Pre-training and Fine-tuning Neural Topic Model: A Simple yet Effective Approach to Incorporating External Knowledge

Anonymous ACL submission

Abstract

Recent years have witnessed growing interests in incorporating external knowledge such as pre-trained word embeddings (PWEs) or pre-trained language models (PLMs) into neural topic modeling. However, we found that employing PWEs and PLMs for topic modeling only achieved limited performance improvements but with huge computational overhead. In this paper, we propose a novel strategy to incorporate external knowledge into neural topic modeling where the neural topic model is pre-trained on a large corpus and then fine-tuned on the target dataset. Experiments have been conducted on three datasets and results show that the proposed approach significantly outperforms both current state-of-the-art neural topic models and some topic modeling approaches enhanced with PWEs or PLMs. Moreover, further study shows that the proposed approach greatly reduces the need for the huge size of training data.

1 Introduction

Topic models have been widely used for discovering hidden themes from a large collection of documents in an unsupervised manner. Recently, to avoid the complex and specific inference process of graph model-based method such as LDA (Blei et al., 2003), neural topic modeling that utilizes neural-network-based black-box inference has been the main research direction in this field (Blei, 2012; Miao et al., 2016; Srivastava and Sutton, 2017). Typically, neural topic models infer topics of a document by utilizing its bag-of-words (BoWs) representation to capture word co-occurrence patterns. The BoWs representation, however, fails to encode rich word semantics, leading to relatively inferior quality of topics generated by the topic models. Therefore, approaches have been proposed to address the limitation of BoWs representation by incorporating the external knowledge, such as pre-trained word embeddings (PWEs) (Das et al., 2015; Wang et al., 2020; Dieng et al., 2020).

In recent years, pre-trained language models (PLMs) (Peters et al., 2018; Devlin et al., 2019; Brown et al., 2020) have achieved state-of-the-art performance on a wide range of natural language processing tasks. Different from PWEs\(^1\) in which a word is mapped to a static word embedding, PLMs generate a specific word embedding for each occurrence of a word depending on the context. It is appealing to incorporate PLMs into topic models since contextualized embeddings generated by PLMs encode richer semantics and naturally deal with word polysemy (Pasini et al., 2020). One straightforward way is to replace BoWs representation with the outputs of PLM (Bianchi et al., 2020b) in existing topic models or take PLM outputs as additional inputs to topic modeling (Bianchi et al., 2020a). A more sophisticated approach is to distill the knowledge of a PLM into a topic model. For example, (Hoyle et al., 2020) employed the probability estimates of a teacher PLM over a text sequence to guide the training of a student topic model.

However, the approaches mentioned above still have limitations. Firstly, using PLMs for topic model training in such ways leads to huge computational overhead. Most neural topic models are based on shallow multi-layer perceptions with few hidden units. However, most popular PLMs are based on deep Transformers (Vaswani et al., 2017) where at each layer expensive self-attention operations are performed, which have a time complexity quadratic in document length. Therefore, the overall training time is dominated by PLM, and it will be worse if PLM is further fine-tuned, as shown in (Hoyle et al., 2020). Secondly, there is the gap of training objectives between PLMs and topic models, where PLMs are trained to learn the semantic and syntactic knowledge within a sentence while PLMs are trained to learn the semantic and syntactic knowledge within a sentence while

\(^1\)In this paper, PWEs refer to context-free embeddings.
topic models focus on extracting main themes over whole corpus. As shown in Table 4, a model based on GloVe embeddings (Pennington et al., 2014) performs better than PLMs-based models such as those proposed in (Bianchi et al., 2020a) and (Bianchi et al., 2020b).

