.

The label as a representation algorithm is shown in Algorithm 2. The input of the algorithm is the matrices Z, Z_1, and Z_2 which are made up of $z_i, z_i^1, z_i^2 (i \in \{1, \cdots, |S|\})$ where z_i, z_i^1, z_i^2 are represented by DistilBERT. The labels L, L_1, L_2 are initialized first. According to the idea of the label as representation, all the represented vectors are projected to K dimensions. The index of the maximum value selects the label for the projected vectors. The time complexity of the Algorithm 2 is $O(K|S|)$.

The Kmeans clustering algorithm is described in Algorithm 3. To describe Algorithm 3 clearly, $c_1, \cdots, c_k$ are denoted as the initial centroids. All the vectors in Z are compared with K centroids. The vector with the minimum distance to the centroid c_i is assigned to C_i. After all the vectors are assigned, new centroids are calculated in terms of new C_i $(i \in \{1, \cdots, K)$. The procedure is repeated until the conditions are not satisfied. The label L for Z is obtained with the final centroids. The labels L_1, L_2 for Z_1 and Z_2 are obtained, similarly. The time complexity for Algorithm 3 is $O(|S|Kt)$.

Algorithm 1: Deep Embedded Clustering Framework

```
Input: Data set $S$, Training epochs $E$, Temperature $\tau$, Structure of augmentation $A^S$
Output: ACC, NMI.
for $e = 1$ to $E$ do
    Sample a mini-batch $\{x_i\}(i \in \{1, 2, \cdots, M\})$ from dataset $S$;
    Sample two augmentations $A_1^S, A_2^S$;
    for $i = 1$ to $|S|$ do
        $x_i, x_i^1, x_i^2$ is represented by DistilBERT;
    Compute the instance loss by Equation (5);
    Calculate the clustering centers;
    Cluster these vectors by different clustering methods;
    Calculate the predicted labels for all vectors;
    Compute the clustering loss by Equation (12);
    Compute the overall loss by Equation (13);
    Calculate the ACC, NMI by comparing predicted labels to ground truth labels;
    Update gradients to minimize the overall loss;
return ACC, NMI.
```

Algorithm 2: Label as Representation Algorithm

```
Input: Matrices $Z, Z^1, Z^2$
Initialize labels $L, L_1, L_2$;
for $i = 1$ to $|S|$ do
    $z_i$ is projected to $K$ dimensions;
    $z_i^1$ is projected to $K$ dimensions;
    $z_i^2$ is projected to $K$ dimensions;
    $ind_{max} \leftarrow 1, ind_{max}^1 \leftarrow 1, ind_{max}^2 \leftarrow 1$;
    $z_{max} \leftarrow z_i[1], z_{max}^1 \leftarrow z_i^1[1], z_{max}^2 \leftarrow z_i^2[1]$;
    for $j = 2$ to $K$ do
        if $z_{max} < z_i[j]$ then
            $z_{max} \leftarrow z_i[j]$;
            $ind_{max} \leftarrow j$;
    $L_i \leftarrow ind_{max}$;
    for $j = 2$ to $K$ do
        if $z_{max}^1 < z_i^1[j]$ then
            $z_{max}^1 \leftarrow z_i^1[j]$;
            $ind_{max}^1 \leftarrow j$;
    $L_i^1 \leftarrow ind_{max}^1$;
    for $j = 2$ to $K$ do
        if $z_{max}^2 < z_i^2[j]$ then
            $z_{max}^2 \leftarrow z_i^2[j]$;
            $ind_{max}^2 \leftarrow j$;
    $L_i^2 \leftarrow ind_{max}^2$;
return $L, L_1, L_2$.
```

The SOM clustering algorithm is described in Algorithm 4. The input data includes parameters M, N for the map, the input matrix Z, the iteration times T, and the parameter α. The weight matrix W is initialized where each raw is a unit vector. To compare the input matrix W to Z, each raw of Z is normalized to a unit vector. By comparing Z to W, the distance from z_i to all the neurons is calculated. The neuron with the minimum distance is selected as the winner. The label of z_i is determined. The learning rate is updated. After that, $h_{ci}(t)$ is calculated by Equation (9) with new $\delta(t)$. The weights are updated by the new $h_{ci}(t)$. The procedure is repeated until $t = T$. The time complexity of Algorithm 4 is $O(MN|S|T)$.

