ADCluster: Adaptive Deep Clustering for Unsupervised Learning from Unlabeled Documents

Arezoo Hatefi
Umeå University
Sweden
arezooh@cs.umu.se

Xuan-Son Vu
Umeå University
Sweden
sonvx@cs.umu.se

Monowar Bhuyan
Umeå University
Sweden
monowar@cs.umu.se

Frank Drewes
Umeå University
Sweden
drewes@cs.umu.se

Abstract

We introduce ADCluster, a deep document clustering approach based on language models that is trained to adapt to the clustering task. This adaptability is achieved through an iterative process where K-Means clustering is applied to the dataset, followed by iteratively training a deep classifier with generated pseudo-labels – an approach referred to as *inner adaptation*. The model is also able to adapt to changes in the data as new documents are added to the document collection. The latter type of adaptation, *outer adaptation*, is obtained by resuming the inner adaptation when a new chunk of documents has arrived. We explore two outer adaptation strategies, namely accumulative adaptation (training is resumed on the accumulated set of all documents) and non-accumulative adaptation (training is resumed using only the new chunk of data). We show that ADCluster outperforms established document clustering techniques on medium and long-text documents by a large margin. Additionally, our approach outperforms well-established baseline methods under both the accumulative and non-accumulative outer adaptation scenarios.

1 Introduction

Document clustering is the task of arranging large volumes of unlabeled documents into clusters according to some notion of similarity. A particularly common goal is to discover the most common topics in a given collection of text documents and to assign each document to its corresponding cluster. Given the ever-growing number of documents available online and the fact that manually structuring them is impossible, there are countless applications of document clustering techniques.

General purpose clustering algorithms not specifically designed to work on text documents can be used for document clustering by creating vector representations of documents using deep neural networks and then clustering those vectors. One way

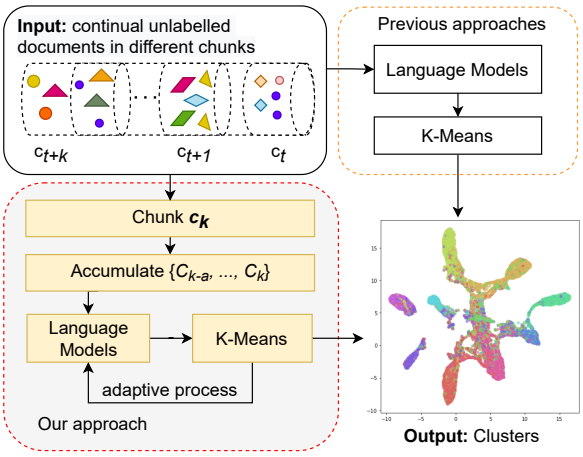


Figure 1: Overview of traditional approaches in comparison to ours in unsupervised text clustering tasks, where chunk data can be accumulated for the adaptive process.

of doing so is to use autoencoders (Ballard, 1987; Schmidhuber, 2015) applied to *term frequency – inverse document frequency* document representations (tf-idf, (Rajaraman and Ullman, 2011)). However, such representations neglect contextual information. Alternatively, one can use contextual representations obtained from pre-trained language models (LMs). Such approaches run a clustering algorithm such as K-Means over the output of the LM (Guan et al., 2022; Subakti et al., 2022; Grootendorst, 2022; Zhang et al., 2022; Eklund and Forsman, 2022). In another line of work, some studies proposed the simultaneous learning of document representations and clustering through a self-learning approach. This involves computing an auxiliary target distribution using the output of the model and minimizing the loss between these distributions (Huang et al., 2020; Xie et al., 2016; Hadifar et al., 2019). A problem with this approach is the risk of self-confirmation bias, potentially leading to trivial solutions. Moreover, the majority of these proposals rely on autoencoders, with limited exploration of LMs. In this paper, we introduce ADCluster, which uses K-Means as a teacher to

train an LM-based classifier in an iterative manner to adapt it to the clustering task. Figure 1 shows the comparison between our approach and previous approaches (which use LMs) in the unsupervised clustering task. We hypothesize that the adaptation process is essential for any real-world application where there is no labeled training data.

In applications that rely on document clustering, the collection of documents is seldom static. For example, consider an online service using web crawlers to find new content of interest for them, or an online advertising service trying to discover appropriate web pages for ad placement (Hatefi et al., 2021). Given that new content is created every day, their document collections will steadily increase. With time, clustering will become unreliable because of subtle topic shifts or previously unknown terms such as Fridays for Future or King Charles III. Our method facilitates resuming the iterative adaptation of the model to the clustering task from its previous state when a new chunk of documents is to be incorporated.

