TAN-NTM: Topic Attention Networks for Neural Topic Modeling Supplementary

Madhur Panwar[2*†], **Shashank Shailabh**[3*†], **Milan Aggarwal**[1*], **Balaji Krishnamurthy**[1]
Media and Data Science Research Labs, Adobe[1]
Birla Institute of Technology and Science, Pilani (BITS Pilani), India[2]
Indian Institute of Technology Kanpur (IIT Kanpur), India[3]
mdrpanwar@gmail.com, shailabhshashank@gmail.com

1 Further Implementation Details

1.1 Preprocessing

For 20NG dataset, we used its preprocessed version downloaded from ProdLDA's (Srivastava and Sutton, 2017) repository[1], whereas AGNews and YRP datasets were downloaded from this[2] link. These two datasets contain **train.csv** and **test.csv** files. The csv files of YRP contain a document body only, whereas the csv files for AGNews contain a document title as well as a document body. For uniformity, we concatenate the title and body in the csv files of AGNews and keep it as a single field. The documents from **train.csv** and **test.csv** are then read into `train` and `test` lists which are passed to PREPROCESS function of Algorithm 1 for preprocessing.

Stepwise working of Algorithm 1 is expained in the following points:

- Before invoking the PREPROCESS function, we initialize the data sampler by a fixed seed so that preprocessing yields the same result when run multiple times.

- For each dataset, we randomly sample `tr_size` documents (as mentioned in Table 1) from the `train` list in step 2. These values of `tr_size` are taken from Table 1 of W-LDA paper (Nan et al., 2019). Note that **# Train** in Table 1 of the main paper represents the number of training documents after preprocessing. Of the `tr_size` documents, some documents may be removed during preprocessing, therefore **# Train** may be less than `tr_size`.

- In steps 3 through 8, we prune the `train` and `test` documents by invoking the PRUNE_DOC function from Algorithm 2. First, we remove the control characters from the documents viz. '\n', '\t', and '\r' (For YRP, we additionally remove **'\\t'**, **'\\n'**, and **'\\r'**). Next, we remove the numeric tokens[3] from the documents, convert them to lowercase and lemmatize each of their tokens using the NLTK's (Bird et al., 2009) WordNetLemmatizer. Finally, we remove punctuations[4] and tokens containing any non-ASCII character.

- In steps 9 through 15, we construct the vocabulary `vocab`, which is a mapping of each token to its occurrence count among the pruned training documents `tr_pruned`. We only count a token if it is not an English stopword[5] and its length is between 3 and 15 (inclusive).

- Steps 16 through 19 filter the `vocab` by removing tokens whose total occurrences are less than `num_below` or whose occurrence count per training document is greater than `fr_abv`, where the values of `num_below` and `fr_abv` are taken from Table 1. For YRP, we follow the W-LDA paper (Nan et al., 2019) and restrict its `vocab` to only contain top $20,000$ most occurring tokens.

- Steps 20 through 24 construct the token-to-index map `w2idx` by mapping each token in `vocab` to an index starting from 1. Next, we map the padding token to index 0 (Step 25).

[*] equal contribution
[†] work done during summer internship at Adobe
[1] Data link for 20NG dataset
[2] Data link for AGNews and YRP datasets

[3] Fully numeric tokens e.g. '1487', '1947', etc. are removed, whereas partially numeric tokens e.g. 'G47', 'DE1080', etc. are retained.
[4] Any of the following 32 characters is regarded as a punctuation !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
[5] Gensim's (Řehůřek and Sojka, 2010) list of English stopwords is used.

- The final step in the preprocessing is to encode the train and test documents by mapping each of their tokens to corresponding indices according to `w2idx`. This is done by the ENCODE function of Algorithm 2 which is invoked in steps 26 and 27.

Algorithm 1 Pseudocode for preprocessing AGNews and YRP datasets.