Algorithm 3: Kmeans Clustering Algorithm

```
Input: Matrices $Z, K$
1  Initialize $K$ centroids $c_1, \cdots, c_K$ randomly;
2  $t \leftarrow 0$;
3  while Conditions are satisfied do
4      $c'_1 \leftarrow c_1, \cdots, c'_K \leftarrow c_K$;
5      Initialize empty sets $C_1, \cdots, C_K$;
6      for $i = 1$ to $|S|$ do
7          $d_1 \leftarrow ||z_i - c_1||^2$, $ind \leftarrow 1$;
8          for $j = 2$ to $K$ do
9              if $d_1 < ||z_i - c_j||^2$ then
10                 $d_1 \leftarrow ||z_i - c_j||^2$;
11                 $ind \leftarrow j$;
12         $C_j \leftarrow C_j \cup z_{ind}$;
13     $c_1 \leftarrow avg(C_1), \cdots, c_K \leftarrow avg(C_K)$;
14     $t \leftarrow t + 1$
15 return $L$.
```

Algorithm 4: SOM Clustering Algorithm

```
Input: $M, N, Z$, iteration times $T$,$\alpha$
1  Initialize the weights $W(0)$;
2  Normalize $Z$ by equation (7);
3  for $t = 1$ to $T$ do
4      for $i = 1$ to $|S|$ do
5          $d \leftarrow 100$, $L \leftarrow 0_{1\times|S|}$;
6          for $j = 1$ to $M$ do
7              for $k = 1$ to $N$ do
8                  if $||z_i - W_{jk}||^2 < d$ then
9                      $d \leftarrow ||z_i - W_{jk}||^2$;
10                 $L_i \leftarrow (j-1) * N + k$;
11     Calculate $h_{ci}(t)$ by Equation (9);
12     Update learning rate $l_r(t) = \alpha * (1 - \frac{t}{T})$;
13     Update weights by Equation (10);
14 return $L$.
```

5. Experiments

In this section, the experimental setup is described and different clustering strategies are compared.

5.1. Experimental Setup

In the proposed deep-embedded clustering algorithm, there are different parameters and clustering components and we present our experiments to calibrate the parameters and components using different datasets, which include the AgNews dataset [59], and StackOverflow dataset [60]. The AgNews dataset is a subset of news titles that contains 4 topics. AgNews is a collection of more than 1 million news articles gathered from more than 2000 news sources. The dataset is used in data mining, including clustering, classification, information retrieval and so on. StackOverflow is a subset of the challenge data published by Kaggle, which contains 20,000 question titles associated with 20 different categories [60]. For each question, it includes: Question ID, Creation date, Closed date, Number of answers and so on. We then present experimental results to compare the proposed deep-embedded clustering algorithm against three existing algorithms using real-life instances. Different algorithms are implemented in the Python programming language with the Pytorch library on a single NVIDIA P100 GPU with 16G memory.

The Adam optimizer, with a batch size of 400, the distilbert-base-nli-stsb-mean-tokens, and the maximum input length are used the same as in [61,62]. The distilbert-base-nli-stsb-mean-tokens in the Sentence Transformers library was used as the backbone [39], and we set the maximum input length to 32. Different learning rates were conducted to optimize the Clustering head and we set $\alpha = 1$ for the dataset. As mentioned in Section 3, different τ were used to optimize the contrastive loss. We tried different τ values in the range of (0, 1]. For fair comparison between the deeply embedded clustering framework and its components or variants, the clustering performance was analyzed for each of them by applying Kmeans, labeled as a representation with clustering centers by Kmeans (KmeansR), and SOM (SOMR), labeled as representations with clustering centers by SOM, and SOM, respectively.

5.2. Algorithm Comparison

To analyze the experimental results on the Agnews and StackOverflow datasets, different parameters, such as learning rates, learning rate scale (lr scale), and the temperature τ, were calibrated and compared with different values on these different algorithm components.

5.2.1. Experimental Results on Learning Rates

To calibrate the learning rates, the values were set from 10^{-7} to 10^{-3} for the Avgnews dataset. The experimental results are shown in Table 1. According to Table 1, With NMI metrics, when the learning rate was 10^{-5}, the NMI was the highest, with 0.824, while for SOMR, the highest NMI was 0.743 with learning rate 10^{-3}. For Kmeans algorithm, the highest NMI was 0.822 with a learning rate 10^{-5} while for KmeansR, the highest NMI was 0.731 with a learning rate 10^{-6}. Comparing SOM, SOMR, Kmeans, and KmeansR, the highest NMI was 0.824, obtained by SOM, followed by 0.822 obtained by Kmeans. For ACC, the highest value of SOM was 0.947 with the learning rate 10^{-5} while it was 0.915 of SOMR with the learning rate 10^{-3}. For Kmeans, the highest value of ACC was 0.944 with the same learning rate 10^{-5} as SOM, while for SOMR, the highest ACC was 0.904 with the learning rate 10^{-6}. Comparing all algorithm components, the highest value was 0.947, obtained by SOM.

