Collaborative is better than Adversarial: Generative Cooperative Networks for Topic Clustering

Andrea Lenzi
Computer Science Department
of Sapienza, University of Rome
Rome, Italy
lenzi@di.uniroma1.it

Paola Velardi
Computer Science Department
of Sapienza, University of Rome
Rome, Italy
velardi@di.uniroma1.it

ABSTRACT

Topic modeling is a popular technique for learning the thematic structure of large corpora composed of unlabeled documents, without human supervision. In recent years, various neural network-based algorithms have been proposed to solve this task. In particular, there is an extensive literature showing how Variational AutoEncoders (VAEs) and Generative Adversarial Networks (GANs) approaches have been successful in identifying recurrent discussion topics. In this paper we propose a new neural topic detection model called Generative Cooperative Topic Modeling (GCTM), in which a Generator and a denoising AutoEncoder, rather than learning through a competitive process, act cooperatively. We show that this cooperative model has a faster convergence and surpasses the adversarial approach, as well as other popular topic detection algorithms based on VAEs, when tested on three common public datasets and with a variety of performance indicators.

CCS CONCEPTS

• **Information systems** → **Clustering**; • **Computing methodologies** → **Natural language processing**; **Topic modeling**; **Neural networks**;

KEYWORDS

Topic Modeling, Clustering, Natural Language Processing, Neural Networks, Generative Adversarial Networks, AutoEncoders

ACM Reference Format:
Andrea Lenzi and Paola Velardi. 2022. Collaborative is better than Adversarial: Generative Cooperative Networks for Topic Clustering. In *The 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22), April 25–29, 2022, Virtual Event, .* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3477314.3506997

1 INTRODUCTION

Topic modeling is a popular technique to learn the thematic structure of large corpora, based on dimensionality reduction and exploratory data analysis of text contents. Using contextual evidence, topic models can connect words with similar meanings and detect the semantic structure of a document collection. In recent years various algorithms based on neural networks have been proposed to solve this unsupervised task. In particular, there is an extensive literature [13, 24–26] showing how Variational AutoEncoders (VAEs) and Generative Adversarial Networks (GANs) have been successful in identifying common discussion topics. GANs are based on a game-theoretic approach, in which the network learns from training data through a 2-player game. Two competitors, a Generator (G) and a Discriminator (D), fight one against the other to generate new synthetic instances that may resemble real data. However, it has been shown in numerous works (e.g., [28] and [22]) that this battle may lead to instability, such as non-convergence, vanishing or exploding gradients, and mode collapse. On the other hand, AutoEncoders tend to overfit and to model the noise present in the data. In some unfavourable circumstances, they merely learn the identity function of the training data. Furthermore, the latent space modeled by AutoEncoders often does not have good exploitable properties. For example, similar samples once encoded will not be close to each other in the compressed space. To overcome these drawbacks, several variants of the classic AutoEncoders scheme have been introduced, such as Variational AutoEncoders (VAEs), a generative model that provides probabilistic descriptions of observations in latent spaces. However, as noted in [9], VAE may tend to generate unfocused samples when using standard loss functions.

In this paper, we propose Generative Cooperative Topic Modeling (GCTM), based on four sequential steps: i) dimensionality reduction through two cooperative neural networks, ii) agglomerative hierarchical clustering, iii) summarization of the most relevant words from the extracted clusters and iv) optimization through embeddings similarity in the inferred vector space. The algorithm has been designed to avoid over-fitting for a better generalization on new unseen documents, and to obtain a faster convergence, thus mitigating the problems encountered with GANs and classical AutoEncoders.

Through the proposed strategy, we make the initial training phase more robust. In fact, in an adaptive way, the generative network introduces randomness in the text compression process implemented by the other agent, an AutoEncoder. The two networks act cooperatively since, as the learning process proceeds, the generator tends to create increasingly more realistic synthetic samples, thus helping the AutoEncoder to learn how to compress increasingly less difficult samples, which leads to a faster convergence.

The paper is organized as follows: in Section 2 we summarize related literature. Section 3 presents the proposed approach for topic modeling. In Section 4, we compare GCTM with state-of-the-art GANs and VAEs systems, on three popular datasets and different performance indicators. Section 5 presents our concluding remarks. To avoid distracting readers' attention from the focus of the article, technical details on the proposed approach for topic modeling are provided as supplementary material in the Appendices.

2 RELATED WORK

Topic modeling has been widely applied in the field of computer science with a specific focus on text mining and information retrieval. In recent years it has also been used to discover abstract themes found in other types of collections like biological data-sets, genetic information, images, and networks ([15], [23], [16]). Topic modeling represents a very complex task, requiring a considerable level of understanding of the text. Furthermore, it is not based on a ground truth, being a totally unsupervised task. Thus, existing models frequently fail to generalize well and to extract valuable knowledge from the data. To advance on LDA [4], an approach that has long be considered the state of the art in topic modeling, a variety of solutions have been presented in recent literature, as surveyed, for example, in [14] and [6]. Among the most recent approaches, [24] present AVITM, a variant of Variational AutoEncoders (VAEs). VAEs are probabilistic generative models, that encode the input data as a probability distribution over the latent space. A VAE can be defined as an AutoEncoder whose encoding distribution is regularised during the training, to ensure that its latent space models a fixed prior distribution, with the aim of improving the properties of the generative process. One key problem of VAEs is the strong assumption about the underlying probability distribution (such as Gaussian) of the source data. Moreover, implementing a Variational AutoEncoder is much more challenging than implementing an AutoEncoder.