To overcome these challenges, we propose a simple yet effective strategy, namely Pre-trained Neural Topic Model (PT-NTM), to utilize extensive knowledge from large corpora for neural topic modeling with low computational complexity. Instead of pre-training the embeddings and acquiring knowledge indirectly, PT-NTM directly pre-trains the topic model itself on the knowledge source corpora. In specific, a neural topic model is firstly trained on a large corpus only once, which is called pre-training. Afterward, it is fine-tuned on any other dataset, which is called fine-tuning. As the architecture of the neural topic model used in pre-training and fine-tuning is the same, it incurs little computational overhead to any subsequent training. Experiments have been conducted on three datasets and the results show that the proposed approach significantly outperforms not only some state-of-the-art neural topic models but also the topic modeling approaches using PWEs and PLMs. Moreover, it is observed that on the NYTimes dataset, the neural topic model trained on 1% of the whole dataset using the proposed approach achieves superior performance than other baseline models that are trained on the whole dataset. It further shows that the proposed approach greatly reduces the need for the huge size of training data.

The main contributions are:

- We proposed a simple yet effective strategy for training neural topic models in which the models are pre-trained on a large corpus and then fine-tuned on a specific dataset.
- We conducted extensive experiments and the results show that the pre-trained neural topic models significantly outperform baselines in terms of topic coherence and topic diversity.
- The proposed approach greatly reduces the amount of training data needed. In our experiments on the NYTimes dataset, a pre-trained model fine-tuned with 1% of documents achieves superior performance than baselines that are trained on the whole dataset.

2 Related Work

2.1 Neural Topic Modeling

Due to the flexible modeling choices and high representation capacity, neural networks have been widely used for topic modeling in recent years. Some approaches (Kingma and Welling, 2013; Miao et al., 2016) model topics with variational autoencoders (VAEs) and view the latent variables of VAEs as document topics. However, topic models typically use Dirichlet distribution as the prior of multinomial topic distributions, while the reparameterization trick required by VAEs hinders the usage of a Dirichlet prior. Therefore, some follow-up works (Srivastava and Sutton, 2017; Card et al., 2018) used logistic normal to approximate Dirichlet. Another family of neural topic models (Nan et al., 2019; Wang et al., 2020; Hu et al., 2020) overcome the problem with adversarial training (Goodfellow et al., 2014) by encouraging the model to generate topic distributions that are similar to samples randomly drawn from a Dirichlet prior.

2.2 Topic Modeling with External Knowledge

There are mainly two ways to incorporate external knowledge into topic modeling, namely by PWEs and PLMs.

Some attempts incorporate pre-trained word representations into neural topic models. For example, (Card et al., 2018; Dieng et al., 2020) used PWEs to initialize word embeddings of topic models. (Wang et al., 2020) built a generative process that models word embeddings with per-topic Gaussian distributions.

Beyond static word embeddings, researchers also tried to utilize PLMs. (Bianchi et al., 2020b,a) treated PLM outputs as an additional knowledge source to enhance or replace BoW-based inputs. (Hoyle et al., 2020) employed knowledge distillation to guide the training of a student topic model with a PLM teacher network. Recently, (Song et al., 2020) proposed TopicOcean to train LDA-based topic models on large corpora and then transfer the knowledge of accumulated topics to new corpora which can also be considered a way of pre-training.

It should be pointed out that the proposed PT-NTM differs from the previous PLMs-based topic models or TopicOcean in that the architecture of neural topic models during pre-training and fine-tuning are the same in PT-NTM while other methods combine the large PLM with the topic models, the two different model architectures.
3 Methodology

In this section, we describe the detailed processes of PT-NTM. First, we will introduce the architecture of neural topic model, which we call NTM in the following, employed in PT-NTM. Then, we will introduce how to pre-train the neural topic model on a large-scale dataset. Finally, we will introduce how to fine-tune the pre-trained neural topic model on the target dataset.

3.1 Neural Topic Model Architecture

For the architecture of NTM, we follow the encoder-decoder architecture, as employed by many neural topic models (Srivastava and Sutton, 2017; Miao et al., 2017; Nan et al., 2019). The encoder takes a document’s BoW $x \in \mathbb{R}^V$ as input and infers its topic distribution $\hat{z} \in \mathbb{R}^K$, where $V$ is the vocabulary size and $K$ the topic number. The decoder then reconstructs the original document from $\hat{z}$, denoted as $\hat{x}$.