Table 1. Learning rate Calibration and Algorithm Component Comparison (10^{-7}–10^{-3}).

Dataset	Metrics	Algorithm	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}
	NMI	SOM	0.763	0.739	**0.824**	0.565	0.625
	NMI	SOMR	0.724	0.729	0.696	0.491	0.743
	NMI	Kmeans	0.767	0.744	0.822	0.731	0.454
	NMI	KmeansR	0.727	0.731	0.685	0.718	0.296
Avgnews	ACC	SOM	0.922	0.909	**0.947**	0.804	0.836
	ACC	SOMR	0.899	0.903	0.887	0.767	0.914
	ACC	Kmeans	0.923	0.926	0.944	0.899	0.694
	ACC	KmeansR	0.901	0.904	0.881	0.865	0.494

Bold implies the best values for the NMI and ACC metrics.

In regard to calibrating the learning rates for the StackOverflow dataset, the experimental results are shown in Figure 3. According to Figure 3, the SOM and SOMR algorithms were affected by the learning rates, while the Kmeans and KmeansR algorithms had little effect on the learning rates. With NMI metrics, when the learning rate was 10^{-6}, the NMI was the highest, with 0.662, while for SOMR, the highest NMI was 0.564 with learning rate 10^{-5}. For the Kmeans algorithm, the highest NMI was 0.667 with learning rates 10^{-6}, 10^{-4}, and 10^{-3}, while for KmeansR, the highest NMI was 0.564 with the same corresponding learning rates. Comparing SOM, SOMR, Kmeans, and KmeansR, the highest NMI was 0.667, obtained by Kmeans, followed by 0.662, obtained by SOM. For ACC, the highest value of SOM was 0.710 with the learning rate 10^{-6}, while it was 0.611 of SOMR, with the learning rate 10^{-6}. For Kmeans, the highest value of ACC was 0.711 with the same learning rate 10^{-6} as SOM, while for SOMR, the highest ACC was 0.619, with the learning rate 10^{-6}, 10^{-5}, and 10^{-3}. Comparing all algorithm components, the highest value was 0.711, obtained by the Kmeans algorithm.

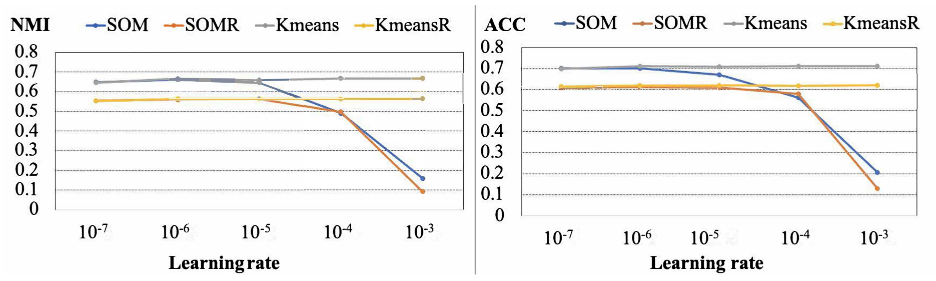

Figure 3. Algorithm Component Comparison with StackOverflow for NMI and ACC.

5.2.2. Experimental Results on Learning Rate Scales

To calibrate the learning rate scale on different clustering algorithms for the Avgnews dataset, the values were set from 50 to 500. The experimental results are shown in Table 2. According to Table 2, with NMI metrics, when the learning rate scale was 500, the NMI for the Avgnews dataset was the highest with 0.883 for SOM, and for SOMR, the highest NMI was 0.712. For the Kmeans algorithm, the highest NMI was 0.904, with a learning rate scale of 500, and for KmeansR, the highest NMI was 0.744, with the same learning rate scale. Comparing SOM, SOMR, Kmeans, and KmeansR, the highest NMI was 0.904, obtained by *Kmeans*, followed by 0.883, obtained by SOM. According to Table 2, with ACC metrics, when the learning rate scale was 500, the NMI was the highest, with 0.968, for SOM and for SOMR, the highest NMI was 0.891. For the Kmeans algorithm, the highest ACC was 0.974, with a learning rate scale of 500, and for KmeansR, the highest ACC was 0.905 with the same learning rate scale. Comparing SOM, SOMR, Kmeans, and KmeansR, the highest NMI was 0.974, obtained by Kmeans, followed by 0.968, obtained by SOM.