1: **function** PREPROCESS(`train`, `test`)
2: `train` $\leftarrow$ `train`.*sample*(`tr_size`)
3: `tr_pruned` $\leftarrow$ [] ▷ empty list
4: `te_pruned` $\leftarrow$ [] ▷ empty list
5: **for** document d **in** `train` **do**
6: `tr_pruned`.*append*(PRUNE_DOC(d))
7: **for** document d **in** `test` **do**
8: `te_pruned`.*append*(PRUNE_DOC(d))
9: `vocab` $\leftarrow$ mapping of each token to 0
10: `num_doc` $\leftarrow$ *len*(`tr_pruned`)
11: **for** document d **in** `tr_pruned` **do**
12: **for** token t **in** d **do**
13: **if** $t \notin$ *stopwords* **and**
14: $len(t) \in [3, 15]$ **then**
15: `vocab`[t]$\leftarrow$ `vocab`[t] $+1$
16: **for** token t **in** `vocab` **do**
17: **if** `vocab`[t] $<$ `num_below` **or**
18: `vocab`[t]/`num_doc` $>$ `fr_abv` **then**
19: `vocab`[t].*remove*(t)
20: $i \leftarrow 1$
21: `w2idx` $\leftarrow$ empty map
22: **for** token t **in** `vocab` **do**
23: `w2idx`[t]$= i$
24: $i \leftarrow i + 1$
25: `w2idx`[0]$\leftarrow \mathbb{PAD}$
26: `trD` $\leftarrow$ ENCODE(`tr_pruned`, `w2idx`)
27: `teD` $\leftarrow$ ENCODE(`te_pruned`, `w2idx`)
28: **return** `trD`, `teD`, `w2idx`

1.2 Learning Rate Scheduler

As mentioned in section 5.2 of the main paper, we use a learning rate scheduler while training T-TAN.

Algorithm 2 Pseudocode for pruning the document and encoding it given a token-to-index mapping.

function PRUNE_DOC(`doc`)
  `doc` $\leftarrow rm\_control$(`doc`)
  `doc` $\leftarrow rm\_numeric$(`doc`)
  `doc` $\leftarrow lowercase$(`doc`)
  `doc` $\leftarrow lemmatize$(`doc`)
  `doc` $\leftarrow rm\_punctuations$(`doc`)
  `doc` $\leftarrow rm\_non\_ASCII$(`doc`)
  return `doc`

function ENCODE(`doc_list`, `w2idx`)
  `encDocList` $\leftarrow$ []
  for document d **in** `doc_list` **do**
    `ecDoc` $\leftarrow$ []
    for token t in d **do**
      `ecDoc`.*append*(`w2idx`[t])
    `encDocList`.*append*(`ecDoc`)
  return `encDocList`

The rate decay follows the following equation:

$$lrate = init\_rate * decay\_rate^{\left\lfloor \frac{train\_step}{decay\_steps} \right\rfloor}$$

This is an exponential staircase function which enables decrease in learning rate every epoch during training. We initialize the learning rate by $init\_rate = 0.002$ & use $decay\_rate = 0.96$. $train\_step$ is a global counter of training steps & $decay\_steps = \frac{\#train\_docs}{batch\_size}$ is the number of training steps taken per epoch. Therefore, effectively, the rate remains constant for all training steps in an epoch and decreases exponentially as per the above equation once the epoch completes.

Dataset	`tr_size`	`num_below`	`fr_abv`
AGNews	96000	3	0.7
YRP	448000	20	0.7

Table 1: Parameters used for preprocessing the AGNews and YRP datasets.

1.3 Regularization

We employ two types of regularization during training:

- **Dropout**: We apply dropout (Srivastava et al., 2014) to z with the rate of $P_{drop} = 0.6$ before it is processed by the decoder for reconstruction.

- **Batch Normalization (BN)**: We apply a BN (Ioffe and Szegedy, 2015) to the inputs of decoder layer and to the inputs of layers being trained for z_μ & $z_{\log \sigma^2}$, with $\epsilon = 0.001$ and $decay = 0.999$.

2 Evaluation Metrics

Topic models have been evaluated using various metrics namely perplexity, topic coherence, topic uniqueness etc. However, due to the absence of a gold standard for the unsupervised task of topic modeling, all of that metrics have received criticism by the community. Therefore, a consensus on the best metric has not been reached so far. Perplexity has been found to be negatively correlated to topic quality and human judgements (Chang et al., 2009). This work presents experimental results which show that in some cases models with higher perplexity were preferred by human subjects.