The whole architecture of NTM is shown in Figure 1. In specific, the encoder is a stack of $N + 1$ MLP layers. From the bottom to the top, the first $N$ layers have an identical structure. Each layer has four sub-layers: Dropout (Srivastava et al., 2014), Linear, BatchNorm (Ioffe and Szegedy, 2015), and LeakyReLU (Maas et al., 2013). The final layer is a Dropout sub-layer and a Linear transformation followed by a Softmax. The decoder shares the same architecture as the encoder, though they may vary in input/output dimensions. In our experiments, we set a Dropout probability of 0.5 in the first encoder layer and 0.2 in the remaining encoder and decoder layers. All LeakyReLU sub-layers have a negative slope of 0.01.

Combining the encoder and the decoder, we now have the reconstruction loss:

$$\mathcal{L}_{\text{rec}}(X, \hat{X}) = -\mathbb{E}(x \log \hat{x}), \quad (1)$$

which encourages the decoder outputs $\hat{X} = \{\hat{x}^{(i)}\}_{i=1}^m$ to be as similar as the corresponding encoder inputs $X = \{x^{(i)}\}_{i=1}^m$ for each training batch, where $m$ is the batch size.

For topic distribution $\hat{z}$, what we have done above is insufficient to generate reasonable topics since $\hat{z}$’s distribution $Q$ is not well defined. To this end, we follow a similar approach proposed in (Nan et al., 2019) and further impose on $\hat{z}$ a Dirichlet prior $P$ by minimizing the Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) between the two distributions $P$ and $Q$:

$$\mathcal{L}_{\text{MMD}}(Z, \hat{Z}) = -\frac{2}{m^2} \sum_{i,j} k(z^{(i)}, \hat{z}^{(j)}) + \frac{1}{m(m-1)} \sum_{i\neq j} (k(z^{(i)}, \hat{z}^{(j)}) + k(\hat{z}^{(i)}, \hat{z}^{(j)})), \quad (2)$$

where $Z = \{z^{(i)}\}_{i=1}^m$ are topic distributions randomly drawn from the prior $P$, $\hat{Z} = \{\hat{z}^{(i)}\}_{i=1}^m$ are encoder outputs, and $k$ is the kernel function that is information diffusion kernel (Lebanon and Lafferty, 2003) in our experiments following (Nan et al., 2019).

The overall training objective is:

$$\mathcal{L} = \mathcal{L}_{\text{rec}}(X, \hat{X}) + \lambda r \mathcal{L}_{\text{MMD}}(Z, \hat{Z}), \quad (3)$$

where we balance $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{MMD}}$ with a hyperparameter $\lambda$ and another factor

$$r = \frac{\|\nabla_b^{(N+1)} \mathcal{L}_{\text{rec}}(X, \hat{X})\|_2}{\|\nabla_b^{(N+1)} \mathcal{L}_{\text{MMD}}(Z, \hat{Z})\|_2}, \quad (4)$$

where $\|\cdot\|_2$ denotes L2 normalization and $b^{(N+1)}$ is the bias term of the last Linear sub-layer of the encoder, i.e., the one just before the Softmax sub-layer. Equation (4) shows that the two losses are balanced with their relative gradient norm with respect to $b^{(N+1)}$. We found in our experiments that $r$ greatly reduces the effort of tuning $\lambda$ and generally produces better results.
3.2 Pre-training

By pre-training the topic model on a large and topically diverse corpus, we expect the model to learn topic-related knowledge that is general enough to be reused on other corpora. For the proposed approach, the knowledge may include word semantics, common senses, and document encoding and decoding patterns at each layer.

The details of the pre-training procedure are presented in Algorithm 1. The pre-training corpus $D$ is the subset00 of the OpenWebText dataset (Gokaslan and Cohen, 2019), an open-source recreation of the WebText dataset as detailed in (Radford et al., 2019). We preprocess data by tokenization, lemmatization, stopword removal, and only keeping words occurred in at least 50 documents. After preprocessing, there are about 392K documents, consisting of 45K unique words, in the resulting dataset. At each training mini-batch, we update model parameters according to Equation (3) using the Adam optimizer (Kingma and Ba, 2014).