Thus, we distinguish between inner and outer adaptation. Inner adaptation adjusts the LM to the clustering task at hand by an iterative training process during which the data is considered immutable. Outer adaptation adjusts the model over time to growing sets of documents by resuming the inner adaptation when a significant amount of new data becomes available, either by considering the entire dataset (*accumulative outer adaptation*) or using only the new data (*non-accumulative outer adaptation*). An obvious third possibility is to rebuild the model from scratch or use a scheduled combination of the three possibilities, depending on the practical conditions under which the model is used.

In this paper, we mainly focus on introducing the model and studying its performance under the accumulative and non-accumulative adaptation regimes. Future work will study the dynamic behavior arising when the model adapts to growing document collections as topics evolve.

Apart from introducing the clustering technique itself, and the algorithm used for training, we experiment with three different datasets, each of which we divide into five chunks in order to simulate growing collections of documents. The empirical results show the following:

1. Under each variant of the outer adaptation (training from scratch, accumulative, and non-accumulative adaptation), ADCluster outperforms the baselines.

2. In the absence of significant topic shifts, the three outer adaptation regimes usually result in comparable performance. Hence, one can choose between them as fits the application.

In addition to these main results, we conduct experiments to show that the method is insensitive to the type of language model used (our main experiments use BERT).

2 Related Work

Clustering is a much studied unsupervised problem in machine learning and data mining which is central to many data-driven applications. Many strategies for clustering arbitary sets of data points in an n-dimensional space have been studied. These include density-based, hierarchical, centroid- and partition-based clustering; see Xu and Tian (2015) for an overview. K-Means (MacQueen et al., 1967) and HDBSCAN (Campello et al., 2013) are two of the most popular traditional clustering algorithms.

The progress in deep learning that has been made during the last decade has made it natural to apply deep learning to clustering tasks (Zhou et al., 2022). An example of this is seen in DEC (Xie et al., 2016), which utilizes a stacked autoencoder to acquire document representations from tf-idf vectors. Subsequently, it improves these representations while learning clustering in a self-supervising manner. Hosseini and Varzaneh (2022) present a hybrid deep clustering method combining a stacked autoencoder and k-Means to organize Persian texts into clusters.

In recent years, large language models trained for language understanding and generation have achieved impressive results across a wide range of tasks. These LMs produce excellent general-purpose contextual representations that reflect topical information and can thus be used for clustering. Guan et al. (2022) generate document representations by pooling the outputs of ELMo (Peters et al., 2018) pre-trained LM and apply K-Means to these representations after normalizing them. Gupta et al. (2022) employ language models for unsupervised model interpretation and syntax induction through deep clustering of text representations. Huang et al. (2020) fine-tune the LM simultaneously with masked language modeling and clustering losses.

To our knowledge, no existing research explores deep clustering with LMs for dynamic scenarios

involving a growing set of documents. Our method provides a simple yet effective approach to improve cluster assignments by training the LM in an adaptive manner to provide clustering-friendly representations that, over time, can be adapted to a growing set of documents.

3 Methodology

We first describe how the *inner* adaptation of the proposed model ADCluster works. Its pseudocode is given in Algorithm 1. It uses a conventional K-Means algorithm and a Deep Neural Network (DNN) classifier. The classifier is adapted iteratively in order to improve the clusterability of the embedding vectors. This is the inner adaptation. The classifier consists of a LM-based text encoder (a pre-trained LM with a mean pooling layer over its last layer) denoted by f_θ (where θ is the set of parameters) followed by a Multi-Layer Perceptron (MLP) head denoted by W that maps document representations to cluster assignments. Suppose we have an unlabeled dataset $D = \{d_n\}_{n=1}^N$ of N documents. At the beginning of each training epoch, we map each document d_n to its contextual representation $f_\theta(d_n)$. So, $E = \{f_\theta(d_n)\}_{n=1}^N$ is the set of document contextual representations. Often, it is beneficial to reduce the dimensionality of these representations using a dimension reduction method such as PCA (Pearson, 1901) or UMAP (McInnes et al., 2020), resulting in a set E' of vectors of fewer dimensions. Next, we use K-Means (based on cosine similarity rather than squared Euclidean distance) to cluster E' into K distinct clusters. We use these cluster assignments $\{p_n\}_{n=1}^N$ as *pseudo-labels* to train the classifier. For this, the MLP W and the encoder f_θ are jointly trained to minimize the cross entropy loss

$$\frac{\sum_{n=1}^{b} -\log \frac{\exp\left(y_{n,p_n}\right)}{\sum_{k=1}^{K} \exp\left(y_{n,k}\right)}}{b} \quad (1)$$

where y_n is the output of the classifier for document d_n and b is the mini-batch size. This cost function is minimized using AdamW (Loshchilov and Hutter, 2019) and backpropagation to compute the gradients. With the goal of preventing the classifier from overfitting to the current pseudo-labels, we employ only a subset of the data in every training epoch and restrict the number of iterations (i.e., *EpochSize* in Algorithm 1).

