Learn The Big Picture: Representation Learning for Clustering

Sumanta Kashyapi
Department of Computer Science
University of New Hampshire
sk1105@wildcats.unh.edu

Laura Dietz
Department of Computer Science
University of New Hampshire
dietz@cs.unh.edu

Abstract

Existing supervised models for text clustering find it difficult to directly optimize for clustering results. This is because clustering is a discrete process and it is difficult to estimate meaningful gradient of any discrete function that can drive gradient based optimization algorithms. So, existing supervised clustering algorithms indirectly optimize for some continuous function that approximates the clustering process. We propose a scalable training strategy that directly optimizes for a discrete clustering metric. We train a BERT-based embedding model using our method and evaluate it on two publicly available datasets. We show that our method outperforms another BERT-based embedding model employing Triplet loss and other unsupervised baselines. This suggests that optimizing directly for the clustering outcome indeed yields better representations suitable for clustering.

1 Introduction

Text clustering is a well-studied problem which finds its application in a wide range of tasks: organizing documents in cluster-based information retrieval (Cutting et al., 2017; Mei and Chen, 2014), representation of search results (Scaiella et al., 2012; Navigli and Crisafulli, 2010), analyzing different opinions about a subject (Tsirakis et al., 2017) among many others. Each of these applications may focus on text contents of different granularities (e.g. words, sentences, passages, articles) but all of them follow a common high-level approach to clustering: represent the documents in form of vectors and then cluster them based on vector similarities. Although clustering is typically employed in an unsupervised setting, many semi-supervised deep learning models have been proposed recently. Many of these approaches formulate this as a representation space learning problem (Yang et al., 2017) that projects initial document vectors into a latent vector space which is more suitable for the clustering task and generate clusters similar to some ground truth. However, most of these algorithms do not directly optimize for a clustering evaluation metric during training. Instead, they optimize for a different criterion that approximates the global clustering error. Semi-supervised clustering approaches (Basu et al., 2002) cast the clustering problem into binary classification by learning pairwise constraints extracted from the available training examples: *must-links* for sample pairs sharing the same cluster and *cannot-links* for different clusters. However, clustering problems with numerous small clusters produce only a few *must-links* among all possible links, leading to highly unbalanced training data. Consequently, the trained model is biased towards predicitng *cannot-links*. Learning triplet-based constraints (Dor et al., 2018) that combine a positive and a negative sample in a single triplet, mitigate such bias towards negative samples. However, the sample complexity (Bartlett, 1998) (number of samples required to cover all interactions in a dataset) grows more rapidly compared to paired samples. Also, such approximation of the original clustering problem may lead to unsatisfactory results because the optimization criterion does not always correspond with the clustering quality. These observations motivate us to hypothesize the following:

1. Instead of learning to solve some approximation of the original clustering problem, we need to directly optimize for a clustering evaluation metric in order to train a model specialized for clustering.

2. Instead of sample-pairs in case of pairwise constraints or triplets in case of Triplet-loss, we can make efficient and scalable use of the available training data by presenting all inter-

Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021), pages 141–151
Bangkok, Thailand (Online), August 6, 2021. ©2021 Association for Computational Linguistics

actions between a set of data points as a single clustering sample. This way the training approach neither suffers from unbalanced data nor from sample complexity.

To test our hypotheses, we propose an alternative training strategy that directly draws its supervision signal from an evaluation metric that measures clustering quality to train a representation model for text documents. During training, it consumes a complete clustering example of a set of data points as a single training sample in form of an interaction matrix. Due to this, we experiment with clustering datasets containing numerous small clustering examples instead of a single instance of a large clustering problem.

It is challenging to derive training signals directly from the clustering ground truth or a clustering evaluation metric because the clustering process is discrete. In other words, a function that estimates the clustering quality of a random partition of the input data is not continuous and hence non-differentiable. As most supervised algorithms rely on gradient-based optimization algorithms, it is difficult for them to orchestrate a useful training process without proper gradient. So far some continuous approximation of the clustering problem is used as discussed earlier to bypass the core optimization issue. Recently a novel gradient approximation method, *blackbox backpropagation* (Vlastelica et al., 2019) is proposed for combinatorial problems that finds solution in a discrete space. We leverage their findings by molding the clustering problem into a combinatorial problem. This allows us to derive meaningful gradients out of the clustering process and to train a representation model by directly optimizing for a clustering evaluation metric.

Our contribution: We make the following contributions through this work.

1. We develop a new training strategy for supervised clustering that directly obtains its supervision signal from optimizing a clustering metric.[1] We utilize recently proposed *blackbox backpropagation* technique to derive gradients from discrete clustering results that drives the training process.

2. We use our training strategy to train a BERT-based (Devlin et al., 2018) representation model suitable for topical clustering. To support the training mechanism, we design a loss function that effectively optimizes a clustering evaluation metric.

3. We empirically show that our method is more efficient in terms of training time and utilizing available training examples when compared to existing supervised clustering methods. The resulting representation model achieves better clustering results than other strong baseline models.