Algorithm 1 Pre-training.

Require: $D$, the pre-training corpus; $E$, the encoder; $D$, the decoder; $\theta$, parameters of $E$ and $D$; $\theta_b$, initial parameters; $m$, the batch size; $n$, the number of training epochs; $P(z)$, the Dirichlet prior.
1: $\theta \leftarrow \theta_0$
2: for $i = 1, \ldots, n$ do
3: Shuffle $D$.
4: for each $X = \{x^{(j)}\}_{j=1}^m$ from $D$ do
5: $Z \leftarrow E(X)$; $\hat{X} \leftarrow D(Z)$
6: Sample $Z = \{z^{(j)}\}_{j=1}^m$ ~ $P(z)$.
7: Compute $\mathcal{L}$ by Equation (3).
8: $\theta \leftarrow \text{Adam}(\nabla \theta, \frac{1}{m} \sum_{j=1}^{m} \mathcal{L}^{(j)}, \theta_p)$
9: end for
10: end for

3.3 Fine-tuning

Fine-tuning is the process of adapting the pre-trained topic model to a specific dataset. However, directly fine-tuning the pre-trained model on a new dataset does not always work and may introduce severe bias to subsequent tuning steps since the ideal number of topics might change and the corpus-wide topic distributions might be different. Therefore, our fine-tuning begins with the pre-trained model but randomly re-initializes parameters in the last encoder layer and the first decoder layer. If we fine-tune the model without any re-initialization, we find that in our experiments the corpus-wide topic distributions discovered by the fine-tuned model would be biased towards the topic distribution of the pre-training corpus, which is unexpected. The proposed fine-tuning strategy with re-initialization solves this issue. Algorithm 2 shows the fine-tuning steps. We keep the pre-trained parameters fixed for the first $n_1$ epochs and use a small learning rate in the remaining training epochs since they have already been well trained before fine-tuning.

Algorithm 2 Fine-tuning.

Require: $D'$, the target corpus; $E$, the encoder; $D$, the decoder; $\theta_r$, randomly initialized parameters; $\theta_p$, pre-trained parameters; $m$, the batch size; $n$, the number of training epochs; $n_1$, $n_1 \in \mathbb{N}$ and $0 \leq n_1 \leq n$; $P(z)$, the Dirichlet prior.
1: for $i = 1, \ldots, n$ do
2: Shuffle $D'$.
3: for each $X = \{x^{(j)}\}_{j=1}^m$ from $D'$ do
4: $\hat{Z} \leftarrow E(X)$; $\hat{X} \leftarrow D(\hat{Z})$
5: Sample $Z = \{z^{(j)}\}_{j=1}^m$ ~ $P(z)$.
6: Compute $\mathcal{L}$ by Equation (3).
7: $\theta_r \leftarrow \text{Adam}(\nabla \theta_r, \frac{1}{m} \sum_{j=1}^{m} \mathcal{L}^{(j)}(\theta_r))$
8: if $i > n_1$ then
9: $\theta_p \leftarrow \text{Adam}(\nabla \theta_p, \frac{1}{m} \sum_{j=1}^{m} \mathcal{L}^{(j)}(\theta_p))$
10: end if
11: end for
12: end for

By comparing Algorithm 1 with Algorithm 2, it can be observed that the fine-tuning process adds little overhead to the training stage. More importantly, the proposed method does not introduce any additional computations or parameters during inference.

4 Experiments

We used three datasets in (Hu et al., 2020): NYTimes2, Grolier3, and 20Newsgroups4. We did not include the DBpedia dataset as it is based on Wikipedia and potentially overlaps with the dataset used for our pre-training. The dataset statistics are shown in Table 1.