Another approach that recently gained popularity is Generative Adversarial Networks (GANs). GANs are generative architectures in which two adversarial neural nets, a discriminator D and a generator G, are trained simultaneously. The D is trained to distinguish real samples of a dataset from fake samples constructed by the generator, while, at the same time, the G is trained to produce synthetic samples from a random source, in order to train the D to distinguish between the fake generated data and the real data. One of the first publications exploiting GANs for topic clustering is [26], where the authors propose to model topics with Dirichlet prior and to employ a G to capture the semantic patterns among latent topics. They use the G to learn the projection function between the document-topic distribution and the document-word distribution while the D is trained to detect if the input document is real or fake. As the authors suggest, the supervision provided by the D in the adversarial training phase helps the G to capture the semantic patterns embedded in the latent topic space. However, as already remarked in the Introduction, GANs do exhibit a number of problems such as slow or non-convergence, vanishing gradient, and mode collapse of the generator, that produces a limited number of examples. Other variants and improvements of VAEs and GANs for topic modeling have been proposed in [25], [13], [1] and [2]. More details on these systems are provided in the experimental Section 4.2.

3 PROPOSED APPROACH

In this Section we present Generative Cooperative Topic Modeling (GCTM), based on the following four sequential steps:

(1) *Dimensionality reduction* through a collaborative variant of a Generative Adversarial Network. The purpose of this step is to reduce feature sparsity, avoid over-fitting in presence of limited data, and achieve a higher generalization;
(2) *Agglomerative hierarchical clustering,* to create clusters of similar documents;
(3) *Summarization* of the most relevant words from the detected clusters;
(4) *Optimization* of the extracted topics through word embeddings computed on the original corpus.

Initially, a neural network extracts the most relevant features representing each message (referred to hereafter as a "document") as a compressed latent vector. Subsequently, these vectors are grouped to obtain a set of clusters. Afterwards, for each cluster of messages, we recover the most significant words and we associate them to a single discussion topic. Finally, we improve the obtained topics trough word embeddings computed on the input data. The algorithm was designed with the goal of identifying high-quality discussion topics in the collection in a limited number of epochs. In what follows, we illustrate the proposed approach in more detail.

3.1 Dimensionality reduction

Starting from the concept of Energy-based Generative Adversarial Network (EBGAN) suggested by Zhao et al. [30] and from the architecture proposed by Glover [10], we developed a model to extract the most significant latent features and learn compressed representations from unlabeled documents in a collection. In the original formulation of GAN proposed by Goodfellow et al. [11], the discriminator was a probabilistic binary classifier with logistic output, and convergence occurred when the distribution produced by the generator matches the data distribution. More recently, Zhao et al. [30] proposed an innovative model called Energy-based Generative Adversarial Network (EBGAN) that exhibits more stable behavior during training and furthermore enables the implementation of an extensive variety of Neural Networks architectures and loss functions for the discriminator. In the EBGANs, the discriminator represents an energy function[1] that is trained to attribute low energies to the "real" data and higher energies to the generated synthetic samples. In parallel, the generator is trained to produce contrasting samples with minimal energies.

Building on the previously illustrated techniques, we propose a model consisting of two neural nets that, instead of acting adversely, are trained simultaneously in a cooperative way. During this simultaneous training of both networks, for each batch of data: the two loss functions of the nets are calculated sequentially, and then the two respective gradient signals are back-propagated. Our model represents a hybrid approach between an EBGAN and a denoising AutoEncoder and is composed by two networks (see Figure 1):

[1]A cost function that maps each point of an input space to a single scalar, which is called "energy".

- The *generative network* learns to produce synthetic instances starting from random uniform inputs. Its purpose is to generate examples progressively more and more similar to real input data. In our model, the generator has the primary objective of adaptively introducing noise into the training process of the two networks, to regularize the process, and consequently *increasing the model's resistance to over-fitting*.
- The *compression network* is a Denoising AutoEncoder. In the first place, it learns how to efficiently encode the input data (both real and synthetic instances) into a lower-dimensional space. Subsequently, it learns how to reconstruct the input data back from the reduced encoded latent space, to a representation that is as close to the original input as possible. To reinforce the ability to generalize, in each layer of this network we applied dropout of random units and L1 regularization of the activation function and weights matrix.

Unlike for GANs, in our model the output of the generative part does not represent the focus of the process, but mainly serves to improve and further regularize the compression network. Furthermore, the two networks tend to quickly converge together, since they are not trained in an adversarial manner. Let X be a real data instance coming from the dataset, $G(Z)$ a synthetic instance produced by the generator from a random source Z[2], and $A(s)$ the output instance reconstructed by the decoder starting from an input instance s[3]; then the AutoEncoder loss L_A and the generator loss L_G are formally defined by:

$$stat\left(\vec{v}\right) = \left\langle min\left(\vec{v}\right), E\left(\vec{v}\right), var\left(\vec{v}\right), max\left(\vec{v}\right)\right\rangle$$

$$L_G\left(G\left(Z\right), X\right) = MSE\left\{stat\left(G\left(Z\right)\right), stat(X)\right\} \quad (1)$$

$$\begin{aligned} L_A\left(G\left(Z\right), X\right) = & \\ & - cos\_sim\left\{G\left(Z\right), A\left(G\left(Z\right)\right)\right\} \\ & - cos\_sim\left\{X, A\left(X\right)\right\} \end{aligned} \quad (2)$$

In our experiments, each instance X is a text document encoded with the TF-IDF weighting scheme. For the purposes of our analysis, we have only considered uni-grams, since low level features are less scattered and more easily map onto dense semantic spaces. After encoding each corpus, we obtained a vocabulary V consisting of $|V|$ uni-grams.

3.2 Agglomerative hierarchical clustering

Once the data samples are reduced to a compressed form containing the essential and most significant features of the original text documents, we generate document clusters, based on these dense representations. To this aim, we applied agglomerative hierarchical clustering with Ward method [27] as linkage criterion. We selected this popular technique for its flexibility. In fact, it presents few hidden hypotheses on the distribution of the underlying data and fits properly in all the contexts in which the shape of the clusters is not hyper spherical.

