Topically Driven Neural Language Model

Jey Han Lau[1,2] **Timothy Baldwin**[2] **Trevor Cohn**[2]

[1] IBM Research

[2] School of Computing and Information Systems,
The University of Melbourne

jeyhan.lau@gmail.com, tb@ldwin.net, t.cohn@unimelb.edu.au

Abstract

Language models are typically applied at the sentence level, without access to the broader document context. We present a neural language model that incorporates document context in the form of a topic model-like architecture, thus providing a succinct representation of the broader document context outside of the current sentence. Experiments over a range of datasets demonstrate that our model outperforms a pure sentence-based model in terms of language model perplexity, and leads to topics that are potentially more coherent than those produced by a standard LDA topic model. Our model also has the ability to generate related sentences for a topic, providing another way to interpret topics.

1 Introduction

Topic models provide a powerful tool for extracting the macro-level content structure of a document collection in the form of the latent topics (usually in the form of multinomial distributions over terms), with a plethora of applications in NLP (Hall et al., 2008; Newman et al., 2010a; Wang and McCallum, 2006). A myriad of variants of the classical LDA method (Blei et al., 2003) have been proposed, including recent work on neural topic models (Cao et al., 2015; Wan et al., 2012; Larochelle and Lauly, 2012; Hinton and Salakhutdinov, 2009).

Separately, language models have long been a foundational component of any NLP task involving generation or textual normalisation of a noisy input (including speech, OCR and the processing of social media text). The primary purpose of a language model is to predict the probability of a span of text, traditionally at the sentence level, under the assumption that sentences are independent of one another, although recent work has started using broader local context such as the preceding sentences (Wang and Cho, 2016; Ji et al., 2016).

In this paper, we combine the benefits of a topic model and language model in proposing a topically-driven language model, whereby we jointly learn topics and word sequence information. This allows us to both sensitise the predictions of the language model to the larger document narrative using topics, and to generate topics which are better sensitised to local context and are hence more coherent and interpretable.

Our model has two components: a language model and a topic model. We implement both components using neural networks, and train them jointly by treating each component as a sub-task in a multi-task learning setting. We show that our model is superior to other language models that leverage additional context, and that the generated topics are potentially more coherent than LDA topics. The architecture of the model provides an extra dimensionality of topic interpretability, in supporting the generation of sentences from a topic (or mix of topics). It is also highly flexible, in its ability to be supervised and incorporate side information, which we show to further improve language model performance. An open source implementation of our model is available at: https://github.com/jhlau/topically-driven-language-model.

2 Related Work

Griffiths et al. (2004) propose a model that learns topics and word dependencies using a Bayesian framework. Word generation is driven by either LDA or an HMM. For LDA, a word is generated based on a sampled topic in the document. For the