The proposed basic model, NTM, is the one described in Section 3 without pre-training. Both the

2http://archive.ics.uci.edu/ml/datasets/Bag+of+Words
3https://cs.nyu.edu/~roweis/data
4http://qwone.com/~jason/20Newsgroups
encoder and the decoder have three layers ($N = 2$) and 300 neurons at each hidden layer. We have four variants:

- **NTM-w2v**, we initialize weights $w_{e1} \in \mathbb{R}^{V \times 300}$ of the first encoder Linear sub-layer and $w_{d3} \in \mathbb{R}^{300 \times V}$ of the the last decoder Linear sub-layer with the corresponding 300-dim Word2Vec embeddings trained on Google News.

- **NTM-glv**, same as NTM-w2v but utilizing 300-dim GloVe embeddings trained on Wikipedia and Gigaword 5.

- **PT-NTM-w2v**, pre-training from NTM-w2v initialization and then fine-tuning.

- **PT-NTM-glv**, pre-training from NTM-glv initialization and then fine-tuning.

The number of training epochs is 200 for pre-training, fine-tuning (PT-* models) and fresh training (NTM). We used the Dirichlet prior distribution whose parameters are all $\frac{1}{K}$, where $K$ is the topic number. MMD loss weight $\lambda$ is 1 for all models expect the fine-tuning of *-pre models in which $\lambda$ is 0.3. We will analyze the effect of $\lambda$ in our experiments. During pre-training, the batch size is 1,024, the learning rate is 2e-2, and the topic number is 200. For fine-tuning, $n_1$ is 100, and the learning rates for reinitialized and pre-trained parameters are 2e-2 and 1e-5, respectively (Algorithm 2), showing that the pre-trained parameters are only slightly tuned. The batch size of fine-tuning and fresh training varies on different datasets depending on their sizes. Specifically, it is set to 128 for 20Newsgroups, 256 for Grolier and 512 for NYT Times. Finally, it should be noted that fine-tuning on each datasets shares the same pre-trained model checkpoint for each model variant.

We compare our models with following baselines:

- **LDA** (Blei et al., 2003), we used the implementation of GibbsLDA++\(^5\).

- **ProdLDA** (Srivastava and Sutton, 2017), a VAE-based model that employs logistic normal prior for topic distributions.

- **W-LDA** (Nan et al., 2019). Our model follows W-LDA loss but differs in training and implementation.

- **BAT** (Wang et al., 2020), an adversarially trained neural topic model.

- **ToMCAT** (Hu et al., 2020), an adversarial neural topic model with cycle-consistency objective.

- **ZeroShotTM** (Bianchi et al., 2020b), taking Sentence-BERT (Reimers and Gurevych, 2019) embeddings as input.

- **CombinedTM** (Bianchi et al., 2020a), same as ZeroShotTM but combining the input with BoWs.

- **G-BAT** (Wang et al., 2020), extending BAT to incorporate pre-trained word embeddings.

- **TopicOcean** (Song et al., 2020), integrating well-trained LDAs and transferring the knowledge of accumulated topics to new corpora, which is re-implemented by ourselves.

We evaluate the model performance with three topic coherence measures and one topic diversity measure. Topic coherence measures first calculate the coherence scores of pairs of top words ranked by their topic-associated probabilities for each topic and then aggregate all topic scores as the final topic coherence. The used topic coherence measures are $C_A$ (Aletras and Stevenson, 2013), $C_P$ (Röder et al., 2015), and NPMI (Aletras and Stevenson, 2013) of top-10 topic words, implemented in Palmetto (Röder et al., 2015)\(^6\). Topic coherence measures are highly correlated with human evaluation but have no penalizing mechanism for repetitive or similar topics. We remedy the problem by also evaluating topic diversity. Our topic diversity measure is calculate by $\text{TD} = 1 - \frac{N_{\text{rep}}}{N_{\text{total}}}$, where $N_{\text{total}} = 10 \times K$ is the total number of topic words and $N_{\text{rep}}$ counts the number of repetitions in all topic words. For example, 5 identical words would add 4 to $N_{\text{rep}}$.