[2] Z is 1-D random uniform vector. Usually, in a GAN, the generator takes a random source as its input and transforms this noise into a meaningful output (data instance).
[3] s could be either an authentic document coming from dataset or a synthetic document produced by the generator.

3.3 Summarization

After determining the groups of highly related documents, we rank the most representative words for each of these clusters. Specifically, for each subset of documents associated to a single cluster, we rank the words in this subset based on a weighting scheme. For each word w belonging to cluster C, we considered as the relevance score for w in C the ratio between the frequency of the term w within the documents belonging to cluster C, and the global frequency of the term w in the whole corpus. In particular, we applied the following formula to compute the relevance score rel of word w in cluster C:

$$rel\left(w, C\right) = \frac{inner\_term\_frequency\left(w, C\right)}{\sqrt{global\_term\_frequency\left(w\right)}} \quad (3)$$

3.4 Embeddings Optimization

In the last step, we refine the extracted topics through a technique based on word embeddings. We calculated the embeddings on the input data via Word2Vec. Then, for each topic, we consider a super-set S of the top m words (according to the previously computed ranking) in each topic ($|S| > m$), where S and m are hyper-parameters, and we iteratively eliminate the most dissimilar words from the current super-set S, until $|S| = m$.

In the experimentation phase, we explored two versions of the proposed model: a variant with (***GCTM-e***) and a variant without embeddings optimization (***GCTM-b***).

Additional implementation details and hyperparameters are provided in the annexed Supplementary Material.

4 EXPERIMENTS

We evaluate the two proposed model variants (*GCTM-b*, *GCTM-e*) and compare them with 5 VAEs and GANs systems, on three commonly used datasets. First, we describe the datasets, the compared models and the evaluation metrics. Subsequently, we discuss the obtained results.

4.1 Datasets Selection

For our experiments, we evaluated the performance of the applied models using the following popular public datasets[4].

- *Twenty Newsgroups*: it comprises 18,846 newsgroups posts cover 20 different categories, from baseball to religion. This dataset is split in two subsets: one for training (11,314 texts) and the other one for testing (7,532 texts). We retrieved this corpus through the Scikit-Learn [19] datasets APIs.
- *IMDB Reviews* [17]: it contains movie reviews along with their associated binary sentiment polarity labels. The core dataset contains 50,000 samples (25,000 for training and 25,000 for testing) and it is well balanced on the classification labels. Moreover, it also includes an additional 50,000 unlabeled documents. We downloaded the dataset from the following URL: https://ai.stanford.edu/~amaas/data/sentiment/.

[4] Each dataset presented a section for training and one for testing. Since topic detection is an untrained task, we extracted the topics only from the training collection.

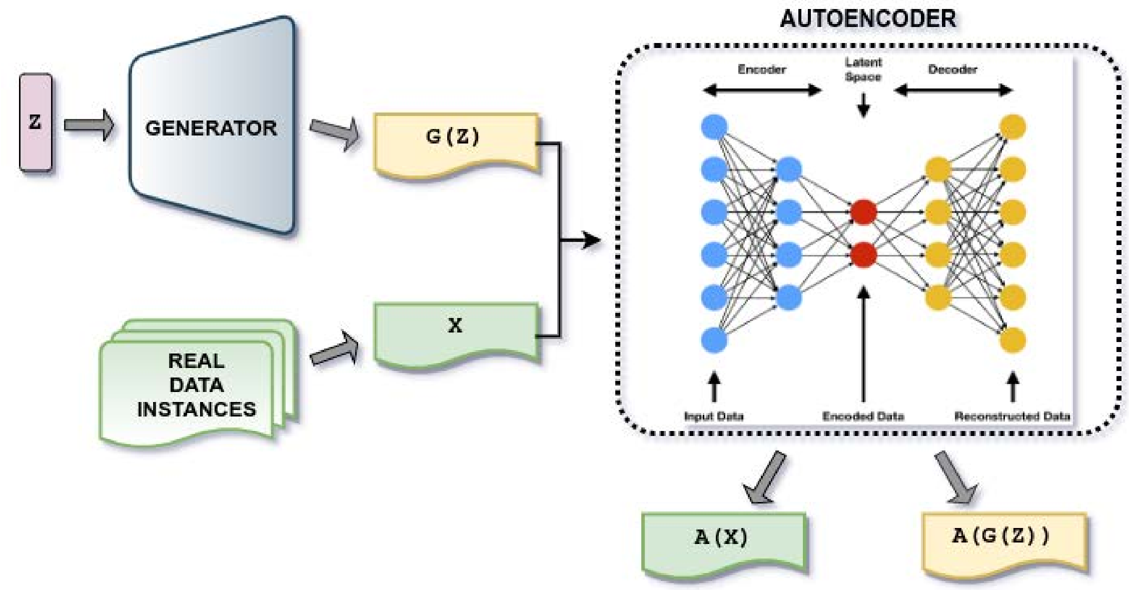

Figure 1: Architecture of the proposed Cooperative Neural Networks for text compression

- *Reuters*: it is a benchmark dataset for document classification. To be specific, it is a multi-class and multi-label dataset. Thus, each document can have associated many different classes that overlap with each other. This dataset presents 90 classes, 7,769 training documents and 3,019 testing documents. We retrieved this corpus through the NLTK [3] datasets APIs.