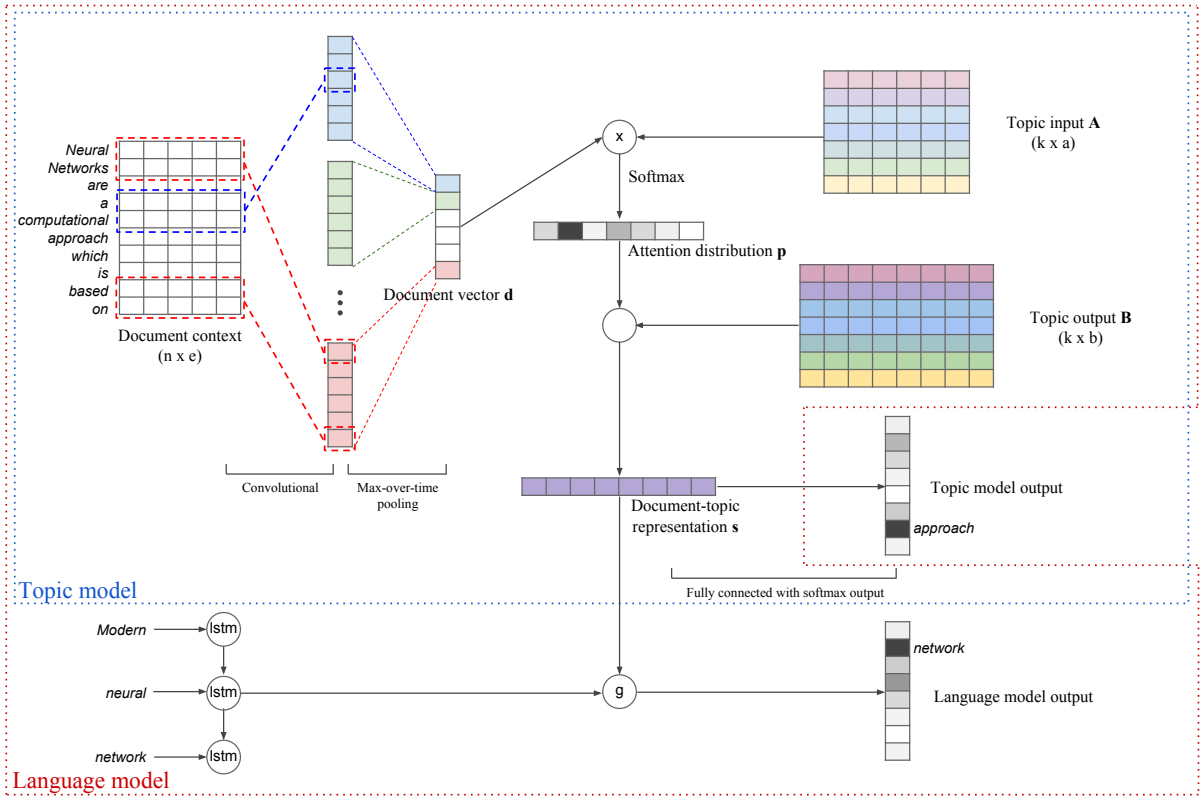


Figure 1: Architecture of `tdlm`. Scope of the models are denoted by dotted lines: blue line denotes the scope of the topic model, red the language model.

HMM, a word is conditioned on previous words. A key difference over our model is that their language model is driven by an HMM, which uses a fixed window and is therefore unable to track long-range dependencies.

Cao et al. (2015) relate the topic model view of documents and words — documents having a multinomial distribution over topics and topics having a multinomial distributional over words — from a neural network perspective by embedding these relationships in differentiable functions. With that, the model lost the stochasticity and Bayesian inference of LDA but gained non-linear complex representations. The authors further propose extensions to the model to do supervised learning where document labels are given.

Wang and Cho (2016) and Ji et al. (2016) relax the sentence independence assumption in language modelling, and use preceeding sentences as additional context. By treating words in preceeding sentences as a bag of words, Wang and Cho (2016) use an attentional mechanism to focus on these words when predicting the next word. The authors show that the incorporation of additional context helps language models.

3 Architecture

The architecture of the proposed topically-driven language model (henceforth "`tdlm`") is illustrated in Figure 1. There are two components in `tdlm`: a language model and a topic model. The language model is designed to capture word relations in sentences, while the topic model learns topical information in documents. The topic model works like an auto-encoder, where it is given the document words as input and optimised to predict them.

The topic model takes in word embeddings of a document and generates a document vector using a convolutional network. Given the document vector, we associate it with the topics via an attention scheme to compute a weighted mean of topic vectors, which is then used to predict a word in the document.

The language model is a standard LSTM language model (Hochreiter and Schmidhuber, 1997; Mikolov et al., 2010), but it incorporates the weighted topic vector generated by the topic model to predict succeeding words.

Marrying the language and topic models allows the language model to be topically driven, i.e. it models not just word contexts but also the document context where the sentence occurs, in the form of topics.

3.1 Topic Model Component

Let $\mathbf{x}_i \in \mathbb{R}^e$ be the e-dimensional word vector for the i-th word in the document. A document of n words is represented as a concatenation of its word vectors:

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus ... \oplus \mathbf{x}_n$$

where $\oplus$ denotes the concatenation operator. We use a number of convolutional filters to process the word vectors, but for clarity we will explain the network with one filter.

Let $\mathbf{w}_v \in \mathbb{R}^{eh}$ be a convolutional filter which we apply to a window of h words to generate a feature. A feature c_i for a window of words $\mathbf{x}_{i:i+h-1}$ is given as follows:

$$c_i = I(\mathbf{w}_v^\intercal \mathbf{x}_{i:i+h-1} + b_v)$$

where b_v is a bias term and I is the identity function.[1] A feature map $\mathbf{c}$ is a collection of features computed from all windows of words:

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}]$$

where $\mathbf{c} \in \mathbb{R}^{n-h+1}$. To capture the most salient features in $\mathbf{c}$, we apply a max-over-time pooling operation (Collobert et al., 2011), yielding a scalar:

$$d = \max_i c_i$$

In the case where we use a filters, we have $\mathbf{d} \in \mathbb{R}^a$, and this constitutes the vector representation of the document generated by the convolutional and max-over-time pooling network.