\(^5\)http://gibbslda.sourceforge.net/
\(^6\)https://github.com/AKSW/Palmetto

| Dataset  | #Documents | Vocabulary Size |
|----------|------------|----------------|
| NYTimes  | 99,992     | 12,604         |
| Grolier  | 29,762     | 15,276         |
| 20Newsgroups | 11,258 | 2,000          |

Table 1: Dataset statistics.
We report results averaged over five runs with topic worse than regular methods. We think the reason the topic modeling results are presented in Table 2.

| Model         | NYTimes C_A | NYTimes C_P | NYTimes NPMI | NYTimes TD | Grolier C_A | Grolier C_P | Grolier NPMI | Grolier TD | 20Newsgroups C_A | 20Newsgroups C_P | 20Newsgroups NPMI | 20Newsgroups TD |
|---------------|-------------|-------------|--------------|------------|-------------|-------------|--------------|------------|------------------|-----------------|------------------|----------------|
| BoWs-based    |             |             |              |            |             |             |              |            |                  |                 |                  |                |
| LDA           | 0.215       | 0.323       | 0.081        | 0.82       | 0.196       | 0.197       | 0.053        | 0.81       | 0.186            | 0.282           | 0.064            | 0.79            |
| ProdLDA       | 0.184       | 0.125       | 0.015        | 0.69       | 0.148       | -0.065      | -0.019       | 0.83       | 0.178            | 0.071           | -0.044           | 0.67            |
| W-LDA         | 0.225       | 0.335       | 0.078        | 0.79       | 0.235       | 0.258       | 0.073        | 0.86       | 0.229            | 0.341           | 0.062            | 0.72            |
| BAT           | 0.236       | 0.375       | 0.095        | 0.80       | 0.211       | 0.231       | 0.061        | 0.73       | 0.199            | 0.296           | 0.055            | 0.69            |
| ToMCAT        | 0.245       | 0.385       | 0.095        | 0.79       | 0.229       | 0.275       | 0.081        | 0.90       | 0.208            | 0.314           | 0.066            | 0.68            |
| NTM           | 0.229       | 0.269       | 0.056        | 0.90       | 0.215       | 0.146       | 0.030        | 0.93       | 0.242            | 0.372           | 0.070            | 0.82            |
| PWEs-based    |             |             |              |            |             |             |              |            |                  |                 |                  |                |
| G-BAT         | 0.249       | 0.414       | 0.108        | 0.72       | 0.219       | 0.258       | 0.074        | 0.78       | 0.229            | 0.394           | 0.087            | 0.78            |
| NTM-w2v       | 0.238       | 0.404       | 0.096        | 0.93       | 0.236       | 0.273       | 0.087        | 0.92       | 0.258            | 0.482           | 0.113            | 0.82            |
| NTM-glv       | 0.247       | 0.388       | 0.103        | 0.90       | 0.257       | 0.334       | 0.106        | 0.93       | 0.278            | 0.526           | 0.129            | 0.80            |
| PLMs-based    |             |             |              |            |             |             |              |            |                  |                 |                  |                |
| ZeroShotTM    | 0.266       | 0.419       | 0.099        | 0.68       | 0.197       | 0.289       | 0.060        | 0.61       | 0.195            | 0.289           | 0.070            | 0.61            |
| CombinedTM    | -           | -           | -            | -          | -           | -           | -            | -          | -                | -               | -                | -               |
| Pretrain-based|             |             |              |            |             |             |              |            |                  |                 |                  |                |
| TopicOcean    | 0.312       | 0.651       | 0.148        | 0.91       | 0.325       | 0.616       | 0.127        | 0.93       | 0.279            | 0.532           | 0.124            | 0.80            |
| PT-NTM        | 0.276       | 0.539       | 0.131        | 0.96       | 0.325       | 0.621       | 0.160        | 0.95       | 0.271            | 0.538           | 0.127            | 0.87            |
| PT-NTM-w2v    | 0.304       | 0.614       | 0.152        | 0.95       | 0.345       | 0.673       | 0.181        | 0.96       | 0.287            | 0.560           | 0.140            | 0.84            |
| PT-NTM-glv    | 0.345       | 0.614       | 0.152        | 0.95       | 0.345       | 0.673       | 0.181        | 0.96       | 0.287            | 0.560           | 0.140            | 0.84            |

Table 2: Average topic coherence (C_A, C_P, and NPMI) and topic diversity (TD) scores of 5 topic number settings (20, 30, 50, 75, 100) on 3 datasets (NYTimes, Grolier, and 20Newsgroups). Bold values indicate best-performing models under corresponding settings. NYTimes and Grolier only have BoW data so we cannot evaluate ZeroShotTM and CombinedTM, which require word order information, on them.