Topic Uniqueness (Nan et al., 2019) quantifies the intersection among topic words globally. However, it also suffers from drawbacks and often penalizes a model incorrectly (Hoyle et al., 2020). Firstly, it does not account for ranking of intersected words in the topics. Secondly, it fails to distinguish between the following two scenarios: **1)** When the intersected words in one topic are all present in a second topic (signifying strong similarity i.e. these two topics are essentially identical) and, **2)** When the intersected words of one topic are spread across all the other topics (signifying weak similarity i.e. the topics are diffused). The first is a problem related to uniqueness among topics while second is a problem related to word intrusion in topics. (Chang et al., 2009) conducted experiments with human subjects on two tasks: word intrusion and topic intrusion. Word intrusion measures the presence of those words (called intruder words) which disagree with the semantics of the topic. Topic intrusion measures the presence of those topics (called intruder topics) which do not represent the document corpus appropriately. These are better estimates of human judgement of topic models in comparison to perplexity and uniqueness. However, since these metrics rely on human feedback, they cannot be widely used for unsupervised evaluation. Further, topic uniqueness unfairly penalizes cases when some words are common between topics, however other uncommon words in those topics change the context as well as topic semantics as also discussed in (Hoyle et al., 2020).

Would a pilot know that one of their crew is armed? The Federal Flight Deck Officer page on Wikipedia says this: Under the FFDO program, flight crew members are authorized to use firearms. A flight crew member may be a pilot, flight engineer or navigator assigned to the flight. To me, it seems like this would be crucial information for the PIC to know, if their flight engineer (for example) was armed; but on the flip-side of this, the engineer might want to keep that to himself if he's with a crew he hasn't flown with before. Is there a guideline on whether an FFDO should inform the crew that he's armed? **GT:** *security, crew, ffdo* **TAKG:** *faa regulations, ffdo, flight training, firearms, far* **TAKG + W-TAN:** *ffdo, crew, flight controls, crewed spaceflight, security*
Do the poisons in "Ode on Melancholy" have deeper meaning? In "Ode on Melancholy", Keats uses the images of three poisons in the first stanza: Wolf's bane, nightshade, and yew-berries. Are these poisons simply meant to connote death/suicide, or might they have a deeper purpose? **GT:** *poetry, meaning, john keats* **TAKG:** *the keats, meaning, poetry, ode, melancholy keats* **TAKG + W-TAN:** *poetry, meaning, the keats, john keats, greek literature*

Table 2: Two randomly selected posts (title in **bold**) from StackExchange dataset with ground truth (**GT**) and top 5 keyphrases predicted by TAKG with and without W-TAN, denoted as **TAKG + W-TAN** & **TAKG** respectively. Keyphrases generated with W-TAN are closer to the ground truth in terms of both prediction and ranking.

According to the work of (Lau et al., 2014), measuring the normalized pointwise mutual information (NPMI) between all the word pairs in a set of topics agrees with human judgements most closely. This is called the NPMI Topic Coherence in the litera-

ture and is widely used for the evaluation of topic models. We therefore adopt this metric in our work. Since the effectiveness of a topic model actually depends on the topic representations that it extracts from the documents, we report the performance of our model on two downstream tasks: document classification and keyphrase generation (which use these topic representations) for a better and holistic evaluation and comparison.

3 Qualitative Analysis

3.1 Key Phrase Predictions

We saw the quantitative improvement in results in Table 5 of the main paper when we used W-TAN as the topic model with TAKG. In Table 2, we display some posts from StackExchange dataset with ground truth keyphrases and top 5 predictions by TAKG with and without W-TAN. We observe that using W-TAN improves keyphrase generation qualitatively.

The first post in Table 2 inquires if a flight officer should inform the pilot in command (PIC) about him being armed or not. For this post, TAKG alone only predicts one ground truth keyphrase correctly and misses *'security'* and *'crew'*. However, when TAKG is used with W-TAN, it gets all three ground truth keyphrases, two of which are its top 2 predictions as well.

The second post is inquiring about a possible deeper meaning of three poisons in a poem by John Keats. TAKG alone predicts two of the ground truth keyphrases correctly but assigns them larger ranks and it misses *'john keats'*. When TAKG is used with W-TAN, it gets all three ground truth keyphrases and its top 2 keyphrases are assigned the exact same rank as they have in the ground truth. This hints that using W-TAN with TAKG improves the prediction as well as ranking of the generated keyphrases compared to using TAKG alone.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-graber, and David Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, volume 22, pages 288–296. Curran Associates, Inc.

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.

S. Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539, Gothenburg, Sweden. Association for Computational Linguistics.

Feng Nan, Ran Ding, Ramesh Nallapati, and Bing Xiang. 2019. Topic modeling with Wasserstein autoencoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6345–6381, Florence, Italy. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488*.

Nitish Srivastava, Geoffrey E. Hinton, A. Krizhevsky, Ilya Sutskever, and R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958.