The topic vectors are stored in two lookup tables $\mathbf{A} \in \mathbb{R}^{k \times a}$ (input vector) and $\mathbf{B} \in \mathbb{R}^{k \times b}$ (output vector), where k is the number of topics, and a and b are the dimensions of the topic vectors.

To align the document vector $\mathbf{d}$ with the topics, we compute an attention vector which is used to compute a document-topic representation:[2]

$$\mathbf{p} = \text{softmax}(\mathbf{A}\mathbf{d}) \quad (1)$$

$$\mathbf{s} = \mathbf{B}^\intercal \mathbf{p} \quad (2)$$

where $\mathbf{p} \in \mathbb{R}^k$ and $\mathbf{s} \in \mathbb{R}^b$. Intuitively, $\mathbf{s}$ is a weighted mean of topic vectors, with the weighting given by the attention $\mathbf{p}$. This is inspired by the generative process of LDA, whereby documents are defined as having a multinomial distribution over topics.

Finally $\mathbf{s}$ is connected to a dense layer with softmax output to predict each word in the document, where each word is generated independently as a unigram bag-of-words, and the model is optimised using categorical cross-entropy loss. In practice, to improve efficiency we compute loss for predicting a sequence of m_1 words in the document, where m_1 is a hyper-parameter.

3.2 Language Model Component

The language model is implemented using LSTM units (Hochreiter and Schmidhuber, 1997):

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{v}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{v}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{v}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\hat{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \mathbf{v}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where $\odot$ denotes element-wise product; $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the input, forget and output activations respectively at time step t; and $\mathbf{v}_t$, $\mathbf{h}_t$ and $\mathbf{c}_t$ are the input word embedding, LSTM hidden state, and cell state, respectively. Hereinafter $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$ are used to refer to the model parameters.

Traditionally, a language model operates at the sentence level, predicting the next word given its history of words in the sentence. The language model of `tdlm` incorporates topical information by assimilating the document-topic representation ($\mathbf{s}$) with the hidden output of the LSTM ($\mathbf{h}_t$) at each time step t. To prevent `tdlm` from memorising the next word via the topic model network, we exclude the current sentence from the document context.

[1] A non-linear function is typically used here, but preliminary experiments suggest that the identity function works best for `tdlm`.

[2] The attention mechanism was inspired by memory networks (Graves et al., 2014; Weston et al., 2014; Sukhbaatar et al., 2015; Tran et al., 2016). We explored various attention styles (including traditional schemes which use one vector for a topic), but found this approach to work best.

We use a gating unit similar to a GRU (Cho et al., 2014; Chung et al., 2014) to allow tdlm to learn the degree of influence of topical information on the language model:

$$\begin{aligned}\mathbf{z}_t &= \sigma(\mathbf{W}_z\mathbf{s} + \mathbf{U}_z\mathbf{h}_t + \mathbf{b}_z)\\ \mathbf{r}_t &= \sigma(\mathbf{W}_r\mathbf{s} + \mathbf{U}_r\mathbf{h}_t + \mathbf{b}_r)\\ \hat{\mathbf{h}}_t &= \tanh(\mathbf{W}_h\mathbf{s} + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_t) + \mathbf{b}_h)\\ \mathbf{h}'_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_t + \mathbf{z}_t \odot \hat{\mathbf{h}}_t\end{aligned} \quad (3)$$

where $\mathbf{z}_t$ and $\mathbf{r}_t$ are the update and reset gate activations respectively at timestep t. The new hidden state $\mathbf{h}'_t$ is connected to a dense layer with linear transformation and softmax output to predict the next word, and the model is optimised using standard categorical cross-entropy loss.

3.3 Training and Regularisation

tdlm is trained using minibatches and SGD.[3] For the language model, a minibatch consists of a batch of sentences, while for the topic model it is a batch of documents (each predicting a sequence of m_1 words).