It is worth mentioning that there is no correspondence between two consecutive cluster assignments. Hence, the final classification layer learned for an assignment becomes irrelevant for the following one and thus needs to be re-initialized from scratch at each epoch. We found that re-initializing the entire MLP head of the classifier rather than the final classifier layer is also beneficial for reducing the risk of overfitting. Since the MLP is a shallow network (having only one hidden layer), it can be trained sufficiently in one epoch.

In addition, we predict cluster assignments for all documents at the end of each epoch using the classifier and stop our procedure when the change in assignments is less than a threshold τ, i.e., the algorithm terminates when the number of documents for which the cluster assignment changes falls below τ.

Algorithm 1: ADCluster (inner adaptation)

Input : D: the set of unlabeled documents
f_θ: LM-based encoder of DNN classifier
W: MLP head of DNN classifier
$MaxIter$: the max training iterations
$EpochSize$: iterations per training epoch
b: the mini-batch size
η, γ: the training learning rates
DR: the dimension reduction method
τ: a threshold for the minimum percentage of changing assignments within two consecutive epochs (convergence threshold)

Output : (θ^*, W^*): The optimal weights
C: final cluster assignments for D

$MaxEpoch \leftarrow MaxIter / EpochSize$;
for $epoch = 1$ **to** $MaxEpoch$ **do**
  $E \leftarrow$ encode D with f_θ;
  $E' \leftarrow DR(E)$ ▷ Apply DR with condition
  $P \leftarrow$ run K-means on E' using cosine similarity;
  $X \leftarrow$ choose $b * EpochSize$ documents from pseudo-labeled set P with a uniform sampler;
  $W \leftarrow$ initialize W with Xavier initialization;
  for $iter = 1$ **to** $EpochSize$ **do**
    $B_{iter} \leftarrow$ choose a mini-batch from X;
    $Y_{iter} \leftarrow W(f_\theta(B_{iter}))$;
    $\hat{Y}_{\text{K-means}} \leftarrow P(B_{iter})$;
    $l \leftarrow$ cross-entropy-loss $(Y_{iter}, \hat{Y}_{\text{K-means}})$;
    $\theta \leftarrow \theta - \eta * l(\theta)$ ▷ Update θ
    $W \leftarrow W - \gamma * l(W)$ ▷ Update W
  end
  $C_{curr} \leftarrow W_{predict}(f_\theta(D))$ ▷ predict cluster assignments for D with DNN classifier
  $t \leftarrow$ compute (C_{curr}, C_{prev}) ▷ Compute the percentage of changing cluster assignments compared to previous epoch;
  if $t < \tau$ **then**
    stop the iterative process
  end
  $C_{prev} \leftarrow C_{curr}$
end
return θ^*,W^*,C;

Table 1: Datasets and statistics. *Silhouette Coefficient* refers to the Silhouette score of Rousseeuw (1987) which measures how similar a document is to its own cluster compared to other clusters, the best and worst values being 1 and -1, respectively. We compute the mean Silhouette Coefficient of all samples of the datasets using their true labels. As our LM for creating document representations, we use a BERT language model.

Dataset	Yahoo!5	Ag News	Fake News
#-Documents	38 812	40 000	480
Avg # sents	25.12	1.45	6.05
Avg # word (in doc)	578.26	36.09	141.20
Avg Silhouette Coefficient	0.01234	0.03736	0.04356

Overall, ADCluster alternates between clustering document representations to produce pseudo-labels and updating the parameters of the classifier by predicting these pseudo-labels using Eq. (1). This iterative adaptation of the encoder teaches the LM to generate more clustering-friendly representations. This distinguishes ADCluster from conventional methods, resulting in an improved K-Means clustering in subsequent epochs. The final clusters are obtained using the adapted classifier to predict cluster assignments.

If K-Means assigns almost all documents to a few large clusters, θ will only discriminate between them. A trivial parameterization occurs when all clusters except one are singletons, and therefore the classifier predicts the same output for all inputs (Caron et al., 2018). To overcome this problem, we train the classifier on uniformly sampled documents from the pseudo-labeled classes. The result is the same as weighting the contribution of a document to the loss function by the inverse of the size of the cluster to which it belongs.