Table 2. Learning Rate Scale Calibration and Algorithm Component Comparison (50–500).

Dataset	Algorithms	Metrics	50	100	150	200	250	500
	SOM	NMI	0.711	0.824	0.827	0.829	0.862	0.883
	SOMR	NMI	0.609	0.694	0.670	0.662	0.662	0.712
AvgNews	Kmeans	NMI	0.823	0.822	0.836	0.844	0.856	**0.904**
	KmeansR	NMI	0.699	0.685	0.664	0.663	0.669	0.744
	SOM	ACC	0.901	0.947	0.947	0.947	0.961	0.968
	SOMR	ACC	0.882	0.837	0.873	0.871	0.871	0.891
	Kmeans	ACC	0.944	0.944	0.950	0.953	0.958	**0.974**
	KmeansR	ACC	0.888	0.881	0.870	0.870	0.868	0.905

Bold implies the best values for the NMI and ACC metrics.

According to Figure 4, the SOM algorithm was a little better than Kmeans algorithm in NMI and ACC metrics. Similarly, SOMR was slightly better than KmeansR. For the StackOverflow dataset, with NMI metrics, when the learning rate scale was 250, the NMI for the StackOverflow dataset was the highest, with 0.670 for SOM, and for SOMR, the highest NMI was 0.570.

For the Kmeans algorithm, the highest NMI was 0.680 with a learning rate scale of 250, and for KmeansR, the highest NMI was 0.568, with the same learning rate scale. Comparing SOM, SOMR, Kmeans, and KmeansR, the highest NMI was 0.680, obtained by Kmeans, followed by 0.670, obtained by SOM. According to Figure 4, with ACC metrics, when the learning rate scale was 250, the NMI was the highest, with 0.716, for SOM and for SOMR, the highest NMI was 0.620, with a learning rate scale of 500. For the Kmeans algorithm, the highest ACC was 0.724 with a learning rate scale of 500 and for KmeansR, the highest ACC was 0.624 with the same learning rate scale. Comparing SOM, SOMR, Kmeans, and

KmeansR, the highest NMI was 0.724, obtained by Kmeans, followed by 0.716, obtained by SOM.

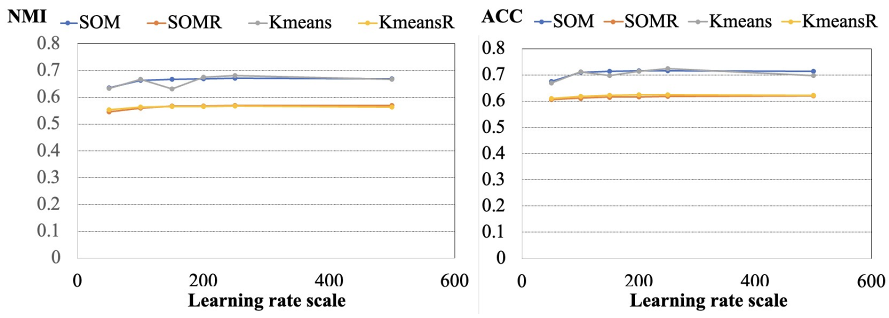

Figure 4. Learning Rate Scale Calibration and Algorithm Component Comparison with StackOverflow for NMI and ACC .

5.2.3. Experimental Results on Temperature

To calibrate the temperature τ, the values were set from 0.4 to 0.9. The experimental results are shown in Table 3. According to Table 3, with NMI metrics, when the temperature was 0.9, the NMI was the highest, with 0.966, for SOM and for SOMR, the highest NMI was 0.821. For the Kmeans algorithm, the highest NMI was 0.909, with a temperature of 0.9, and for KmeansR, the highest NMI was 0.907, with the same temperature. Comparing SOM, SOMR, Kmeans, and KmeansR, the highest NMI was 0.966, obtained by Kmeans, followed by 0.953, obtained by Kmeans. According to Table 3, with ACC metrics, when the temperature was 0.9, the ACC was the highest, with 0.992, for SOM and for SOMR, the highest NMI was 0.949. For the Kmeans algorithm, the highest ACC was 0.989, with a temperature of 0.9, and for KmeansR, the highest ACC was 0.939, with the same temperature. Comparing SOM, SOMR, Kmeans, and KmeansR, the highest ACC was 0.992, obtained by SOM, followed by 0.989, obtained by Kmeans.