2 Related Work

Traditionally, text clustering is achieved by employing a distance-based clustering algorithm (e.g. KMeans) on vector representations of documents such as TF-IDF (Jones, 1972). Recent works focus on learning text representaions suitable for clustering (Chen, 2017; Xu et al., 2017; Hadifar et al., 2019). Alternatively, they explore different similarity metrics between the vectors that govern the clustering algorithm through pairwise binary constraints (Basu et al., 2002; Kulis et al., 2009). In this work, we focus on the former – representation learning of documents, suitable for text clustering.

Deep clustering (Min et al., 2018) is an active field of research that utilizes recent advancements of deep learning techniques to improve supervised clustering. The primary focus is to learn a suitable representation space that optimizes some clustering criterion (e.g. cluster assignment loss) along with a representation criterion (e.g. reconstruction loss) (Xie et al., 2016; Li et al., 2018; Ghasedi Dizaji et al., 2017; Jiang et al., 2016). It has also been shown that clustering criterions alone are sufficient to train such representation space (Yang et al., 2016). However, none of these approaches attempt to receive direct supervision from a clustering evaluation metric. Motivated by earlier works that learn a representation model under pairwise binary constraints, Chang et al. (2017) envisions the clustering task as a binary classification task of paired data samples and achieves state-of-the-art results on multiple image clustering datasets. Reimers and Gurevych (2019) propose Sentence-BERT which trains a BERT-based sentence embedding model by employing Triplet loss (Dor et al., 2018) that uses triples of sentences as training samples where exactly two of them are from the same section of Wikipedia. Although both

[1] The source code is available at https://github.com/nihilistsumo/Blackbox_clustering

of these approaches are supervised, each training sample only consists of a fraction of the whole clustering instance. Hence, during training, these methods mostly ignore the overall relationships between multiple data samples and how they form clusters.

The main hindrance of drawing a supervision signal directly from a clustering evaluation metric is the combinatorial nature of the clustering problem. Some research introduce differentiable building blocks for special cases of combinatorial algorithms such as satisfiability (SAT) problems (Wang et al., 2019). Wilder et al. (2019) use a differentiable variant of the K-means algorithm to approximate a harder combinatorial problem (e.g. graph optimization). Such relaxations of the original combinatorial problem may lead to sub-optimal results. Recently, Vlastelica et al. (2019) proposed a novel technique of differentiating combinatorial solvers as a blackbox without any relaxation that allows us to use an optimal combinatorial algorithm as a component of a deep representation learning model and optimize it end-to-end. We give a brief background of their approach in the following section.

Blackbox backpropagation. In their approach to optimize for a combinatorial function Vlastelica et al. (2019) formalize combinatorial solvers as a mapping function between continuous input, $w \in W \subseteq \mathbb{R}^N$ and discrete output, $\hat{y} \in Y$ as $w \mapsto \hat{y}$ such that the output $\hat{y} = \arg\min_{y \in Y} c(w, y)$ where c is the cost that the solver tries to minimize. Here W is the N-dimensional continuous input space and Y is a finite set of all possible solutions. For a linear cost function c, a continuous interpolation of the original cost function is constructed and the gradient of this interpolation is used during backpropagation. The closeness of the interpolation to the original function is controlled by a single hyperparameter, λ. In our work, we extend this approach for clustering framework to draw the supervision signals directly from the clustering results and learn our model parameters.

3 Methodology

Our text clustering method works in two steps: 1. Train a text representation model directly from example clusters of text snippets, 2. Cluster the trained embedding vectors using hierarchical agglomerative clustering (HAC). Our primary contribution lies in the training strategy of step 1 which we refer here as **C**lustering **O**ptimization as **B**lackbox (**COB**). We describe COB in the following sections.

3.1 Overall Approach

Supervised text clustering is a combinatorial problem. Let $\mathcal{P}$ be a set of N documents and Y be the set of all possible k-partitions of set $\mathcal{P}$. Also let V_ϕ be a representation model with trainable parameters ϕ. We obtain the set of representation vectors $V_\phi(\mathcal{P})$ for each of the documents in set $\mathcal{P}$ using the model, V_ϕ. Based on the Euclidean distances between representation vectors in $V_\phi(\mathcal{P})$, a clustering algorithm chooses a particular k-partition $\hat{y} \in Y$ that minimizes some linear cost function $c(V_\phi(\mathcal{P}), y)$ e.g. intra-cluster distances for HAC. Hence the clustering process can be expressed as the following mapping:

$$V_\phi(\mathcal{P}) \mapsto \hat{y} \quad \text{such that} \quad \hat{y} = \arg\min_{y \in Y} c(V_\phi(\mathcal{P}), y)$$

The clustering ground truth $y^* \in Y$ is the correct k-partition of set $\mathcal{P}$. The training process of COB is governed by a loss function $\mathcal{L}(y^*, \hat{y})$ that optimizes a clustering evaluation metric.

However, we want to emphasize here that the minimization of the cost function $c(V_\phi(\mathcal{P}), y)$ takes place inside the clustering algorithm and remains opaque for our supervised model. As a result, COB is not dependent on the exact clustering algorithm we choose. In this work however, we choose to use HAC as our clustering algorithm. We optimize for RAND index in this work but our method can be applied to optimize for other clustering evaluation metrics as well (e.g. purity).