Let us now briefly explain the *outer* adaptation of ADCluster. Imagine a data stream where new data arrives sequentially in chunks C_t, where t denotes the time step. In the *accumulative* scenario, we resume the inner adaptation of ADCluster at time t using $C_0 \cup \cdots \cup C_t$ as training data when a new chunk C_t arrives. In contrast, the *non-accumulative* approach resumes inner adaptation solely with the latest chunk C_t.

4 Experiments

4.1 Datasets

We employ the following three datasets whose statistics are summarized in Table 1:

Yahoo!5 is a subset of Yahoo! Answers (Zhang et al., 2015). The dataset comprises 10 classes, each document consisting of a question, a title, and the best answer to the question. We obtain the text to be clustered by concatenating these parts. To obtain a long-text dataset we only choose samples of over 500 tokens. The resulting dataset includes 38 812 documents.

Ag News (Zhang et al., 2015) consists of 4 classes: World, Sports, Business, and Sci/Tech news. The number of training and testing samples for each class is 30 000 and 1 900, respectively. We choose 40 000 documuments at random from the training set. To have a very short-text dataset, we only consider the news text and ignore the titles.

Fake News (Pérez-Rosas et al., 2018) comprises 480 medium-length news articles belonging to six different domains. While half of the articles are real and the other half are fake news, we do not make use of this distinction but use only the six topics of the dataset as labels.

Following the approach of prior studies (Huang et al., 2020; Xie et al., 2016; Hadifar et al., 2019), we form unlabelled documents by removing all labels for the training set, using the labels only to evaluate unsupervised performance.

4.2 Baselines

We use the following baselines for comparisons:

Traditional clustering algorithms We compare our model with K-Means and HDBSCAN. For HDBSCAN, we use the soft (or fuzzy) implementation[1] of the algorithm that predicts probability vectors for all dataset samples; no samples are considered noise. These vectors show the membership probability for each cluster, so we assign the sample to the cluster for which the highest probability has been determined. Instead of using pure BERT vectors, we apply normalization on them prior to performing dimension reduction and clustering. Before running HDBSCAN on the datasets, we perform dimension reduction using UMAP[2]. For each dataset, we test several values

[1] https://hdbscan.readthedocs.io/en/latest/soft_clustering.html

[2] https://umap-learn.readthedocs.io/en/latest/

for parameters of HDBSCAN and UMAP and report the highest accuracy we get. On Yahoo! Answers, we perform PCA dimension reduction ($n\_components = 0.8$; preserving at least 80% of variance) before K-Means.

DEC-tfidf we compare our model with that of Xie et al. (2016), using the available PyTorch implementation from https://github.com/vlukiyanov/pt-dec. We slightly adjust the parameters reported in the paper to our datasets and present the highest value obtained.

DEC-BERT To have a more fair comparison between ADCluster and DEC (Xie et al., 2016), we replace the stacked autoencoder part of DEC with a BERT language model followed by a mean pooling layer to encode documents and train it with the same objective function as in DEC.

UFT We compare our model with the model presented in Huang et al. (2020). We refer to this baseline as UFT. We obtained the source code from the authors of the paper and applied it to our datasets.

ADCluster-noIter is a non-iterative version of ADCluster. We run K-Means only once using contextual representations of documents from BERT and train the neural classifier with the generated pseudo-labels for some iterations.

Centroid-ADCluster Since in ADCluster there is no correspondence between two consecutive cluster assignments, the final classification layer learned for an assignment becomes irrelevant for the following one and thus needs to be reinitialized from scratch at each epoch. We do this to prevent the model from overfitting to the noisy pseudo-labels. For verification, we implemented another version of ADCluster in which we, instead of learning a classification layer predicting the cluster assignments, perform explicit comparisons between features and centroids.

4.3 Evaluation Metric

We adopt a standard unsupervised evaluation metric that is widely used in deep clustering studies to compare our proposed method to other algorithms. For all the algorithms, the number of clusters is set to the number of ground-truth categories of each dataset, and we evaluate the clustering performance using the unsupervised clustering accuracy (ACC):

$$ACC = \max_{m} \frac{\sum_{n=1}^{N} 1\{l_n = m(c_n)\}}{N}$$

where N is the total number of documents, l_n is the ground-truth label of document d_n, c_n is the cluster assignment that is predicted by the clustering algorithm for d_n, and m maps cluster assignments to labels, ranging over all possible one-to-one mappings. This metric seeks the best possible alignment between the ground-truth label and the cluster assignments generated by an unsupervised clustering algorithm. The Hungarian algorithm, presented in the work of Xu et al. (2003), offers a means to efficiently calculate the most effective mapping function within the context of a linear assignment problem.