Table 3. Temperature Calibration and Algorithm Component Comparison. (0.4–0.9)

Dataset	Temperature	Metrics	0.4	0.5	0.6	0.7	0.8	0.9
	SOM	NMI	0882	0.883	0.881	0.948	0.958	**0.966**
	SOMR	NMI	0.679	0.712	0.735	0.784	0.821	0.844
	Kmeans	NMI	0.881	0.904	0.905	0.905	0.909	0.953
Avgnews	KmeansR	NMI	0.765	0.744	0.764	0.784	0.807	0.819
	SOM	ACC	0.967	0.968	0.967	0.987	0.990	**0.992**
	SOMR	ACC	0.874	0.891	0.902	0.920	0.938	0.949
	Kmeans	ACC	0.966	0.974	0.975	0.975	0.976	0.989
	Kmeans	ACCR	0.885	0.905	0.914	0.922	0.931	0.939

Bold implies the best values for the NMI and ACC metrics.

The experimental results are shown in Figure 5 for the StackOverflow dataset. According to Figure 5, parameter temperature had little effect on SOM, SOMR, Kmeans, and KmeansR algorihtms. With NMI metrics, when the temperature was 0.9, the NMI was the highest, with 0.699. for SOM and for SOMR, the highest NMI was 0.598. For the Kmeans algorithm, the highest NMI was 0.716, with a temperature of 0.7, and for KmeansR, the highest NMI was 0.577 with the same temperature. Comparing SOM, SOMR, Kmeans, and KmeansR, the highest NMI was 0.716, obtained by Kmeans, followed by 0.699, obtained by SOM. According to Figure 5, with ACC metrics, when the temperature was 0.9, the ACC was the highest, with 0.737, for SOM and for SOMR, the highest NMI was 0.639. For Kmeans algorithm, the highest ACC was 0.757, with a temperature of 0.7, and for KmeansR, the highest ACC was 0.632, with the same temperature. Comparing SOM, SOMR, Kmeans, and KmeansR, the highest ACC was 0.757, obtained by Kmeans, followed by 0.737, obtained by SOM.

According to the above experiments, Kmeans and SOM algorithms were always better than SOMR and KmeansR. For the Avgnews dataset, SOM performed the best, while for the StackOverflow dataset, Kmeans performed the best.

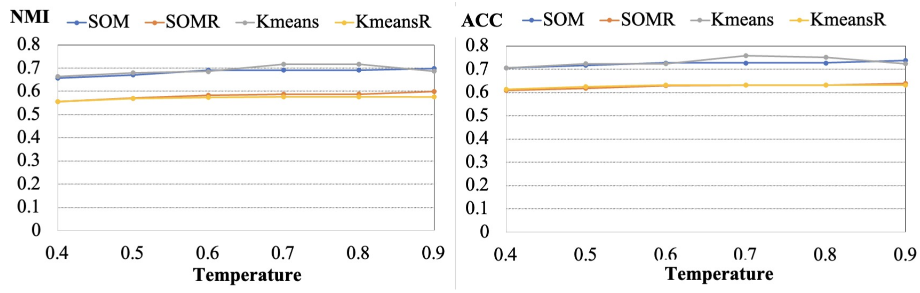

Figure 5. Learning Rate Scale Calibration and Algorithm Component Comparison with StackOverflow for NMI and ACC .

6. Conclusions

There are various kinds of texts in education and news systems. To increase efficiency, how to cluster these texts is important. In this paper, a text clustering problem is considered which combines BERT with different clustering techniques and contrastive learning. To solve the problem, we proposed a general framework that uses constructive learning on deep embedding to cluster similar texts. The NLP technique is used to represent texts to vectors. Different clustering algorithm components, which include label as representation, Kmeans, and SOM (Self Organizing Map) algorithms, are used. By combining contrastive learning with representation and clustering algorithms, similar texts in the same group can be distinguished. Different experiments were conducted to compare the NMI (Normalized Mutual Information) and ACC (Accuracy) for different datasets. The results showed the effectiveness of our proposed model in clustering texts, which implies that the performance of SOM is better than SOM and the performance of Kmeans is better than KmeansR. In future, we will further study the relationship among different texts in a cluster. In addition, we also plan to construct and study different relationship graphs for texts.