3.2 Optimizing for RAND index

Our goal is to train the representation model, V_ϕ, such that the resulting clusters maximize a clustering evaluation metric of our choice. In this work, we focus on optimizing for RAND index, a widely used clustering metric, which measures the similarity between the generated clusters and the clustering ground truth. If $y^* \in Y$ be the ground truth partition or the ideal clustering of $\mathcal{P}$, then the clustering quality of a candidate cluster $\hat{y}$ is expressed in terms of RAND index (RI):

$$RI = \frac{\begin{array}{l}\text{No. of unordered data pairs that agrees}\\ \text{between } y^* \text{ and } \hat{y}\end{array}}{\binom{n}{2}}$$

where $n =$ total number of data samples.

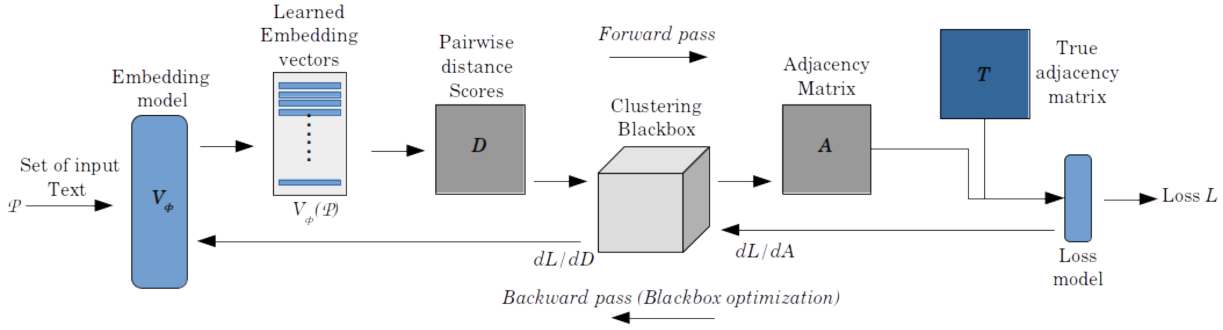

Figure 1: Training loop of our proposed supervised clustering approach.

Table 1: Description of variables used in Figure 1.

Variable	Description
$\mathcal{P}$	Set of documents to be clustered
V_ϕ	Embedding model with trainable parameters ϕ
$V_\phi(\mathcal{P})$	Representation vectors of $\mathcal{P}$ obtained using V_ϕ
D	Pairwise distance matrix of vectors in $V_\phi(\mathcal{P})$
A	Adjacency matrix denoting clustering result
T	Adjacency matrix denoting ground truth clusters

3.3 COB Training Loop

Figure 1 and Table 1 presents the overall training approach. The focus of the training loop is to train the representation model V_ϕ. First, the set of representation vectors $V_\phi(\mathcal{P})$ is obtained for all documents in set $\mathcal{P}$. Then we encode the input to the clustering algorithm as a square symmetric matrix D with pairwise Euclidean distance scores between vectors in $V_\phi(\mathcal{P})$.

$$D_{ij} = ||V_\phi(p_i) - V_\phi(p_j)||_2 \quad \text{where} \quad p_i, p_j \in \mathcal{P}$$

The solution to the clustering problem is expressed in form of an adjacency matrix A such that

$$A_{ij} = 1 \text{ if } i, j \text{ share same cluster and } 0 \text{ otherwise}$$

We denote the adjacency matrix of the clustering ground truth as T. Now, we can express RI using the following form:

$$RI = 1 - \frac{\sum_{ij} |A_{ij} - T_{ij}|}{2\binom{n}{2}} \quad \text{see Appendix}$$

It is clear from the above equation that if we want to maximize RI, we need to minimize the difference between A and T. Intuitively, if we are able to produce ideal clustering results, then A and T would be identical, meaning $A - T$ is a zero matrix. Hence, we define our loss function $\mathcal{L}$ as the sum of $A - T$. Formally:

$$\mathcal{L} = \sum_{ij} |A_{ij} - T_{ij}|$$

The backward pass of this training loop involves estimating the gradient of the loss $\mathcal{L}$ with respect to the distance matrix D, the input to the clustering algorithm. This is achieved using blackbox backpropagation technique and the resulting gradient is used to drive a gradient descent algorithm for training the representation model V_ϕ.