4.4 Experimental Setup

We implemented ADCluster using the PyTorch framework, utilizing bert-base-uncased LM of Hugging Face[3]. Documents are truncated to their first 256 tokens. To generate document embeddings, we employ average pooling over the output of the language model. For label prediction, we employ a two-layer MLP with a single hidden layer. The hidden layer size is set to 128 for Yahoo!5 and Fake News and 768 for Ag News. The hyperbolic tangent function is used as the activation function for the MLP.

We set the mini-batch size to 4 and the learning rate of the LM and MLP head to 10^{-6} and 10^{-4} correspondingly. We also use a cosine scheduler for the learning rate of the LM. We train ADCluster for at most 10 000 iterations and reassign the clustering labels by applying K-Means on document representations every 200 iteration (which we call an *epoch*). The threshold for stopping training when cluster assignments do not significantly change anymore is set to 1% of the documents. The model is trained using the AdamW optimizer with α and β equal to 0.999. We use the first 200 iterations as warm-up steps for the LM. To initialize the centroids of K-Means we use the K-Means++ seeding strategy proposed by Arthur and Vassilvitskii (2007) and to initialize weights of MLP head in each epoch we use Xavier initialization (Glorot and Bengio, 2010). We train ADCluster-noIter and Centroid-ADCluster under the same settings. The only difference for Centroid-ADCluster is that the size of the hidden layer of the MLP head is 768 for all datasets and the weights of the last layer ($768 \cdot K$, where K is the number of classes in the dataset) are initialized with the centroids of the K-

[3] https://huggingface.co/bert-base-uncased

Means which are constant during training. For the other baselines, we test several sets of values for their hyperparameters and report the best results.

5 Results and Discussions

5.1 Overall Performance

Generally, ADCluster achieves better performances than most of the baseline methods across multiple datasets (see Table 2). Compared to traditional clustering algorithms, ADCluster outperforms K-Means from 1.84% (Ag News) up to 23.3% (Yahoo!5), indicating that the iterative learning process (inner adaptation) of our model is effective. We can also note that HDBSCAN achieves better performance than K-Means in most cases but outperforms ADCluster only in the case of Ag News. In Table 1, we see that Ag News consists of very short texts, its average number of sentences per document being 1.45 and the average number of words being 36.09. It does not seem to provide enough context for BERT to make distinctive representations, thus limiting the efficacy of our model on this particular dataset. However, in Section 5.5 we will see that by replacing BERT with more advanced LMs the performance of our model on this dataset improves. For Yahoo!5 and Fake News, HDBSCAN gains better performance than most of the other methods except ADCluster. In fact, for these datasets, ADCluster displays better performance than all baselines. This holds even in the case of Fake News, which consists of a very limited number of documents (i.e., 480 documents).

The comparison with DEC-based models yields the following observations. Firstly, ADCluster outperforms DEC-tfidf, which we attribute to its use of BERT contextual representations (whereas tf-idf representations only consider text as a bags of words and neglect their semantic relations). Secondly, even though DEC-BERT has similar access to the contextual information of the language model, its performance is still lower than that of our model. The same applies to the UFT baseline. The reason could be that these models are trained in a self-learning fashion and may thus suffer from self-confirmation. Our model avoids this by using K-Means as an external teacher for our neural classifier. It also uses a uniform sampling technique for batch creation, mitigating biases stemming from imbalanced clusters.

5.2 Dynamic Performance Analysis of ADCluster Across Varied Dataset Sizes

In this experiment, we examine the performance of ADCluster in comparison to baselines as the dataset size gradually increases. The outcomes of this experiment are presented in Table 3, illustrating the results as the document size expands from 10% to 100%. In general, ADCluster consistently maintains stable performance throughout these experiments and surpasses baseline models for all datasets, with the exception of the 10% case for Fake News.

5.3 Illustration of Learned Representations by ADCluster

In order to investigate how ADCluster develops clustering-friendly representations through internal adaptation, we visualize the evolution of clusters during the training process using the Yahoo!5 dataset. Figure 2 shows how ADCluster clusters the documents during different epochs with ground-truth classes represented by different colors. The figure clearly demonstrates that at the very beginning, the structure is random. Along with the adaptation process, documents are arranged into more distinct groups, which is signified by both color separation and spatial characteristics. This trend is further confirmed by the continuous enhancement in clustering performance observed in each successive epoch.