Author Contributions: Conceptualization, S.W. and A.B.; methodology, A.B. and S.E.; validation, Y.W. and J.L.; formal analysis, Q.Z.S. and H.A.-R.; investigation, H.A.-R.; resources, S.E.; writing—original draft preparation, S.W.; writing—review and editing, A.B. and Q.Z.S.; supervision, A.B.; project administration, A.B.; funding acquisition, S.E. and A.B. All authors have read and agreed to the published version of the manuscript.

Funding: We acknowledge the Centre for Applied Artificial Intelligence at Macquarie University and ITIC Pty Ltd for funding this research. The paper is supported by the Natural Science Foundation of Jiangsu Province of China (Grants No. BK20220803).

Data Availability Statement: The experimental code is available on the https://www.researchgate.net/publication/367284597_DEC_som (accessed on 30 January 2023).

Conflicts of Interest: The authors declare that there are no conflict of interests, we do not have any possible conflicts of interest.

References

1. Wang, S.; Beheshti, A.; Wang, Y.; Lu, J.; Sheng, Q.Z.; Elbourn, S.; Alinejad-Rokny, H.; Galanis, E. Assessment2Vec: Learning Distributed Representations of Assessments to Reduce Marking Workload. In Proceedings of the International Conference on Artificial Intelligence in Education, Utrecht, The Netherlands, 14–18 June 2021 ; Springer: Cham, Switzerland, 2021; Volume 12749, pp. 384–389.
2. Singh, A.; Karayev, S.; Gutowski, K.; Abbeel, P. Gradescope: A fast, flexible, and fair system for scalable assessment of handwritten work. In Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale, Cambridge, MA, USA, 20–21 April 2017; pp. 81–88.
3. Piech, C.; Gregg, C. BlueBook: A computerized replacement for paper tests in computer science. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, MD, USA, 21–24 February 2018; pp. 562–567.