We treat the language and topic models as subtasks in a multi-task learning setting, and train them jointly using categorical cross-entropy loss. Most parameters in the topic model are shared by the language model, as illustrated by their scopes (dotted lines) in Figure 1.

Hyper-parameters of tdlm are detailed in Table 1. Word embeddings for the topic model and language model components are not shared, although their dimensions are the same (e).[4] For m_1, m_2 and m_3, sequences/documents shorter than these thresholds are padded. Sentences longer than m_2 are broken into multiple sequences, and documents longer than m_3 are truncated. Optimal hyper-parameter settings are tuned using the development set; the presented values are used for experiments in Sections 4 and 5.

To regularise tdlm, we use dropout regularisation (Srivastava et al., 2014). We apply dropout to $\mathbf{d}$ and $\mathbf{s}$ in the topic model, and to the input word embedding and hidden output of the LSTM in the language model (Pham et al., 2013; Zaremba et al., 2014).

4 Language Model Evaluation

We use standard language model perplexity as the evaluation metric. In terms of dataset, we use document collections from 3 sources: APNEWS, IMDB and BNC. APNEWS is a collection of Associated Press[5] news articles from 2009 to 2016. IMDB is a set of movie reviews collected by Maas et al. (2011). BNC is the written portion of the British National Corpus (BNC Consortium, 2007), which contains excerpts from journals, books, letters, essays, memoranda, news and other types of text. For APNEWS and BNC, we randomly sub-sample a set of documents for our experiments.

For preprocessing, we tokenise words and sentences using Stanford CoreNLP (Klein and Manning, 2003). We lowercase all word tokens, filter word types that occur less than 10 times, and exclude the top 0.1% most frequent word types.[6] We additionally remove stopwords for the topic model document context.[7] All datasets are partitioned into training, development and test sets; preprocessed dataset statistics are presented in Table 2.

We tune hyper-parameters of tdlm based on development set language model perplexity. In general, we find that optimal settings are fairly robust across collections, with the exception of m_3, as document length is collection dependent; optimal hyper-parameter values are given in Table 1. In terms of LSTM size, we explore 2 settings: a small model with 1 LSTM layer and 600 hidden units, and a large model with 2 layers and 900 hidden units.[8] For the topic number, we experiment with 50, 100 and 150 topics. Word embeddings are pre-trained 300-dimension word2vec Google News vectors.[9]

For comparison, we compare tdlm with:[10]

vanilla-lstm: A standard LSTM language model, using the same tdlm hyper-parameters where applicable. This is the baseline model.

lclm: A larger context language model that incorporates context from preceding sentences (Wang and Cho, 2016), by treating the preceding sentence as a bag of words, and using an

[3] We use Adam as the optimiser (Kingma and Ba, 2014).

[4] Word embeddings are updated during training.

[5] https://www.ap.org/en-gb/.

[6] For the topic model, we remove word tokens that correspond to these filtered word types; for the language model we represent them as ⟨unk⟩ tokens (as for unseen words in test).

[7] We use Mallet's stopword list: https://github.com/mimno/Mallet/tree/master/stoplists.

[8] Multi-layer LSTMs are vanilla stacked LSTMs without skip connections (Gers and Schmidhuber, 2000) or depth-gating (Yao et al., 2015).

[9] https://code.google.com/archive/p/word2vec/.

[10] Note that all models use the same pre-trained word2vec vectors.

Hyper-parameter	Value	Description
m_1	3	Output sequence length for topic model
m_2	30	Sequence length for language model
m_3	300,150,500	Maximum document length
n_{batch}	64	Minibatch size
n_{layer}	1,2	Number of LSTM layers
n_{hidden}	600,900	LSTM hidden size
n_{epoch}	10	Number of training epochs
k	100,150,200	Number of topics
e	300	Word embedding size
h	2	Convolutional filter width
a	20	Topic input vector size or number of features for convolutional filter
b	50	Topic output vector size
l	0.001	Learning rate of optimiser
p_1	0.4	Topic model dropout keep probability
p_2	0.6	Language model dropout keep probability

Table 1: `tdlm` hyper-parameters; we experiment with 2 LSTM settings and 3 topic numbers, and m_3 varies across the three domains (APNEWS, IMDB, and BNC).