4.2 Models for comparison

We compare with the following state-of-the-art VAEs and GANs models:

Autoencoded Variational Inference For Topic Model (AVITM)[5] [24]: it is a neural topic modeling approach based on a Variational AutoEncoder and designed to obtain readable topics. Toward this aim, it replaces the multivariate gaussian distribution of the Neural Variational Document Model (NVDM) [18] with a logistic normal distribution. *AVITM* presents two models: ***neuralLDA***, that approximates the Dirichlet prior using a Logistic-Normal distribution as the prior of the latent space; and ***prodLDA***, a variant of the previous model that assumes that the distribution over individual words is a product of experts rather than a mixture model.

Adversarial-neural Topic Model (ATM)[6] [26]: it is a neural topic modeling algorithm based on adversarial training in which two neural networks (a generator and a discriminator) compete against each other, trying to optimize the Wasserstein Loss with a Gradient Penalty. To capture the multi-modality, the generative part of this architecture tries to model topics with a Dirichlet prior instead of a logistic-normal prior.

Contextualized Topic Models (CTM)[7]: They represent a recent family of Neural Variational topic models that exploits transfer learning, using pre-trained representations of natural language (e.g., Transformer based networks). This family constitutes an evolution of *prodLDA* and presents two models: ***CTM-ZeroShoot*** [2], that models the documents with contextual embeddings and applies variational inference, and ***CTM-combined*** [1], that combines contextual embeddings and the Bag Of Words representation to model documents.

4.3 Evaluation Metrics

Evaluation of clustering results is notoriously a complex problem, especially in the absence of a ground truth [12]. For humans, when the description of objects by their features reaches higher dimensions, intuitive judgements are less easy to obtain and justify.

Researchers have attempted to solve the problem of cluster evaluation using a variety of *internal validity measures* based on the notions of cohesion, separation or some combination of these quantities. We selected five metrics widely used in literature for unsupervised evaluation of clustering results and topic coherence, described below:

- *calinski-harabasz score (chs)* [7]: it is also known as the Variance Ratio Criterion. This measure is defined as the ratio of between-clusters dispersion (B) and of inter-cluster dispersion (W) for all clusters (dispersion represents the sum of

[5]We performed the experiments with the following implementation: https://github.com/estebandito22/PyTorchAVITM [8].
[6]We developed and provided our own implementation following the indications of the paper, as an official implementation was unavailable.
[7]We started from the Bert base model "bert-base-nli-mean-tokens", and we exploited the official implementation: https://github.com/MilaNLProc/contextualized-topic-models.

distances squared). A higher Calinski-Harabasz score relates to a model with better defined clusters.

- *silhouette score (ss)* [20]: it represents the mean silhouette coefficient of all samples. For this measure, the best value is 1 and the worst value is -1, while values near 0 indicate overlapping clusters.
- Furthermore, in line with [25], [2] and many others, we compute *topic coherence* [21], based on the idea that a set of statements/facts is coherent if they support each other. It combines a number of different measures (*segmentation, confirmation, aggregation* and *probability estimation*[8]) into a framework to evaluate the coherence among the word clusters inferred by a model, estimating the degree of semantic similarity between high scoring words in the topic, similarly to human judgment. We use three different coherence measures: i) *C_umass*: it is based on document co-occurrence counts, a “one-preceding” segmentation, and a logarithmic conditional probability as confirmation measure; ii) *C_uci*: it is based on a sliding window and the Pointwise Mutual Information (PMI) of all word pairs of the given top words; iii) *C_npmi*: it is an enhanced version of the *C_uci* coherence measure, using the Normalized Pointwise Mutual Information (NPMI). For all these measures, higher values are better.

In addition to these well known metrics, we introduce a new performance indicator, the *Quality Score (qs)*. In our proposed measure, let $T = \{t_1, t_2, \cdots, t_k\}$ be the set of all the k extracted topics, where $t_i = \{v[w_1], v[w_2], \cdots, v[w_n]\}$ represents the *i-th* topic containing the embedding vectors of its top n words; let $centr(t_i)$ be the vector that represents the multidimensional mean of the word vectors in topic t_i; let $m \in \mathbb{N}$ and $comb(S, m)$ be the set that contains all the combinations of size m from the set S, then we define this metric as follows:

$$intra\_sim\,(t) = mean\{cos\_sim\left(v\,[w]\,', v\,[w]\,''\right) \mid \forall\left(v\,[w]\,', v\,[w]\,''\right) \in comb\,(t, 2)\} \tag{4}$$

$$inter\_sim\left(t', t''\right) = cos\_sim\left(centr\left(t'\right), centr\left(t''\right)\right) \tag{5}$$

$$qs = \frac{mean\,\{intra\_sim(t) \mid \forall t \in T\}}{mean\,\{inter\_sim\,(t', t'') \mid \forall\,(t', t'') \in comb\,(T, 2)\}} \tag{6}$$

This score is in the range [0, 1] (higher values are correlated with better clusters). We introduced this additional measure, because it more interpretable than the other metrics, being normalized in the range [0,1]; moreover, it presents a higher variance than the other metrics.

To compute these six measures, we use the following strategy: initially, for each extracted topic we considered only the n most relevant words, after which we encoded each term in its respective embedding vector, and finally we calculated the internal validity scores for the obtained clusters based on the six evaluation metrics. To compute the embeddings, we now exploited pre-trained word vectors with FastText [5] model for the English language [9]. We chose to use embeddings vectors computed on an external large corpus (rather than on the same corpus, as in Section 3.4), because using the same dataset that generated the topics could reinforce noise and bias the evaluation.

[8]Details can be retrieved from the [21].

[9]Continuous Bag Of Words with position weights, size 300 and n-grams characters of length 5.