3.4 Regularization

The purpose of any clustering algorithm is to identify groups of similar data points. By optimizing for a clustering metric such as RI, we learn a notion of similarity that most likely yields the ground truth clusters when used in HAC. However, we want to encourage a large margin between similar and dissimilar data points. This is achieved when the loss function encourages *inter-cluster* distances to increase and *intra-cluster* distances to decrease. While this is part of the optimization process within the clustering algorithm, it is opaque during neural network training, due to the blackbox optimization technique. The clustering evaluation metric does not encourage a margin that is larger than necessary. Hence we incorporate a measure of intra versus inter-cluster distance as a regularizer in our optimization criterion as described below.

$$\begin{aligned}\mathcal{L}_r &= \mathcal{L} + r \cdot [\text{mean intra-cluster distance} \\ &\quad - \text{mean inter-cluster distance}] \\ &= \mathcal{L} + r \cdot \left[\underbrace{\frac{\sum_{ij} D_{ij} T_{ij}}{\sum_{ij} T_{ij}}}_{\text{intra-cluster}} - \underbrace{\frac{\sum_{ij} D_{ij}(1 - T_{ij})}{\sum_{ij}(1 - T_{ij})}}_{\text{inter-cluster}} \right]\end{aligned}$$

where r is the regularization constant

The regularization constant r controls how much emphasis is placed on increasing the margin between similar and dissimilar data points versus optimizing the clustering evaluation metric.

Table 2: Dataset statistics: $N =$ total no. of documents, $C =$ total no. of clustering instances, $n =$ average number of documents per clustering instance, $k =$ average number of clusters per clustering instance.

Dataset	N	C	n	k	
20NG train	11314	226	50	18	
20NG test	7532	150	50	18	
				k(coarse)	k(fine)
CAR train	6.8M	597K	11	3.84	5.04
CAR test	6K	126	47	7.78	17.16

4 Experimental Results

In this section, we describe the datasets used for our experiments, discuss our evaluation paradigm and present experimental results that demonstrate efficacy of the representation model trained using our proposed training strategy over our baseline models.

4.1 Datasets

To evaluate our proposed approach, we use two publicly available datasets: 20 newsgroups (20NG[2]) and TREC Complex Answer Retrieval (CAR[3]). As discussed earlier, for our proposed method, each training example consists of the ideal clustering of a set of documents. To produce enough such training samples, we choose to train and evaluate on multiple smaller clustering instances instead of a single but large clustering instance. We note that it will not make any difference in the way our baseline model is trained because they consume the training data in form of triples (SBERT Triplet), as long as we ensure that all models are trained on the same set of clustering examples. We take the following approach to construct such clustering benchmarks from the datasets (detailed statistics are presented in Table 2):

20NG dataset is a widely used public collection of 18846 documents, each categorized into any one of twenty topics. To convert this to a clustering benchmark, both train and test split of 20NG dataset is randomly grouped into sets of 50 documents along with their topic labels, resulting in 226 and 150 clustering instances respectively. Each set of 50 documents represents a single instance of clustering problem.

CAR dataset (version 2.0 year 1) is a large collection of Wikipedia articles. Each article consists of text passages about a topic, segmented into hierarchical subtopics using sections. From the CAR dataset, we use `train.v2.0` as train split (CAR train) and `benchmarkY1test` as test split (CAR test). This dataset is originally designed for a passage retrieval task where passages in CAR articles are relevant for different sections under the overarching topic of the article. This relevance information is part of the dataset in form of the ground truth. We assume that all relevant passages for an article are already retrieved and our focus is to cluster these passages. So each article is a separate clustering problem where our task is to cluster all the passages of the article such that passages from same sections in the original article share the same cluster. We treat the section label under which a passage appears as the clustering label of the passage.

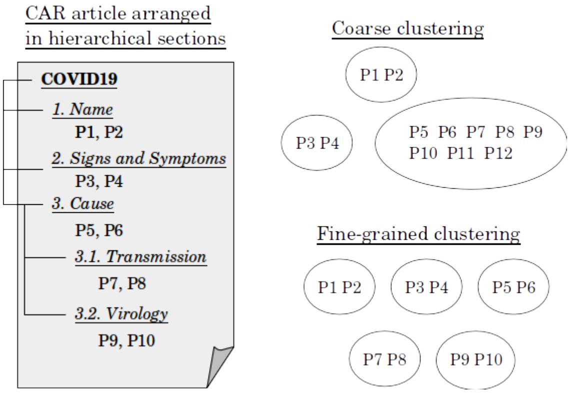

Figure 2: Coarse and fine-grained clustering benchmarks from CAR dataset.

Section labels in CAR dataset are hierarchical. This provides an opportunity to evaluate our clustering models under different levels of granularity. As depicted in Figure 2, passages p_6 and p_7 in article *COVID 19* belong to the sections *Cause* and *Cause/Transmission* respectively. For a coarse-grained view of the clustering, we consider p_6, p_7 under the same topic cluster *Cause*. However, for fine-grained clustering we have to consider p_6, p_7 under separate subtopic clusters. The CAR dataset provides both in form of *top-level* (coarse) and *hierarchical* (fine-grained) benchmarks. We train and evaluate our models on both flavors of the dataset.

4.2 Evaluation Paradigm

Our primary focus is to evaluate the efficacy of our proposed training strategy for supervised clustering and compare it with other training methods while ensuring the fairness of our evaluation. Hence, we train the same text embedding model with the same training data differing only in the training strate-

[2] Part of scikit-learn datasets Pedregosa et al. (2011)

[3] `http://trec-car.cs.unh.edu/`

gies. For the embedding model, we use Sentence-BERT (Reimers and Gurevych, 2019), a recent BERT-based embedding model. Finally, macro-average performance on all clustering instances on the test sets are reported with statistical significance testing. We use three clustering evaluation metrics, RAND index (RI), Adjusted RAND index (ARI) and Normalized Mutual Information (NMI).