Collection	Training		Development		Test	
	#Docs	#Tokens	#Docs	#Tokens	#Docs	#Tokens
APNEWS	50K	15M	2K	0.6M	2K	0.6M
IMDB	75K	20M	12.5K	0.3M	12.5K	0.3M
BNC	15K	18M	1K	1M	1K	1M

Table 2: Preprocessed dataset statistics.

attentional mechanism when predicting the next word. An additional hyper-parameter in `lclm` is the number of preceeding sentences to incorporate, which we tune based on a development set (to 4 sentences in each case). All other hyper-parameters (such as n_{batch}, e, n_{epoch}, k_2) are the same as `tdlm`.

`lstm+lda`: A standard LSTM language model that incorporates LDA topic information. We first train an LDA model (Blei et al., 2003; Griffiths and Steyvers, 2004) to learn 50/100/150 topics for APNEWS, IMDB and BNC.[11] For a document, the LSTM incorporates the LDA topic distribution ($\mathbf{q}$) by concatenating it with the output hidden state ($\mathbf{h}_t$) to predict the next word (i.e. $\mathbf{h}'_t = \mathbf{h}_t \oplus \mathbf{q}$). That is, it incorporates topical information into the language model, but unlike `tdlm` the language model and topic model are trained separately.

We present language model perplexity performance in Table 3. All models outperform the baseline `vanilla-lstm`, with `tdlm` performing the best across all collections. `lclm` is competitive over the BNC, although the superiority of `tdlm` for the other collections is substantial. `lstm+lda` performs relatively well over APNEWS and IMDB, but very poorly over BNC.

The strong performance of `tdlm` over `lclm` suggests that compressing document context into topics benefits language modelling more than using extra context words directly.[12] Overall, our results show that topical information can help language modelling and that joint inference of topic and language model produces the best results.

5 Topic Model Evaluation

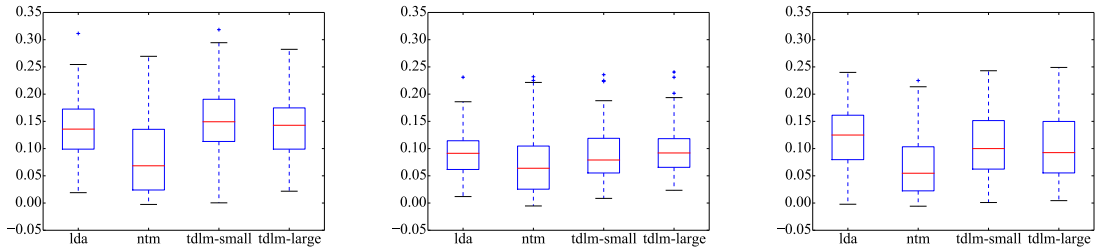
We saw that `tdlm` performs well as a language model, but it is also a topic model, and like LDA it produces: (1) a probability distribution over topics for each document (Equation (1)); and (2) a probability distribution over word types for each topic.

[11] Based on Gibbs sampling; $\alpha = 0.1$, $\beta = 0.01$.

[12] The context size of `lclm` (4 sentences) is technically smaller than `tdlm` (full document), however, note that increasing the context size does not benefit `lclm`, as the context size of 4 gives the best performance.

Domain	LSTM Size	vanilla-lstm	lclm	lstm+lda			tdlm		
				50	100	150	50	100	150
APNEWS	small	64.13	54.18	57.05	55.52	54.83	53.00	52.75	**52.65**
	large	58.89	50.63	52.72	50.75	50.17	48.96	48.97	**48.21**
IMDB	small	72.14	67.78	69.58	69.64	69.62	63.67	**63.45**	63.82
	large	66.47	67.86	63.48	63.04	62.78	58.99	59.04	**58.59**
BNC	small	102.89	87.47	96.42	96.50	96.38	87.42	**85.99**	86.43
	large	94.23	80.68	88.42	87.77	87.28	82.62	81.83	**80.58**

Table 3: Language model perplexity performance of all models over APNEWS, IMDB and BNC. Boldface indicates best performance in each row.



(a) APNEWS (b) IMDB (c) BNC

Figure 2: Boxplots of topic coherence of all models; number of topics = 100.