Table 1: Results obtained on 20 newsgroups dataset

	ATM	CTM-Comb	CTM-ZS	GC TM-b	GC TM-e	neural LDA	prod LDA
K = 10							
qs	0.134	0.295	0.316	0.303	**0.453**	0.19	0.263
chs	0.999	4.218	4.726	3.83	**5.876**	2.316	3.419
ss	-0.027	0.013	**0.022**	0.018	0.019	-0.022	0.002
c_umass	-12.536	-5.932	-5.585	**-4.234**	-4.366	-5.268	-7.103
c_uci	-8.687	-3.349	-3.42	-2.201	**-2.025**	-6.802	-4.663
c_npmi	-0.308	-0.07	-0.071	-0.024	**-0.01**	-0.228	-0.132
K = 20							
qs	0.134	0.292	0.301	0.307	**0.399**	0.187	0.286
chs	1.103	3.835	3.988	3.842	**4.928**	2.161	3.873
ss	-0.027	-0.01	-0.009	**-0.004**	-0.005	-0.048	-0.011
c_umass	-15.258	-6.346	-6.046	-7.191	**-6.025**	-5.792	-6.784
c_uci	-9.727	-4.602	-3.949	-4.211	**-2.988**	-6.997	-4.417
c_npmi	-0.348	-0.112	-0.087	-0.094	**-0.039**	-0.237	-0.109
K = 30							
qs	0.128	0.268	0.276	0.282	**0.39**	0.191	0.281
chs	1.008	3.455	3.575	3.457	**4.889**	2.353	3.654
ss	-0.031	-0.028	-0.025	**-0.02**	-0.005	-0.05	-0.026
c_umass	-18.198	-7.324	**-6.772**	-9.003	-7.47	-7.165	-6.884
c_uci	-10.68	-5.143	-4.956	-5.513	**-4.293**	-6.836	-5.0
c_npmi	-0.384	-0.131	-0.126	-0.144	**-0.086**	-0.226	-0.127

4.4 Results

For each baseline model, we ran the evaluation tests, starting from the default configuration and the hyper-parameters set described in the original papers. Details on the hyper-parameters of our model are available on the appendices.

We report in Tables 1, 2, and 3 the scores calculated using the previously defined metrics on the three selected datasets. These values are obtained with all the compared topic detection approaches, varying the selected number of clusters and considering the top $n = 50$ words[10] for each topic.

The Table show the average scores computed in three separate experiments with different seeds (random states). In all the tables, bold numbers are the “best”[11] absolute values for each of the six considered topic validity scores, when varying the number of topics K (10, 20, 30). We note that GTCM surpasses all methods in at least 3, and more often 4, out of 6 measures in all experiments. No other method systematically predominates. Furthermore, in the *20 newsgroups* and *IMDB reviews* datasets the embedding optimization (GCTM-e) significantly improves performances, while in the Reuters dataset GTCM-b surpasses GTCM-e. This is understandable, since the Reuters dataset has 90 classes, fewer documents and a multi-label classification, therefore learning embeddings locally from these data leads to worst results. In general, the *Reuters* dataset (Table 3) is semantically more heterogeneous with respect to the other datasets, leading to comparatively worst performances for all systems.

[10]n is an hyper-parameter that should be tuned depending on the dataset, but for comparative purposes, we empirically selected $n = 50$ in all experiments and systems.

[11]Note that “best” means the highest or lowest values, based on the considered metric, as previously explained.

Table 2: Results obtained on IMDB reviews dataset

	ATM	CTM-Comb	CTM-ZS	GC TM-b	GC TM-e	neural LDA	prod LDA
K = 10							
qs	0.147	0.252	0.278	0.237	**0.312**	0.164	0.285
chs	0.484	3.134	3.681	2.374	**4.231**	1.394	3.838
ss	-0.027	-0.026	-0.016	-0.033	**-0.015**	-0.03	-0.02
c_umass	-6.495	-8.838	-8.211	-6.598	-5.403	**-3.714**	-6.911
c_uci	-4.662	-4.522	-3.964	-2.776	**-1.999**	-2.883	-3.594
c_npmi	-0.165	-0.151	-0.125	-0.09	**-0.048**	-0.107	-0.11
K = 20							
qs	0.156	0.269	0.271	0.243	**0.3**	0.168	0.276
chs	1.298	3.579	3.492	2.436	**3.668**	1.459	3.538
ss	-0.035	-0.049	-0.04	-0.05	**-0.028**	-0.043	-0.047
c_umass	-11.197	-8.614	-7.668	-8.97	-5.925	**-4.315**	-8.209
c_uci	-7.146	-4.477	-3.875	-3.906	**-2.334**	-3.456	-4.258
c_npmi	-0.256	-0.14	-0.116	-0.124	**-0.06**	-0.125	-0.127
K = 30							
qs	0.153	0.269	0.271	0.236	**0.309**	0.172	0.264
chs	1.404	3.453	3.407	2.309	**3.909**	1.636	3.264
ss	**-0.041**	-0.058	-0.045	-0.053	-0.045	-0.051	-0.06
c_umass	-13.367	-8.181	-7.872	-11.22	-7.718	**-4.969**	-8.696
c_uci	-8.353	-4.071	-4.011	-5.247	**-3.591**	-4.381	-4.79
c_npmi	-0.299	-0.124	-0.118	-0.172	**-0.104**	-0.153	-0.143