5.4 The Model Behavior on Data Streams

Notation. Hereafter, if not otherwise specified, we use *Ac* to abbreviate *Accumulation*. We randomly split each unlabelled data collection into 5 chunks and denote them by C_1 (1–20%), C_2 (21–40%), C_3 (41–60%), C_4 (61–80%), C_5 (81–100%).

We now analyze the outer adaptation behavior of ADCluster. In this experiment, we assume the number of the clusters to be constant over time, only receiving new samples. We compare our model with three baselines:

Word2vec+KM We generate document representations as the average of the Word2vec embeddings of all words in the document and use K-Means to cluster these representations.

BERT+KM We create document representations by taking the average of the output of the last BERT layer for non-pad tokens and use K-Means to cluster these representations.

ADCluster-scratch This baseline is the same

Table 2: Overall performances of ADCluster in comparison to baselines. ♥ indicates short-text datasets.

Method		Yahoo!5	Ag News♥	Fake News
Classic Clustering	Kmeans (BERT)	44.64	81.6	73.96
	HDBSCAN (BERT)	58.8	**83.68**	72.71
DEC (Xie et al., 2016)*	tf-idf	50.23	68.93	45.41
	BERT	46.43	78.32	75.83
UFT (Huang et al., 2020)*		46.94	65.46	66.67
ADCluster (ours)	Centroid-ADCluster	60.64	80.93	76.67
	ADCluster-Final	**67.94**	83.44	**77.50**

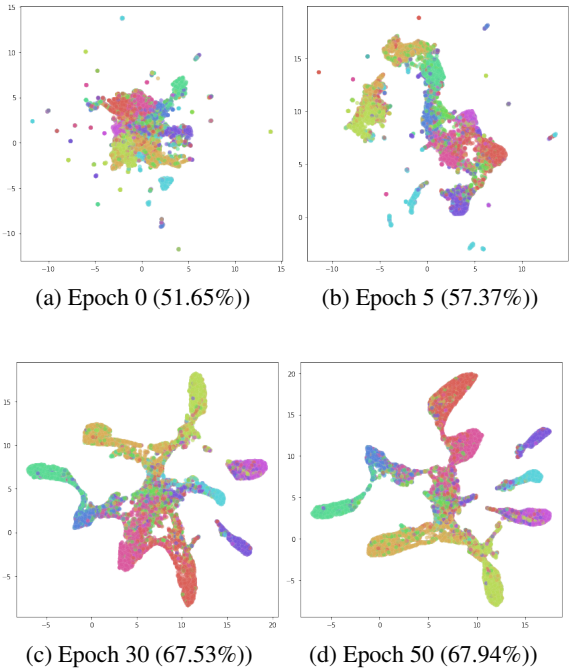
* The result is produced by us following the original paper

Table 3: Performance analysis of ADCluster across varied dataset sizes compared to baselines. Note that, because of the unsupervised setting, there is no expectation of monotonic increases in performance.

Dataset	Method	10%	50%	80%	100%
Ag News	K-Means	82.4	81.39	81.41	81.6
	DEC-BERT	79.3	78.22	78.4	78.32
	ADCluster	**84.08**	**82.56**	**84.3**	**83.44**
Yahoo!5	K-Means	53.23	53.5	59.95	52.17
	DEC-BERT	45.74	46.44	46.56	46.43
	ADCluster	**66.3**	**66.03**	**67.38**	**67.94**
Fake News	K-Means	64.58	77.08	77.34	73.96
	DEC-BERT	**68.75**	79.58	77.60	75.83
	ADCluster	64.58	**83.75**	**79.95**	**77.50**

as ADCluster except that instead of performing outer adaptation, we train the model from scratch (accumulatively on the whole dataset or non-accumulatively on the last chunk only, respectively). Thus, we remove the outer adaptation and the model only benefits from the inner adaptation. Tables 4–6 show the results of our experiments.

As our main take-aways from these experiments, we note that ADCluster outperforms the Word2vec+KM and BERT+KM baselines in all cases in both the *Ac* and *non-Ac* settings. The superior accuracy of ADCluster on chunk C_1 can



(a) Epoch 0 (51.65%)) (b) Epoch 5 (57.37%))

(c) Epoch 30 (67.53%)) (d) Epoch 50 (67.94%))

Figure 2: Illustration of clustered contextual representations according to ADCluster for Yahoo! Answer during inner adaptation. Colors indicate ground-truth classes. We have used UMAP to map 768-dimensional representations to a 2D feature space for illustration.