Recall that $\mathbf{s}$ is a weighted mean of topic vectors for a document (Equation (2)). Generating the vocabulary distribution for a particular topic is therefore trivial: we can do so by treating $\mathbf{s}$ as having maximum weight (1.0) for the topic of interest, and no weight (0.0) for all other topics. Let $\mathbf{B}_t$ denote the topic output vector for the t-th topic. To generate the multinomial distribution over word types for the t-th topic, we replace $\mathbf{s}$ with $\mathbf{B}_t$ before computing the softmax over the vocabulary.

Topic models are traditionally evaluated using model perplexity. There are various ways to estimate test perplexity (Wallach et al., 2009), but Chang et al. (2009) show that perplexity does not correlate with the coherence of the generated topics. Newman et al. (2010b); Mimno et al. (2011); Aletras and Stevenson (2013) propose automatic approaches to computing topic coherence, and Lau et al. (2014) summarises these methods to understand their differences. We propose using automatic topic coherence as a means to evaluate the topic model aspect of `tdlm`.

Following Lau et al. (2014), we compute topic coherence using normalised PMI ("NPMI") scores. Given the top-n words of a topic, coherence is computed based on the sum of pairwise NPMI scores between topic words, where the word probabilities used in the NPMI calculation are based on co-occurrence statistics mined from English Wikipedia with a sliding window (Newman et al., 2010b; Lau et al., 2014).[13]

Based on the findings of Lau and Baldwin (2016), we average topic coherence over the top-5/10/15/20 topic words. To aggregate topic coherence scores for a model, we calculate the mean coherence over topics.

In terms of datasets, we use the same document collections (APNEWS, IMDB and BNC) as the language model experiments (Section 4). We use the same hyper-parameter settings for `tdlm` and do not tune them.

For comparison, we use the following topic models:

`lda`: We use a LDA model as a baseline topic model. We use the same LDA models as were used to learn topic distributions for `lstm+lda` (Section 4).

[13] We use this toolkit to compute topic coherence: https://github.com/jhlau/topic_interpretability.

Topic No.	System	Coherence		
		APNEWS	IMDB	BNC
50	lda	.125	.084	**.106**
	ntm	.075	.064	.081
	tdlm-small	**.149**	**.104**	.102
	tdlm-large	.130	.088	.095
100	lda	.136	.092	**.119**
	ntm	.085	.071	.070
	tdlm-small	**.152**	.087	.106
	tdlm-large	.142	**.097**	.101
150	lda	.134	**.094**	**.119**
	ntm	.078	.075	.072
	tdlm-small	**.147**	.085	.100
	tdlm-large	.145	.091	.104

Table 4: Mean topic coherence of all models over APNEWS, IMDB and BNC. Boldface indicates the best performance for each dataset and topic setting.

ntm: ntm is a neural topic model proposed by Cao et al. (2015). The document-topic and topic-word multinomials are expressed from a neural network perspective using differentiable functions. Model hyper-parameters are tuned using development loss.

Topic model performance is presented in Table 4. There are two models of tdlm (tdlm-small and tdlm-large), which specify the size of its LSTM model (1 layer+600 hidden vs. 2 layers+900 hidden; see Section 4). tdlm achieves encouraging results: it has the best performance over APNEWS, and is competitive over IMDB. lda, however, produces more coherent topics over BNC. Interestingly, coherence appears to increase as the topic number increases for lda, but the trend is less pronounced for tdlm. ntm performs the worst of the 3 topic models, and manual inspection reveals that topics are in general not very interpretable. Overall, the results suggest that tdlm topics are competitive: at best they are more coherent than lda topics, and at worst they are as good as lda topics.

To better understand the spread of coherence scores and impact of outliers, we present box plots for all models (number of topics = 100) over the 3 domains in Figure 2. Across all domains, ntm has poor performance and larger spread of scores. The difference between lda and tdlm is small (tdlm > lda in APNEWS, but lda < tdlm in BNC), which is consistent with our previous observation that tdlm topics are competitive with lda topics.

Partition	#Docs	#Tokens
Training	9314	2.6M
Development	2000	0.5M
Test	7532	1.7M

Table 5: 20NEWS preprocessed statistics.

6 Extensions

One strength of tdlm is its flexibility, owing to it taking the form of a neural network. To showcase this flexibility, we explore two simple extensions of tdlm, where we: (1) build a supervised model using document labels (Section 6.1); and (2) incorporate additional document metadata (Section 6.2).

6.1 Supervised Model

In datasets where document labels are known, supervised topic model extensions are designed to leverage the additional information to improve modelling quality. The supervised setting also has an additional advantage in that model evaluation is simpler, since models can be quantitatively assessed via classification accuracy.