Table 3: Results obtained on Reuters dataset

	ATM	CTM-Comb	CTM-ZS	GC TM-b	GC TM-e	neural LDA	prod LDA
K = 10							
qs	0.133	0.23	**0.228**	0.211	0.218	0.202	0.222
chs	0.503	2.739	**3.117**	2.254	2.465	2.491	2.726
ss	-0.028	-0.009	-0.009	-0.026	-0.016	-0.019	**-0.007**
c_umass	-8.254	-8.647	-8.543	**-6.739**	-8.301	-9.449	-8.355
c_uci	-7.467	-4.436	-4.431	**-3.282**	-3.75	-6.089	-4.374
c_npmi	-0.252	-0.119	-0.12	**-0.077**	-0.078	-0.2	-0.118
K = 20							
qs	0.132	0.211	0.218	0.205	**0.238**	0.178	0.21
chs	0.843	2.492	2.61	2.155	**2.667**	1.934	2.39
ss	-0.032	**-0.024**	-0.025	-0.032	-0.029	-0.036	-0.026
c_umass	-12.452	-8.612	**-8.288**	-8.957	-10.003	-11.075	-8.526
c_uci	-9.238	-4.798	-4.553	**-4.418**	-5.132	-6.767	-4.782
c_npmi	-0.321	-0.126	-0.115	**-0.108**	-0.131	-0.225	-0.127
K = 30							
qs	0.135	0.206	0.212	0.195	**0.232**	0.181	0.209
chs	1.113	2.335	2.48	1.972	**2.554**	2.053	2.512
ss	**-0.032**	-0.033	**-0.032**	-0.037	-0.035	-0.043	-0.033
c_umass	-15.996	**-9.065**	-9.185	-10.848	-10.345	-11.529	-9.269
c_uci	-10.505	**-5.308**	-5.348	-5.542	-5.379	-7.03	-5.325
c_npmi	-0.372	-0.142	-0.144	-0.152	**-0.14**	-0.232	-0.145

4.5 Convergence

As intuitively discussed in Sections 1 and 3.1, the collaborative approach adopted in GTCM favours faster convergence. To experimentally demonstrate this claim, Table 4 shows the performance of all systems on all datasets[12] after 5 epochs, while for GTCM-b and GTCM-e the performance refers to the first epoch only. In fact, we noticed that the method tend to stabilize after just one epoch, as it can be seen by comparing GTCM results in Table 4 with those in previous Tables 1, 2 and 3. This is a distinctive feature of the proposed approach when compared with GANs and VAEs.

[12]For the sake of space, the parameter K is set to 20.

Table 4: Comparison of models after fitting on data for 5 epochs for all systems, and only one epoch for GCTM

		ATM	CTM-Comb	CTM-ZS	GC TM-b	GC TM-e	neural LDA	prod LDA
Twenty	*qs*	0.129	0.218	0.227	0.29	**0.376**	0.176	0.212
news	*chs*	1.009	2.869	2.854	3.572	**4.693**	1.758	2.44
groups	*ss*	-0.028	-0.034	-0.034	-0.014	**-0.01**	-0.047	-0.035
	umass	-16.436	-6.681	-7.269	-7.254	**-5.533**	-6.838	-9.973
	uci	-10.27	-4.99	-4.186	-4.22	**-2.803**	-6.958	-5.728
K=20	*npmi*	-0.367	-0.152	-0.123	-0.094	**-0.029**	-0.243	-0.186
IMDB	*qs*	0.139	0.21	0.236	0.254	**0.315**	0.165	0.195
reviewers	*chs*	1.079	2.126	2.752	2.709	**4.202**	1.43	2.053
	ss	-0.033	-0.039	-0.044	-0.041	**-0.028**	-0.041	-0.043
	umass	-14.794	-6.608	-6.362	-8.84	-6.904	**-4.305**	-8.941
	uci	-9.571	-3.858	**-2.863**	-3.829	-2.994	-3.521	-5.366
K=20	*npmi*	-0.342	-0.136	-0.095	-0.125	**-0.083**	-0.129	-0.191
Reuters	*qs*	0.127	0.209	0.201	0.204	**0.238**	0.156	0.184
	chs	0.989	2.337	2.355	2.204	**2.714**	1.729	2.347
	ss	**-0.029**	-0.038	-0.037	-0.03	-0.033	-0.037	-0.037
	umass	-18.802	**-7.578**	-8.838	-8.436	-8.714	-13.319	-11.139
	uci	-11.572	**-3.438**	-4.16	-4.248	-4.469	-7.346	-5.364
K=20	*npmi*	-0.415	-0.103	-0.128	**-0.1**	-0.105	-0.258	-0.174

4.6 Ablation study

As a final analysis, we conducted an ablation study. To be specific, we removed the proposed generative network and explored the model with only the remaining part. In the first conducted test, we replaced the generative network with a deterministic function that returned a Gaussian random distribution[13] (*AutoEncoder with Gaussian Randomness*). Whereas in the second test, we completely removed the randomness generation part and we considered only the compression network (*AutoEncoder with No Randomness*). In Table 5, it is possible to observe the average results obtained with the ablation study in three different simulations[14]. As can be observed, introducing a source of randomness into the process improves the robustness of topic modeling. In addition, albeit slightly, it appears that adaptive randomness generation via a neural network improves overall performance.

5 CONCLUSIONS

Generative Adversarial Networks (GANs) and Variational AutoEncoders (VAEs) have been recently considered a promising solution for the complex untrained problem of topic modeling from texts. However, opposing agents in GANs may cause problems of convergence, and VAEs may tend to overfit. In this paper we propose a collaborative architecture, Generative Cooperative Topic Modeling (GTCM), where a Generator and an AutoEncoder work cooperatively instead of opposing each other, leading to faster convergence and state-of-the-art performance. In the experimental Section, we have shown that GTCM achieves better performance or at least state-of-the-art performance on 3 datasets, six cluster internal validity measures, and a variable number of topics, when compared with the five most recent GANs and VAEs systems presented in literature. We have also shown that GTCM obtains close-to-convergence results already after the first epoch, contrary to the other approaches that require many epochs before convergence, and have been shown

[13]The mean and standard deviation of the Gaussian random distribution were calculated at each batch of data.
[14]Again, for reasons of space, the K parameter is set to 20.