Table 4: Comparing the outer adaptation performance of ADCluster with baselines on Yahoo!5.

Method	Ac	C_1	C_2	C_3	C_4	C_5
Word2vec+KM	Yes	52.09	41.86	47.08	44.94	49.02
BERT+KM	Yes	46.28	53.84	53.67	55.24	53.70
ADCluster-scratch	Yes	**67.33**	66.44	64.06	64.51	62.06
ADCluster	Yes	**67.33**	**67.99**	**68.07**	**67.8**	**67.48**
Word2vec+KM	No	52.09	42.51	45.72	49.79	50.22
BERT+KM	No	46.28	57.02	52.00	54.86	55.04
ADCluster-scratch	No	**67.33**	67.11	65.19	61.79	65.50
ADCluster	No	**67.33**	**68.07**	**68.24**	**67.61**	**67.98**

Table 5: Comparing the outer adaptation performance of ADCluster with baselines on Ag News.

Method	Ac	C_1	C_2	C_3	C_4	C_5
Word2vec+KM	Yes	80.65	79.98	80.55	80.87	80.83
BERT+KM	Yes	81.66	81.42	81.50	81.51	81.52
ADCluster-scratch	Yes	**84.07**	84.56	**84.09**	**83.07**	81.76
ADCluster	Yes	**84.07**	**84.81**	82.56	83.05	**84.03**
Word2vec+KM	No	80.65	79.59	81.49	80.80	80.85
BERT+KM	No	81.66	81.43	81.20	81.82	81.05
ADCluster-scratch	No	**84.07**	83.74	81.95	**83.87**	82.51
ADCluster	No	**84.07**	**84.01**	**84.25**	83.6	**83.44**

Table 6: Comparing the outer adaptation performance of ADCluster with baselines on Fake News.

Method	Ac	C_1	C_2	C_3	C_4	C_5
Word2vec+KM	Yes	67.71	79.69	78.47	71.35	74.58
BERT+KM	Yes	57.29	77.60	77.08	77.34	77.29
ADCluster-scratch	Yes	**69.79**	82.81	**84.37**	79.69	79.58
ADCluster	Yes	**69.79**	**83.33**	83.68	**81.25**	**80.62**
Word2vec+KM	No	67.71	80.21	62.50	54.17	57.29
BERT+KM	No	57.29	77.08	58.33	53.12	51.04
ADCluster-scratch	No	**69.79**	82.29	67.71	57.29	59.37
ADCluster	No	**69.79**	**86.46**	**79.17**	**61.46**	**73.96**

be attributed to the inner adaptation which the baseline models lack. However, interestingly the outer adaptation results in superior performances in most cases on chunks C_2–C_5 even compared to ADCluster-scratch, which is remarkable and shows the effectiveness of outer adaptation.

5.5 Ablation study

In this ablation study, we design two settings to study the effectiveness of each ADCluster component. First, we replace the default BERT language model with recent models such as RoBERTa, SBERT, and BART. Second, we test various settings: (1) removing outer adaptation, (2) using a random sampler instead of a uniform sampler, and (3) Using UMAP for dimension reduction (instead of PCA for the Yahoo!5, and instead of not using dimension reduction for Ag News and Fake News). Figure 3 clearly shows that recent advanced language models yield better performance on all of the datasets. Table 7 summarizes the performance of ADCluster in the second setting. Across all experiments, the final model of ADCluster shows better performance than these variants.

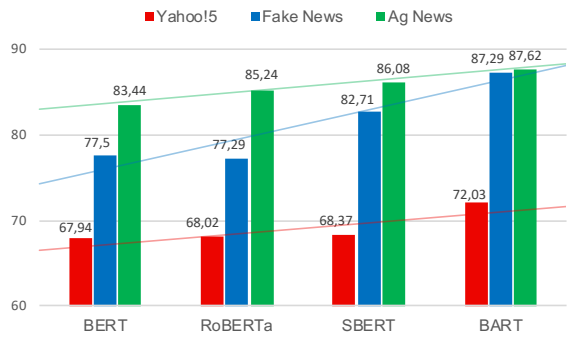


Figure 3: Ablation study w.r.t. different language models being used for the inner adaptation of ADCluster.

Table 7: Ablation study to evaluate the impact of different components of ADCluster to the final performance.