Compared methods. In this section we discuss all the methods which are compared in our experiments. All methods are trained until no significant improvement is observed on the validation set. For each method, models are saved on regular interval and we use the best model found during training in terms of validation ARI score to evaluate on the test set.

SBERT COB. We train Sentence-BERT with our proposed training strategy and refer the obtained model as **SBERT COB**.

SBERT Triplet. To compare our approach with a strong supervised baseline, we train Sentence-BERT with Triplet loss function (Dor et al., 2018). It is designed to generate document representations that capture topical similarities. Here, each training example consists of two similar (d, d^+) and one dissimilar (d^-) documents. Triplet loss trains the document representation model V_{trip} so that the Euclidean distance between the similar pair of representations $||V_{trip}(d) - V_{trip}(d^+)||_2$ is less than the negative pair $||V_{trip}(d) - V_{trip}(d^-)||_2$ by at least a margin ϵ.

$$\begin{aligned}\mathcal{L}_{triplet} = \max(0, &||V_{trip}(d) - V_{trip}(d^+)||_2 \\ &- ||V_{trip}(d) - V_{trip}(d^-)||_2 + \epsilon)\end{aligned}$$

Unsupervised baselines. To compare the performances of unsupervised clustering approaches for our use cases, we also include:

1. *SBERT raw*, the pre-trained Sentence-BERT model without any finetuning and
2. *TFIDF* with cosine similarity as a more canonical approach.

4.3 Hyperparameter Optimization

The interpolation parameter λ (Section 2) and regularization constant r (Section 3.4) are two hyperparameters we have to tune in SBERT COB. We use Optuna (Akiba et al., 2019), a recently proposed hyperparameter optimization framework, to search for optimum λ, r pair in terms of validation performance for each dataset. Table 3 presents the optimum hyperparameter values used for our experiments.

Table 3: Optimum values for interpolation parameter λ and regularization constant r found using Optuna.

Dataset	λ	r
NG20	90.0	1.0
CAR coarse	47.0	3.8
CAR fine-grained	103.0	0.3

Table 4: Clustering performance on NG20 dataset in terms of mean RAND index (RI), its corrected for chance version Adjusted RAND Index (ARI) and mean Normalized Mutual Information (NMI). Paired t-test ($\alpha = 0.05$) is carried out with respect to SBERT Triplet (denoted with *) and ▲ and ▼ denotes significantly higher or lower performance.

Method	RI	ARI	NMI
SBERT COB	**0.925**	**0.233▲**	**0.725▲**
SBERT Triplet*	0.924	0.223	0.721
SBERT raw	0.754▼	0.041▼	0.582▼
TFIDF	0.624▼	0.008▼	0.506▼

4.4 Clustering Evaluation

Here we present details of all the experiments carried out and discuss the results. All experiments are executed on a single NVIDIA Titan XP GPU with 12GB memory. For all the SBERT models, we use uncased DistilBERT (Sanh et al., 2019) as the underlying BERT embedding model.

4.4.1 Experiment 1: 20NG

We train SBERT COB and other supervised methods using 80% of the train split of 20NG dataset and the remainder is held out for validation. Table 4 presents the performance on the test set evaluated using mean RI, ARI and NMI.

We observe that our proposed method SBERT COB outperforms all other baselines in terms of RI, ARI and NMI. For ARI and NMI, the improvement is statistically significant in terms of paired t-test with $\alpha = 0.05$ carried out with respect to the best performing baseline, SBERT Triplet. Both TFIDF and SBERT raw fail to obtain meaningful clusters, demonstrating the efficacy of supervised representation models in clustering context.

4.4.2 Experiment 2: CAR

Due to large size of the CAR training split (`train.v2.0`), it is impractical to train SBERT Triplet with all possible triplets in the training set.

Table 5: Dataset statistics: N, C, n, k denotes the same as Table 2, t denotes the total number of available triples to train SBERT Triplet method.

Subset	N	C	k(coarse)	k(fine)	t(coarse)	t(fine)
n=30	71K	2.4K	5.97	10.64	8.6M	5.8M
n=35	56K	1.6K	6.27	12.17	9.3M	5.9M
n=40	50K	1.2K	6.73	13.62	10.8M	6.5M