Table 5: Ablation study results

		GCTM-b	AutoEnc (gauss-rand)	AutoEnc (no-rand)
Twenty	*qs*	**0.293**	0.279	0.288
news	*chs*	**3.575**	3.536	3.493
groups	*ss*	-0.014	-0.013	**-0.008**
	umass	-7.201	**-7.161**	-7.26
	uci	**-4.211**	-4.375	-4.468
K=20	*npmi*	**-0.092**	-0.103	-0.105
IMDB	*qs*	**0.256**	0.251	0.239
reviews	*chs*	**2.712**	2.605	2.423
	ss	**-0.042**	-0.046	-0.047
	umass	**-8.88**	-9.365	-9.264
	uci	**-3.869**	-4.523	-4.305
K=20	*npmi*	**-0.124**	-0.14	-0.142
Reuters	*qs*	**0.208**	0.202	0.2
	chs	**2.223**	2.15	2.126
	ss	**-0.028**	-0.03	-0.033
	umass	-8.436	**-8.315**	-8.503
	uci	**-4.251**	-4.306	-4.365
K=20	*npmi*	**-0.1**	-0.103	-0.107

to occasionally incur in problems such as non convergence, vanishing gradient, overfitting.

A GCTM TECHNICAL DETAILS

In this section, we report some implementation details of the proposed approach.

A.1 Cooperative network Architecture and Hyper-parameters

For the GCTM model, we selected the neural architecture, the hyper-parameters and the training processes that are schematically shown underneath.

A.1.1 Generator. Starting from an input random vector Z, it produces as output a synthetic document $G(Z)$.

- *input*: dense layer with $|Z|$[15] neurons, "relu" activation function, "l1" regularization (weights matrix and activation function) and 20% dropout rate;
- *hidden1*: dense layer with 256 neurons, "relu" activation function, "l1" regularization (weights matrix and activation function) and 30% dropout rate;
- *output*: dense layer with $|V|$[16] neurons, "relu" activation function, "l1" regularization (weights matrix and activation function), 40% dropout rate and kernel norm constraint in (0, 1).

A.1.2 AutoEncoder. Firstly, starting from an input document s, it encodes this document to a latent space. Finally, starting from the obtained latent space, it decodes and returns as output the reconstructed version of the input document $A(s)$.

- *encoder input*: dense layer with $|V|$ neurons, "relu" activation function, "l1" regularization (weights matrix and activation function) and 40% dropout rate;
- *encoder hidden1*: dense layer with 256 neurons, "relu" activation function, "l1" regularization (weights matrix and activation function) and 10% dropout rate;
- *encoder output*: dense layer with K[17] neurons, "relu" activation function and "l1" regularization (weights matrix and activation function);
- *decoder input*: dense layer with K neurons, "relu" activation function and "l1" regularization (weights matrix and activation function);
- *decoder hidden1*: dense layer with 256 neurons, "relu" activation function, "l1" regularization (weights matrix and activation function) and 10% dropout rate;
- *decoder output*: dense layer with $|V|$ neurons, "relu" activation function, "l1" regularization (weights matrix and activation function), 40% dropout rate and kernel norm constraint in (0, 1).

A.1.3 Training procedure. The two networks are trained simultaneously for 5 epochs, considering a batch size of 64 documents. Generator and AutoEncoder were optimized using two separate instances of Adam with a learning rate respectively of 0.001 and 0.005. Moreover, for both the optimizers, we set β_1 value to 0.5. The β_1 value was suggested by Yu et al. [29], and, similarly to what reported by the authors, seems to stabilize the training phase in our model.

A.2 Hierarchical clustering

Hierarchical clustering consists in the decomposition of the data based on group similarities. There are two types of hierarchical clustering approaches: agglomerative and divisive. In GCTM we used an agglomerative approach, since empirically it appeared to be a more robust and stable approach. The clustering hyper-parameters that we used were the following:

- *n_clusters*: we set this value with the number of desired topics K;
- *linkage* (linkage criterion that determines which distance to use between sets of observation): "ward";
- *affinity* (metric used to compute the linkage): "euclidean".

A.3 Embeddings Optimization

We calculated the embeddings on the input data via Word2Vec and the following hyper-parameters:

- *vector_size*: 100,
- *alpha*: 0.025,
- *window*: 5,
- *min_count*: $corpus\_size * 0.0005$,
- *min_alpha*: 0.00001,
- *sg*: 0,
- *hs*: 0,
- *negative*: 5,
- *epochs*: 100.

[15]$|Z|$ represents the size of the input vector with uniform random noise. We have set |Z| to 128.
[16]$|V|$ represents the size of the vocabulary (number of features).

[17]K represents the selected number of topics that we want to extract.

Moreover, we empirically set the two values m and $|S|$ respectively to the values 50 (n) and 150 ($n * 3$).

B TEXT PREPROCESSING

For the preprocessing of the raw texts, we performed the following steps:

- word tokenization;
- filtering by character typology (we filtered tokens that not presented all alphabetical character);
- filtering by token size (we discarded tokens with less than three characters and more than eighteen characters);
- filtering by Part Of Speech (POS) tagging[18] (we solely considered tokens labeled as nouns and proper nouns);
- lemmatization[19];
- conversion to lowercase.

C COMPUTING INFRASTRUCTURE AND EXECUTION TIME OF THE EXPERIMENTS

All the experiments were performed on a desktop workstation with the following hardware and software features:

- CPU: Intel Core i7-10700
- GPU: Nvidia GeForce RTX 2060
- RAM: 32 GB DDR4
- OS: Ubuntu 20.04 (Linux)

For each individual experiment, the computation required less than ten minutes to complete.