4. Pessutto, L.R.C.; Vargas, D.S.; Moreira, V.P. Multilingual aspect clustering for sentiment analysis. *Knowl.-Based Syst.* **2020**, *192*, 105339. [CrossRef]
5. Dilokthanakul, N.; Mediano, P.A.; Garnelo, M.; Lee, M.C.; Salimbeni, H.; Arulkumaran, K.; Shanahan, M. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv* **2016**, arXiv:1611.02648.
6. MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, 21 June–18 July 1965 and 27 December 1965–7 January 1966; Volume 1, pp. 281–297.
7. Vesanto, J.; Alhoniemi, E. Clustering of the self-organizing map. *IEEE Trans. Neural Netw.* **2000**, *11*, 586–600. [CrossRef]
8. Wold, S.; Esbensen, K.; Geladi, P. Principal component analysis. *Chemom. Intell. Lab. Syst.* **1987**, *2*, 37–52. [CrossRef]
9. Ren, Y.; Zhang, G.; Yu, G.; Li, X. Local and global structure preserving based feature selection. *Neurocomputing* **2012**, *89*, 147–157. [CrossRef]
10. Li, Y.; Cai, J.; Wang, J. A Text Document Clustering Method Based on Weighted BERT Model. In Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 12–14 June 2020; Volume 1, pp. 1426–1430.
11. Fard, M.M.; Thonet, T.; Gaussier, E. Deep k-means: Jointly clustering with k-means and learning representations. *Pattern Recognit. Lett.* **2020**, *138*, 185–192. [CrossRef]
12. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised deep embedding for clustering analysis. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 478–487.
13. Ren, Y.; Hu, K.; Dai, X.; Pan, L.; Hoi, S.C.; Xu, Z. Semi-supervised deep embedded clustering. *Neurocomputing* **2019**, *325*, 121–130. [CrossRef]
14. Beheshti, A. Knowledge base 4.0: Using crowdsourcing services for mimicking the knowledge of domain experts. In Proceedings of the 2022 IEEE International Conference on Web Services (ICWS), Barcelona, Spain, 10–16 July 2022; pp. 425–427.
15. Guo, X.; Gao, L.; Liu, X.; Yin, J. Improved Deep Embedded Clustering with Local Structure Preservation. In Proceedings of the IJCAI, Melbourne, Australia, 19–25 August 2017; pp. 1753–1759.
16. Pena, J.M.; Lozano, J.A.; Larranaga, P. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognit. Lett.* **1999**, *20*, 1027–1040. [CrossRef]
17. Li, Y.; Hu, P.; Liu, Z.; Peng, D.; Zhou, J.T.; Peng, X. Contrastive Clustering. In Proceedings of the 35th AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; Volume 35, pp. 8547–8555.
18. Debnath, T.; Reza, M.M.; Rahman, A.; Beheshti, A.; Band, S.S.; Alinejad-Rokny, H. Four-layer ConvNet to facial emotion recognition with minimal epochs and the significance of data diversity. *Sci. Rep.* **2022**, *12*, 6991. [CrossRef]
19. Stefanovič, P.; Kurasova, O. Approach for Multi-Label Text Data Class Verification and Adjustment Based on Self-Organizing Map and Latent Semantic Analysis. *Informatica* **2022**, *33*, 109–130. [CrossRef]
20. Stefanovič, P.; Kurasova, O.; Štrimaitis, R. The N-Grams Based Text Similarity Detection Approach Using Self-Organizing Maps and Similarity Measures. *Appl. Sci.* **2019**, *9*, 1870. [CrossRef]
21. Dai, A.M.; Le, Q.V. Semi-supervised sequence learning. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 3079–3087.
22. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. In Proceedings of the NAACL-HLT 2018, New Orleans, LA, USA, 1–6 June 2018; pp. 2227–2237.
23. Uto, M.; Xie, Y.; Ueno, M. Neural Automated Essay Scoring Incorporating Handcrafted Features. In Proceedings of the 28th International Conference on Computational Linguistics; International Committee on Computational Linguistics, Online, 8–13 December 2020; pp. 6077–6088.
24. Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; Krishnan, D. Supervised Contrastive Learning. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–12 December 2020; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 18661–18673.
25. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP; Association for Computational Linguistics, Brussels, Belgium, 1 November 2018; pp. 353–355.
26. Pejić Bach, M.; Krstić, Ž.; Seljan, S.; Turulja, L. Text Mining for Big Data Analysis in Financial Sector: A Literature Review. *Sustainability* **2019**, *11*, 1277. [CrossRef]
27. Nguyen, B.V.; Nguyen, V.H.; Ho, T.P. Sentiment Analysis of Customer Feedback in Online Food Ordering Services. *Bus. Syst. Res. J.* **2021**, *12*, 46–59. [CrossRef]
28. Isada, F. An Empirical Study on Inter-Organisational Network Structures for Connected Cars. In Proceedings of the ENTRENOVA-ENTerprise REsearch InNOVAtion Conference, Hybrid Conference, Zagreb, Croatia, 9–10 September 2021; IRENET–Society for Advancing Innovation and Research in Economy: Zagreb, Croatia, 2021; Volume 7, pp. 324–334.
29. Asgari, T.; Daneshvar, A.; Chobar, A.P.; Ebrahimi, M.; Abrahamyan, S. Identifying key success factors for startups with sentiment analysis using text data mining. *Int. J. Eng. Bus. Manag.* **2022**, *14*, 18479790221131612. [CrossRef]
30. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
31. Sundermeyer, M.; Schlüter, R.; Ney, H. LSTM neural networks for language modeling. In Proceedings of the Thirteenth Annual Conference of the International Speech Communication Association, Portland, OR, USA, 9–13 September 2012; pp. 194–197.
32. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.