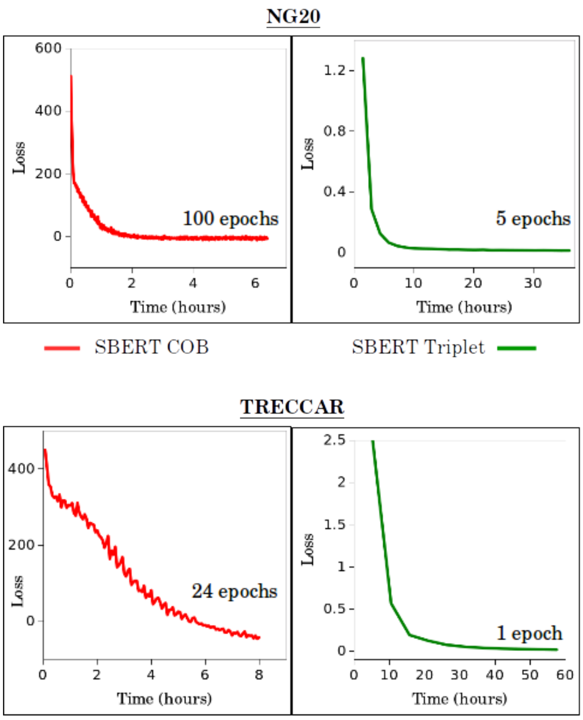

Figure 3: Comparison between SBERT COB and SBERT Triplet in terms of total training time.

Table 6: Coarse-level clustering performance on CAR dataset using top-level benchmarks. Supervised models are trained with set of clustering examples each containing n passages. Paired t-test ($\alpha = 0.05$) is carried out with respect to SBERT Triplet ($n = 30$) and marked with *.

Method	RI	ARI	NMI
Trained on n=30 subset			
SBERT COB	0.742	0.230	0.502
SBERT Triplet*	0.738	0.214	0.494
Trained on n=35 subset			
SBERT COB	**0.744**	**0.236**	0.512▲
SBERT Triplet	0.715▼	0.167▼	0.460▼
Trained on n=40 subset			
SBERT COB	0.726	0.231	**0.514▲**
SBERT Triplet	0.704▼	0.145▼	0.438▼
Unsupervised			
SBERT raw	0.563▼	0.101▼	0.406▼
TFIDF	0.544▼	0.072▼	0.375▼

Table 7: Fine-grained clustering performance on CAR dataset using hierarchical benchmarks. Notations used are same as in Table 6.

Method	RI	ARI	NMI
Trained on n=30 subset			
SBERT COB	0.849	**0.178**	**0.682**
SBERT Triplet*	0.848	0.173	0.678
Trained on n=35 subset			
SBERT COB	0.837▼	0.163	0.672
SBERT Triplet	0.830▼	0.152▼	0.665▼
Trained on n=40 subset			
SBERT COB	0.832▼	0.154	0.666▼
SBERT Triplet	**0.860▲**	0.138▼	0.662▼
Unsupervised			
SBERT raw	0.796▼	0.130▼	0.646▼
TFIDF	0.788▼	0.110▼	0.631▼

Instead, we compare the supervised models trained on three smaller subsets of the training dataset. Each subset contains articles with exactly n passages where $n = 30, 35$ and 40. However, they are always evaluated on the same CAR test set. These values of n are chosen so that we obtain reasonable numbers of training samples while their statistics remain close to the CAR test set on which we are evaluating. Table 5 presents statistics about these three training subsets.

We report the coarse and fine-grained clustering performance in Table 6 and Table 7 respectively. For both coarse and fine-grained clustering, we observe that for each of the training splits ($n = 30, 35, 40$), our proposed method SBERT COB consistently performs better than the best performing baseline, SBERT Triplet ($n = 30$) in terms of both ARI and NMI. As expected, clustering performance in terms of RI score mostly correlates with ARI score. The only exception is SBERT Triplet trained on $n = 40$ for fine-grained clustering. However, we also observe overall decrease in ARI scores for all methods in case of fine-grained clustering. This is expected as fine-grained clustering is a harder problem largely due to fewer passage pairs sharing a cluster. Note that RI and NMI measures are only comparable within table because unlike ARI, it is not adjusted for chance.

4.4.3 Experiment 3: Training Convergence

Existing methods for learning clustering representation spaces, focus solely on classifying individual pairs as similar or different, and hence ignore to which extent other data points already form clusters. The key difference in our work is that we learn the representation space to directly optimize for the clustering evaluation metric, which is based on the clustering results of HAC when used with pairwise Euclidean distances. This allows the model to reach convergence much faster, leading to reduced overall training time, when compared to other methods