Ablation setting	Yahoo!5	Ag News	Fake News
Non iterative	53.89	82.88	73.96
UMAP	64.74	58.33	66.25
Random sampler	65.78	79.2	76.04

6 Conclusion and Future Work

We have introduced ADCluster, a neural document clustering model that iterates between a contextual language model and K-Means. K-Means is applied to contextualized document representations created by a BERT language model in order to obtain pseudo-labels. The weights of the language model are then iteratively adapted to improve the prediction of cluster assignments using discriminative loss. Not only does this *inner adaptation* result in superior clustering performance, it also enables us to resume training when the dataset grows (outer adaptation), as is often the case in real-world applications. Our empirical results show that for medium to long-text documents, ADCluster consistently outperforms conventional clustering models by a considerable margin with respect to the unsupervised accuracy measure.

Future work will have to study the inner and outer adaptation in more detail. For instance, one interesting direction could be a "soft adaptation", which continuously measures how much weight the outer adaptation shall place on earlier and later chunks. So far, we only presented two extreme cases, i.e., accumulation or non-accumulation.

Moreover, text data is often accompanied by additional modalities such as images, audio, and video. Such multimodal data has the potential to help the model understand the semantics of documents and assign them to the right cluster (Chen et al., 2021; Jiang et al., 2019). Multimodality can also open the door to new real-world downstream applications. Therefore, we are interested in extending our model to multimodal data clustering in the future.

References

David Arthur and Sergei Vassilvitskii. 2007. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, USA. Society for Industrial and Applied Mathematics.

Dana H. Ballard. 1987. Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, volume 1 of *AAAI'87*, page 279284. AAAI Press.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer Berlin Heidelberg.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149.

Brian Chen, Andrew Rouditchenko, Kevin Duarte, Hilde Kuehne, Samuel Thomas, Angie Boggust, Rameswar Panda, Brian Kingsbury, Rogerio Feris, David Harwath, et al. 2021. Multimodal clustering networks for self-supervised learning from unlabeled videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8012–8021.

Anton Eklund and Mona Forsman. 2022. Topic modeling by clustering language model embeddings: Human validation on an industry dataset. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 635–643, Abu Dhabi, UAE. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *Computing Research Repository*, arXiv:2203.05794.

Renchu Guan, Hao Zhang, Yanchun Liang, Fausto Giunchiglia, Lan Huang, and Xiaoyue Feng. 2022. Deep feature-based text clustering and its explanation. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3669–3680.

Vikram Gupta, Haoyue Shi, Kevin Gimpel, and Mrinmaya Sachan. 2022. Deep clustering of text representations for supervision-free probing of syntax. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10720–10728.

Amir Hadifar, Lucas Sterckx, Thomas Demeester, and Chris Develder. 2019. A self-training approach for short text clustering. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 194–199, Florence, Italy. Association for Computational Linguistics.

Arezoo Hatefi, Xuan-Son Vu, Monowar Bhuyan, and Frank Drewes. 2021. Cformer: Semi-supervised text clustering based on pseudo labeling. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3078–3082, Virtual Event, Queensland, Australia. Association for Computing Machinery.

Soodeh Hosseini and Zahra Asghari Varzaneh. 2022. Deep text clustering using stacked autoencoder. *Multimedia Tools Appl.*, 81(8):10861–10881.

Shaohan Huang, Furu Wei, Lei Cui, Xingxing Zhang, and Ming Zhou. 2020. Unsupervised fine-tuning for text clustering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5530–5534, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yangbangyan Jiang, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2019. Dm2c: Deep mixed-modal clustering. In *Neural Information Processing Systems*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297.

Leland McInnes, John Healy, and James Melville. 2020. Umap: Uniform manifold approximation and projection for dimension reduction. *Computing Research Repository*, arXiv:1802.03426. Version 3.

Karl Pearson. 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 2227–2237. Association for Computational Linguistics.

Anand Rajaraman and Jeffrey David Ullman. 2011. *Data Mining*, pages 1–17. Cambridge University Press.

Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Alvin Subakti, Hendri Murfi, and Nora Hariadi. 2022. The performance of BERT as data representation of text clustering. *Journal of Big Data*.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, volume 48, pages 478–487, New York, NY, USA.

Dongkuan Xu and Yingjie Tian. 2015. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2:165–193.

Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zihan Zhang, Meng Fang, Ling Chen, and Mohammad Reza Namazi Rad. 2022. Is neural topic modelling better than clustering? an empirical study on clustering with contextual embeddings for topics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3886–3893, Seattle, United States. Association for Computational Linguistics.

Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, Martin Ester, et al. 2022. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *Computing Research Repository*, arXiv:2206.07579.