4.1 Topic Modeling Results

The topic modeling results are presented in Table 2. We report results averaged over five runs with topic number set to 20, 30, 50, 75, and 100 respectively in all our experiments unless otherwise specified.

From Table 2, we can observe that: 1) Among all models, PT-NTM and its variants outperform other methods by a large margin. Since PT-NTM and NTM share the identical model architecture, we attribute the improvements of PT-NTM over NTM to the pre-training strategy. 2) For PLMs-based methods, both ZeroShotTM and CombinedTM performs badly, for some metric even worse than regular methods. We think the reason maybe the gap between the learning objectives of PLMs (word order-based) and topic models (word-cooccurrence based). 3) For PWEs-based methods, non-pretrained methods (NTM, BAT) benefits a lot from the PWEs. We think the reason maybe the PWEs are also trained based on word-cooccurrence, so the gap between PWEs and topic models is relatively small. Another interesting thing is that the benefit of using PWEs in topic modeling seems diminishing with our proposed topic model pre-training strategy. For example, PT-NTM gives similar results compared to PT-NTM-w2v and PT-NTM-glv. This shows that word semantic knowledge has somehow been captured to a certain degree by pre-training the topic model on a large corpus. 4) For pre-training-based models, PT-NTM outperforms TopicOcean, consider the performance gap between their base models (NTM for PT-NTM and LDA for TopicOcean), the improvement of PT-NTM is even larger. What’s more, our method is based on neural network, which is easier to incorporated with PWEs or other information than TopicOcean, which is based on graphical models.

One concern about PT-NTM may be that the whether the fine-tuning stage works. To get a sense of the topics extracted by our model, we list in Table 3 top 4 topics extracted by PT-NTM on the pre-training and fine-tuning dataset. The topic labels are assigned manually. The whole topics are presented in the attachment.

4.2 Contextualized vs. Static word embeddings

Contextualized word embeddings like those produced by BERT (Devlin et al., 2019) provide richer semantic than static ones like Word2Vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014). Thus we also conducted experiments to test their performance on topic modeling. The baseline models are ZeroShotTM (Bianchi et al., 2020b) and CombinedTM (Bianchi et al., 2020a). ZeroShotTM and CombinedTM both take SentenceBERT (Reimers and Gurevych, 2019) embeddings as inputs but CombinedTM additionally uses BoW. We also implement three NTM-based models, namely BERT-NTM, Word2Vec-NTM, and GloVeNTM, according to the input embeddings they used. BERT-NTM follows the idea of ZeroShotTM, aim-
Table 3: Top 4 topics extracted by PT-NTM on OpenWebText, NYTimes, Grolier and 20Newsgroups dataset.

Table 4: Topic modeling results on 20Newsgroups.

Table 5: The impact of the #layers on 20Newsgroups.
PT-NTM-glv. PT-NTM also has a similar trend but with more drastic changes. Given these findings, it seems that there is a trade-off towards generating more coherent or diverse topics.

Nevertheless, it is worth noting that in comparison to NTM, the PT-NTM-glv is very robust to the choices of $\lambda$. The NPMI values of PT-NTM-glv only fluctuate in the range of [0.11, 0.14] while its TD values vary between 0.74 and 0.86. This is in contrast to NTM in which it has poor topic coherence for $\lambda \leq 0.1$ and low topic diversity for $\lambda \geq 10$. We attribute the advantage of the pre-trained model to our proposed fine-tuning strategy. During fine-tuning, we mainly update a small set of parameters that are directly related to topics while only slightly tune others, which consequently enables more controllable data/gradient flows and thus produces more stable results.