33. Dozat, T.; Manning, C.D. Deep biaffine attention for neural dependency parsing. In Proceedings of the 5th International Conference on Learning Representations, ICLR, Toulon, France, 24–26 April 2017; pp. 1–8.
34. Tan, M.; dos Santos, C.; Xiang, B.; Zhou, B. LSTM-based deep learning models for non-factoid answer selection. *arXiv* **2015**, arXiv:1511.04108.
35. Yuan, X.; Wang, S.; Wan, L.; Zhang, C. SSF: Sentence similar function based on Word2vector similar elements. *J. Inf. Process. Syst.* **2019**, *15*, 1503–1516.
36. Miller, D. Leveraging BERT for extractive text summarization on lectures. *arXiv* **2019**, arXiv:1906.04165.
37. Ghodratnama, S.; Beheshti, A.; Zakershahrak, M.; Sobhanmanesh, F. Intelligent narrative summaries: From indicative to informative summarization. *Big Data Res.* **2021**, *26*, 100257. [CrossRef]
38. Ghodratnama, S.; Zakershahrak, M.; Beheshti, A. Summary2vec: Learning semantic representation of summaries for healthcare analytics. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–8.
39. Reimers, N.; Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv* **2019**, arXiv:1908.10084.
40. Wang, S.; Hu, L.; Wang, Y.; He, X.; Sheng, Q.Z.; Orgun, M.A.; Cao, L.; Ricci, F.; Yu, P.S. Graph learning based recommender systems: A review. In Proceedings of the 30th International Joint Conference on Artificial Intelligence, Montreal, QC, Canada, 19–27 August 2021; AAAI Press: Palo Alto, CA, USA, 2021; pp. 1–9.
41. Tian, F.; Gao, B.; Cui, Q.; Chen, E.; Liu, T.Y. Learning deep representations for graph clustering. In Proceedings of the AAAI Conference on Artificial Intelligence, Montreal, QC, Canada, 27–31 July 2014; Volume 28, pp. 1293–1299.
42. Kohonen, T. The self-organizing map. *Proc. IEEE* **1990**, *78*, 1464–1480. [CrossRef]
43. Kohonen, T. Essentials of the self-organizing map. *Neural Netw.* **2013**, *37*, 52–65. [CrossRef]
44. Yang, J.; Tang, Y.; Beheshti, A. Design methodology for service-based data product sharing and trading. In *Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future: Essays Dedicated to Michael Papazoglou on the Occasion of His 65th Birthday and His Retirement*; Springer: Cham, Switzerland, 2021; pp. 221–235.
45. Ni, M.; Cheng, H.; Lai, J. GAN–SOM: A clustering framework with SOM-similar network based on deep learning. *J. Supercomput.* **2021**, *77*, 4871–4886. [CrossRef]
46. Sadowski, Ł.; Nikoo, M.; Nikoo, M. Principal component analysis combined with a self organization feature map to determine the pull-off adhesion between concrete layers. *Constr. Build. Mater.* **2015**, *78*, 386–396. [CrossRef]
47. Stefanovič, P.; Kurasova, O. Creation of Text Document Matrices and Visualization by Self-Organizing Map. *Inf. Technol. Control.* **2014**, *43*, 37–46. [CrossRef]
48. Jaiswal, A.; Babu, A.R.; Zadeh, M.Z.; Banerjee, D.; Makedon, F. A survey on contrastive self-supervised learning. *Technologies* **2021**, *9*, 2. [CrossRef]
49. Fang, H.; Xie, P. Cert: Contrastive self-supervised learning for language understanding. *arXiv* **2020**, arXiv:2005.12766.
50. Dasgupta, T.; Naskar, A.; Dey, L.; Saha, R. Augmenting textual qualitative features in deep convolution recurrent neural network for automatic essay scoring. In Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications, Melbourne, Australia, 19 July 2018; pp. 93–102.
51. Sung, C.; Dhamecha, T.I.; Mukhi, N. Improving short answer grading using transformer-based pre-training. In Proceedings of the International Conference on Artificial Intelligence in Education, Chicago, IL, USA, 25–29 June 2019; Springer: Cham, Switzerland, 2019; pp. 469–481.
52. Taghipour, K.; Ng, H.T. A neural approach to automated essay scoring. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 1882–1891.
53. Uto, M.; Okano, M. Robust Neural Automated Essay Scoring Using Item Response Theory. In Proceedings of the Artificial Intelligence in Education, Ifrane, Morocco, 6–10 July 2020; Bittencourt, I.I., Cukurova, M., Muldner, K., Luckin, R., Millán, E., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 549–561.
54. Wang, Y.; Wei, Z.; Zhou, Y.; Huang, X.J. Automatic essay scoring incorporating rating schema via reinforcement learning. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 791–797.
55. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers); Association for Computational Linguistics, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
56. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
57. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
58. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
59. Zhang, X.; LeCun, Y. Text understanding from scratch. *arXiv* **2015**, arXiv:1502.01710.
60. Xu, J.; Xu, B.; Wang, P.; Zheng, S.; Tian, G.; Zhao, J.; Xu, B. Self-Taught convolutional neural networks for short text clustering. *Neural Netw.* **2017**, *88*, 22–31. [CrossRef]

61. Zhang, D.; Nan, F.; Wei, X.; Li, S.; Zhu, H.; McKeown, K.; Nallapati, R.; Arnold, A.; Xiang, B. Supporting Clustering with Contrastive Learning. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; Association for Computational Linguistics, Online, 10–15 July 2021; pp. 5419–5430.
62. Schiliro, F.; Moustafa, N.; Beheshti, A. Cognitive privacy: AI-enabled privacy using EEG signals in the internet of things. In Proceedings of the 2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys), Nadi, Fiji, 14–16 December 2020; pp. 73–79.