REFERENCES

[1] Federico Bianchi, Silvia Terragni, and Dirk Hovy. 2021. Pre-training is a Hot Topic: Contextualized Document Embeddings Improve Topic Coherence. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Online, 759–766. https://doi.org/10.18653/v1/2021.acl-short.96

[2] Federico Bianchi, Silvia Terragni, Dirk Hovy, Debora Nozza, and Elisabetta Fersini. 2021. Cross-lingual Contextualized Topic Models with Zero-shot Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, Online, 1676–1683. https://www.aclweb.org/anthology/2021.eacl-main.143

[3] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python* (1st ed.). O'Reilly Media, Inc., USA.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3, null (March 2003), 993–1022.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146. https://doi.org/10.1162/tacl_a_00051

[6] Sophie Burkhardt and Stefan Kramer. 2019. A Survey of Multi-Label Topic Models. *SIGKDD Explor. Newsl.* 21, 2 (Nov. 2019), 61–79. https://doi.org/10.1145/3373464.3373474

[7] T. Caliński and J. Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics* 3, 1 (1974), 1–27. https://doi.org/10.1080/03610927408827101

[8] Stephen Carrow. 2018. PyTorchAVITM: Open Source AVITM Implementation in PyTorch. Github.

[9] Alexey Dosovitskiy and Thomas Brox. 2016. Generating Images with Perceptual Similarity Metrics based on Deep Networks. arXiv:cs.LG/1602.02644

[10] John Glover. 2016. Modeling documents with Generative Adversarial Networks. *ArXiv* abs/1612.09122 (2016).

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. 2014. Generative Adversarial Networks. *Advances in Neural Information Processing Systems* 3 (06 2014). https://doi.org/10.1145/3422622

[12] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. 2002. Cluster Validity Methods: Part I. *SIGMOD Rec.* 31, 2 (June 2002), 40–45. https://doi.org/10.1145/565117.565124

[13] Xuemeng Hu, Rui Wang, Deyu Zhou, and Yuxuan Xiong. 2020. Neural Topic Modeling with Cycle-Consistent Adversarial Training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 9018–9030. https://doi.org/10.18653/v1/2020.emnlp-main.725

[14] S. Likhitha, B. S. Harish, and H. M. Keerthi Kumar. 2019. A Detailed Survey on Topic Modeling for Document and Short Text Data. *International Journal of Computer Applications* 178, 39 (Aug 2019), 1–9. https://doi.org/10.5120/ijca2019919265

[15] Kar Wai Lim and Wray L. Buntine. 2016. Bibliographic Analysis with the Citation Network Topic Model. *CoRR* abs/1609.06826 (2016). arXiv:1609.06826 http://arxiv.org/abs/1609.06826

[16] Lin Liu, Lin Tang, Wen Dong, Shaowen Yao, and Wei Zhou. 2016. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus* 5 (09 2016). https://doi.org/10.1186/s40064-016-3252-8

[17] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 142–150. http://www.aclweb.org/anthology/P11-1015

[18] Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural Variational Inference for Text Processing. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, 1727–1736.

[19] 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 (2011), 2825–2830.

[20] Peter Rousseeuw. 1987. Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Comput. Appl. Math. 20, 53-65. *J. Comput. Appl. Math.* 20 (11 1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

[21] Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the Space of Topic Coherence Measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15)*. Association for Computing Machinery, New York, NY, USA, 399–408. https://doi.org/10.1145/2684822.2685324

[22] Divya Saxena and Jiannong Cao. 2020. Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions. *CoRR* abs/2005.00065 (2020). arXiv:2005.00065 https://arxiv.org/abs/2005.00065

[23] P. Shamna, V.K. Govindan, and K.A. Abdul Nazeer. 2019. Content based medical image retrieval using topic and location model. *Journal of Biomedical Informatics* 91 (2019), 103112. https://doi.org/10.1016/j.jbi.2019.103112

[24] Akash Srivastava and Charles Sutton. 2017. Autoencoding Variational Inference For Topic Models. *arXiv e-prints*, Article arXiv:1703.01488 (March 2017), arXiv:1703.01488 pages. arXiv:stat.ML/1703.01488

[25] Rui Wang, Xuemeng Hu, Deyu Zhou, Yulan He, Yuxuan Xiong, Chenchen Ye, and Haiyang Xu. 2020. Neural Topic Modeling with Bidirectional Adversarial Training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 340–350. https://doi.org/10.18653/v1/2020.acl-main.32

[26] Rui Wang, Deyu Zhou, and Yulan He. 2019. ATM: Adversarial-neural Topic Model. *Information Processing & Management* 56, 6 (2019), 102098. https://doi.org/10.1016/j.ipm.2019.102098

[27] Joe H. Ward. 1963. Hierarchical Grouping to Optimize an Objective Function. *J. Amer. Statist. Assoc.* 58, 301 (1963), 236–244. https://doi.org/10.1080/01621459.1963.10500845

[28] Maciej Wiatrak, Stefano V. Albrecht, and Andrew Nystrom. 2020. Stabilizing Generative Adversarial Networks: A Survey. arXiv:cs.LG/1910.00927

[29] Yang Yu, Zhiqiang Gong, Ping Zhong, and Jiaxin Shan. 2017. Unsupervised Representation Learning with Deep Convolutional Neural Network for Remote Sensing Images. In *Image and Graphics*, Yao Zhao, Xiangwei Kong, and David Taubman (Eds.). Springer International Publishing, Cham, 97–108.

[30] Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. 2016. Energy-based Generative Adversarial Network. *ArXiv* abs/1609.03126 (2016).

[18]We performed POS tagging through the specific pre-trained English model of the "*spaCy*" library.

[19]We performed lemmatization through pre-trained English model of the "*spaCy*" library.