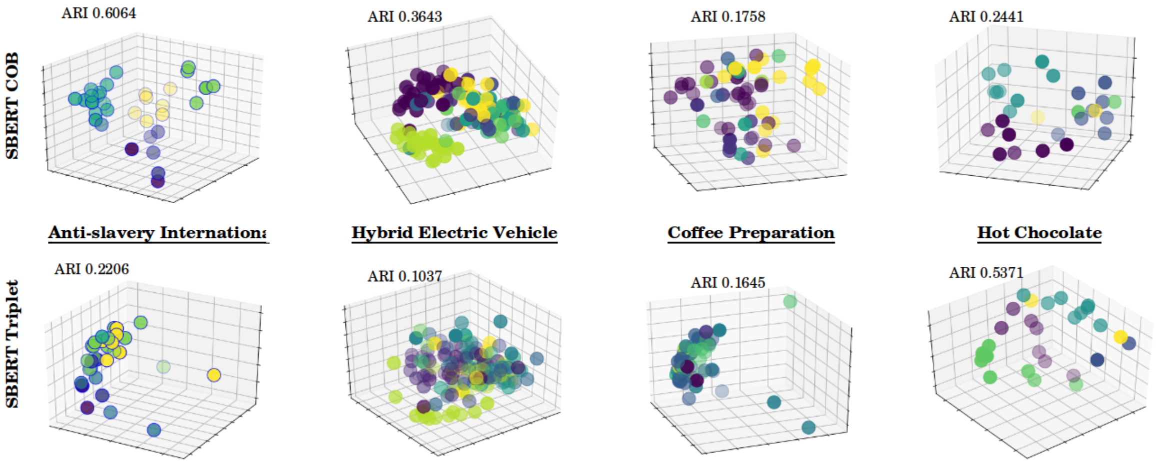

Figure 4: Visual comparison of clustering results between SBERT COB and SBERT Triplet ($n = 35$). Each dot denotes a passage from an article projected into the representation space after applying PCA. Different color denotes different subtopics. Clear separation of different colored blobs indicates good clustering quality.

that uses only a sub-sample of each clustering example (e.g. Triplets). This is particularly helpful in scenarios when we want to regularly update our model to incorporate new training examples.

To demonstrate this we present Figure 5 that compares the time taken to reach convergence during training of SBERT Triplet and SBERT COB on 20NG dataset and CAR dataset (coarse $n = 35$) respectively. For both the datasets, SBERT COB is able to converge at least five times sooner than SBERT Triplet, leading to much faster overall training time. Moreover, for NG20 dataset each epoch of SBERT COB is about 100 times faster than SBERT Triplet. This leads to decrease in overall training time even though SBERT COB takes many more epochs to converge than SBERT Triplet. We observe similar training behaviour for CAR dataset.

4.5 Qualitative Evaluation

Here, we demonstrate efficacy of SBERT COB over SBERT Triplet ($n = 35$) through visual comparison of clustering results from the CAR dataset. Principle Component Analysis (PCA) is used to transform the representation vectors into 3D vectors which are then visualized as points in 3D vector space. Figure 4 compares the results obtained for four articles from CAR test split.

For articles *Anti-slavery International* and *Hybrid Electric Vehicle*, SBERT COB is able to clearly identify clusters of different topics and projects them in different regions of the embedding space. On the contrary, it is difficult to find any clear cluster boundaries in the SBERT Triplet representation space which is also reflected in the ARI scores obtained by the methods. For the article *Coffee Preparation*, both the methods perform poorly in terms of ARI scores. But in case of SBERT COB we see a tendency to separate dissimilar passages. SBERT Triplet projects almost all the passages in a dense region except for a few outlier passages. For the article *Hot Chocolate*, SBERT Triplet obtains numerous small clusters of similar passages. As ARI metric is based on sample-pairs, SBERT Triplet obtains better ARI score even though it does not achieve clear groupings of similar elements.

It is clear from the examples that SBERT COB provides better global clustering quality than SBERT Triplet. This is expected because unlike SBERT Triplet, SBERT COB observes the relationships between all passages in a clustering instance at once to directly optimize for RAND index. Hence, SBERT COB is able to make better global clustering decisions than other pair-based methods.

4.6 Quadratic Scaling of SBERT COB

As SBERT COB learns from all possible interactions of data points in a clustering instance at once, it requires all the adjacency matrices in a batch of clustering samples to fit in memory. Thus the space complexity increases quadratically with the size of each clustering instance. Hence, the batch size is kept small to allow training with a limited GPU memory. However, even with batch size of 1, SBERT COB is observed to obtain superior results in terms of training speed and clustering performance as reported earlier.

5 Conclusion

In this work, we propose an alternative training strategy to train a representation model, for clustering. Our training strategy, COB (Clustering Optimization as Blackbox), directly optimizes the RAND index, a clustering evaluation metric. Using our method, we train SBERT COB, a BERT-based text representation model. We empirically show that SBERT COB significantly outperforms other supervised and unsupervised text embedding model on two separate datasets in terms of RI, ARI and NMI, indicating better cluster quality. Visual representations of the resulting vectors also confirm that SBERT COB learns to holistically distinguish clusters of different topics. Moreover, each epoch in SBERT COB training loop is about 100 times faster when compared to SBERT Triplet, our best performing baseline method. This leads to a significant decrease in overall training time even though SBERT COB requires more iterations to converge than SBERT Triplet. This makes SBERT COB suitable for applications that require clustering models to be updated on a regular basis as new training samples become available. Lastly, although we have conducted experiments with a specific clustering algorithm (HAC) and a clustering metric to optimize (RAND index), our model is independent of the particular choice of algorithm or the metric.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Peter L Bartlett. 1998. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536.

Sugato Basu, Arindam Banerjee, and Raymond Mooney. 2002. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer.

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2017. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 5879–5887.

Chien-Hsing Chen. 2017. Improved tfidf in big news retrieval: An empirical study. *Pattern Recognition Letters*, 93:113–122.