**Data efficiency** With pre-training, a topic model indeed captures extensive knowledge from an external corpus. As have been shown in our experiments, the acquired knowledge can improve the performance of subsequent fine-tuning on other datasets. It would be interesting to see to what extent such knowledge can increase data efficiency. To this end, we conducted experiments that take subsets of NYTimes dataset of varying sizes as training datasets. Specifically, we used dataset sizes including 1K, 2K, 4K, · · · , 64K, and 100K. For each size, we averaged the results over five runs whose training datasets are randomly sampled from the whole dataset with different random seeds.

The results are shown in Figure 3. PT-NTM-glv has a very high starting point when the document number is 1000: the NPMI and TD is about 0.15 and 0.89 respectively. While at the same time, NTM has extremely poor performance with negative NPMI and low TD. Only when the document number increases to 8000, the topics generated by NTM has comparable topic diversity to topics from PT-NTM-glv. But even when the whole dataset is used by PT-NTM, i.e., the document number is 100K, NTM’s NPMI is still about 0.08 lower than the 1000-document PT-NTM-glv, which indeed represents a significant difference in topic quality. In summary, pre-training the topic model greatly reduces the need for training data and helps the model achieve superior performance with only 1% of documents on the NYTimes dataset.

5 Conclusion

In this paper, we proposed a simple yet effective strategy to incorporating external knowledge into neural topic modeling by pre-training topic models on a large corpus before fine-tuning them on specific datasets. By experiments, we have presented the effectiveness of the method of pre-trained neural topic model in terms of topic coherence, topic diversity, and data efficiency over other methods such as by incorporating PWEs and PLMs. Another advantage of this approach is that it introduces little overhead to the training and none to the inference. Limited by computing resources, we did not experiment pre-trainings on larger datasets, though we believe there is still room for improvement given more pre-training data. For future research, we encourage further explorations in model architectures, pre-training objectives, and fine-tuning procedures.
References

Nikolaos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers, pages 13–22, Potsdam, Germany. Association for Computational Linguistics.

Federico Bianchi, Silvia Terragni, and Dirk Hovy. 2020a. Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. arXiv preprint arXiv:2004.03974.

Federico Bianchi, Silvia Terragni, Dirk Hovy, Debora Nozza, and Elisabetta Fersini. 2020b. Cross-lingual contextualized topic models with zero-shot learning. arXiv preprint arXiv:2004.07737.

David M. Blei. 2012. Probabilistic topic models. Communications of the ACM, 55(4):77.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Dallas Card, Chenhao Tan, and Noah A. Smith. 2018. Neural models for documents with metadata. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2031–2040, Melbourne, Australia. Association for Computational Linguistics.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 795–804, Beijing, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. 2020. Topic modeling in embedding spaces. Transactions of the Association for Computational Linguistics, 8:439–453.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Advances in Neural Information Processing Systems 27, pages 2672–2680. Curran Associates, Inc.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. J. Mach. Learn. Res., 13:723–773.

Alexander Miserlis Hoyle, Pranav Goel, and Philip Resnik. 2020. Improving Neural Topic Models using Knowledge Distillation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1752–1771, Online. Association for Computational Linguistics.

Nikolaos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers, pages 13–22, Potsdam, Germany. Association for Computational Linguistics.

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), pages 9018–9030, Online. Association for Computational Linguistics.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning, volume 37, pages 448–456, Lille, France. PMLR.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Guy Lebanon and John D. Lafferty. 2003. Information diffusion kernels. In S. Becker, S. Thrun, and K. Obermayer, editors, Advances in Neural Information Processing Systems 15, pages 391–398. MIT Press.

Andrew L. Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In ICML Workshop on Deep Learning for Audio, Speech and Language Processing. Citeseer.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance Kaplan, and Jiawei Han. 2019. Spherical text embedding. In Advances in Neural Information Processing Systems, volume 32, pages 8208–8217. Curran Associates, Inc.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 2410–2419, International Convention Centre, Sydney, Australia. PMLR.