Douglass R Cutting, David R Karger, Jan O Pedersen, and John W Tukey. 2017. Scatter/gather: A cluster-based approach to browsing large document collections. In *ACM SIGIR Forum*, volume 51, pages 148–159. ACM New York, NY, USA.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Liat Ein Dor, Yosi Mass, Alon Halfon, Elad Venezian, Ilya Shnayderman, Ranit Aharonov, and Noam Slonim. 2018. Learning thematic similarity metric from article sections using triplet networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54.

Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. 2017. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, pages 5736–5745.

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.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2016. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. 2009. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22.

Fengfu Li, Hong Qiao, and Bo Zhang. 2018. Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, 83:161–173.

Jian-Ping Mei and Lihui Chen. 2014. Proximity-based k-partitions clustering with ranking for document categorization and analysis. *Expert systems with applications*, 41(16):7095–7105.

Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. 2018. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514.

Roberto Navigli and Giuseppe Crisafulli. 2010. Inducing word senses to improve web search result clustering. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 116–126.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ugo Scaiella, Paolo Ferragina, Andrea Marino, and Massimiliano Ciaramita. 2012. Topical clustering of search results. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 223–232, New York, NY, USA.

Nikos Tsirakis, Vasilis Poulopoulos, Panagiotis Tsantilas, and Iraklis Varlamis. 2017. Large scale opinion mining for social, news and blog data. *Journal of Systems and Software*, 127:237–248.

Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. 2019. Differentiation of blackbox combinatorial solvers. *arXiv preprint arXiv:1912.02175*.

Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. 2019. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR.

Bryan Wilder, Eric Ewing, Bistra Dilkina, and Milind Tambe. 2019. End to end learning and optimization on graphs. *arXiv preprint arXiv:1905.13732*.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR.

Jiaming Xu, Bo Xu, Peng Wang, Suncong Zheng, Guanhua Tian, and Jun Zhao. 2017. Self-taught convolutional neural networks for short text clustering. *Neural Networks*, 88:22–31.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR.

Jianwei Yang, Devi Parikh, and Dhruv Batra. 2016. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5147–5156.

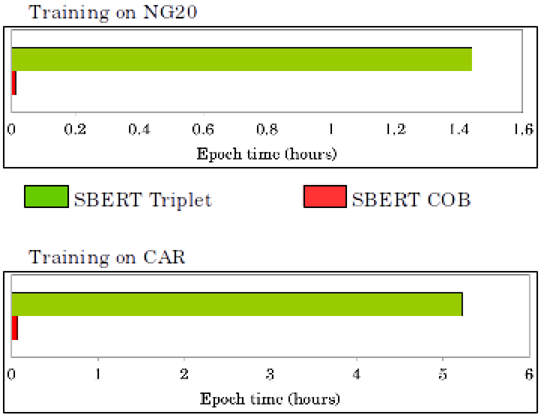

Figure 5: Comparison between SBERT COB and SBERT Triplet in terms of epoch time.

A Relation between RAND index and Adjacency matrix

Given a set of n data points $\mathcal{P}$, let us compare two clustering results of $\mathcal{P}$, C_T and C_A, in terms of RAND index. We know that RAND index is expressed as:

$$RI = \frac{a+b}{\binom{n}{2}}$$

where $a =$ number of pairs that share the same cluster both in C_T and C_A

where $b =$ number of pairs that are from different clusters both in C_T and C_A

Now we can express any clustering result C_M in form of an adjacency matrix M where $M_{ij} = 1$ if the i, j-th data points in $\mathcal{P}$ share the same cluster in C_M and $M_{ij} = 0$ otherwise. We represent the clustering results C_T and C_A with such adjacency matrices T and A respectively. Also, the difference matrix of A, T denoted as $|A - T|$ indicates the ordered pairs that do not agree between A, T. In other words, $|A_{ij} - T_{ij}| = 1$ denotes that the i, j-th data points do not agree between A and T. Now, we can express RAND index in terms of A and T as follows:

$$
\begin{aligned}
RI &= \frac{a+b}{\binom{n}{2}} \\
&= \frac{\text{No. of agreements between } C_T, C_A}{\binom{n}{2}} \\
&= \frac{\text{No. of unordered pairs in } \mathcal{P} \text{ that agrees between } C_T, C_A}{\binom{n}{2}} \\
&= \frac{\text{No. of ordered pairs in } \mathcal{P} \text{ that agrees between } C_T, C_A}{2\binom{n}{2}} \\
&= \frac{\text{Total ordered pairs in } \mathcal{P} - \sum_{ij} |A_{ij} - T_{ij}|}{2\binom{n}{2}} \\
&= \frac{2\binom{n}{2} - \sum_{ij} |A_{ij} - T_{ij}|}{2\binom{n}{2}} \\
&= 1 - \frac{\sum_{ij} |A_{ij} - T_{ij}|}{2\binom{n}{2}}
\end{aligned}
$$

B Comparison of Epoch Time

Figure 5 shows the mean epoch time of SBERT Triplet and SBERT COB on 20NG dataset and CAR dataset (coarse n = 35) respectively.