To incorporate supervised document labels, we treat document classification as another sub-task in tdlm. Given a document and its label, we feed the document through the topic model network to generate the document vector $\mathbf{d}$ and document-topic representation $\mathbf{s}$, and then we concatenate both and connect it to another dense layer with softmax output to generate the probability distribution over classes.

During training, we have additional minibatches for the documents. We start the document classification training after the topic and language models have completed training in each epoch.

We use 20NEWS in this experiment, which is a popular dataset for text classification. 20NEWS is a collection of forum-like messages from 20 newsgroups categories. We use the "bydate" version of the dataset, where the train and test partition is separated by a specific date. We sample 2K documents from the training set to create the development set. For preprocessing we tokenise words and sentence using Stanford CoreNLP (Klein and Manning, 2003), and lowercase all words. As with previous experiments (Section 4) we additionally filter low/high frequency word types and stopwords. Preprocessed dataset statistics are presented in Table 5.

Topic No.	System	Accuracy
50	lda	.567
	ntm	**.649**
	tdlm	.606
100	lda	.581
	ntm	**.639**
	tdlm	.602
150	lda	.597
	ntm	**.628**
	tdlm	.601

Table 6: 20NEWS classification accuracy. All models are supervised extensions of the original models. Boldface indicates the best performance for each topic setting.

Topic No.	Metadata	Coherence	Perplexity
50	No	.128	52.45
	Yes	**.131**	**51.80**
100	No	**.142**	52.14
	Yes	.139	**51.76**
150	No	.135	52.25
	Yes	**.143**	**51.58**

Table 7: Topic coherence and language model perplexity by incorporating classification tags on APNEWS. Boldface indicates optimal coherence and perplexity performance for each topic setting.

For comparison, we use the same two topic models as in Section 5: ntm and lda. Both ntm and lda have natural supervised extensions (Cao et al., 2015; McAuliffe and Blei, 2008) for incorporating document labels. For this task, we tune the model hyper-parameters based on development accuracy.[14] Classification accuracy for all models is presented in Table 6. We present tdlm results using only the small setting of LSTM (1 layer + 600 hidden), as we found there is little gain when using a larger LSTM.

ntm performs very strongly, outperforming both lda and tdlm by a substantial margin. Comparing lda and tdlm, tdlm achieves better performance, especially when there is a smaller number of topics. Upon inspection of the topics we found that ntm topics are much less coherent than those of lda and tdlm, consistent with our observations from Section 5.

[14]Most hyper-parameter values for tdlm are similar to those used in the language and topic model experiments; the only exceptions are: $a = 80$, $b = 100$, $n_{epoch} = 20$, $m_3 = 150$. The increase in parameters is unsurprising, as the additional supervision provides more constraint to the model.

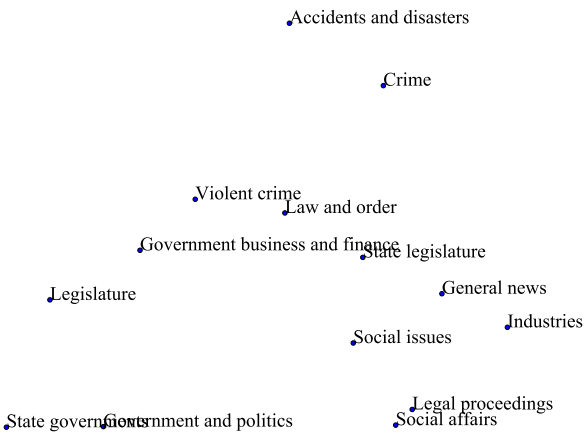


Figure 3: Scatter plots of tag embeddings (model=150 topics)

6.2 Incorporating Document Metadata

In APNEWS, each news article contains additional document metadata, including subject classification tags, such as "General News", "Accidents and Disasters", and "Military and Defense". We present an extension to incorporate document metadata in tdlm to demonstrate its flexibility in integrating this additional information.

As some of the documents in our original APNEWS sample were missing tags, we re-sampled a set of APNEWS articles of the same size as our original, all of which have tags. In total, approximately 1500 unique tags can be found among the training articles.

To incorporate these tags, we represent each of them as a learnable vector and concatenate it with the document vector before computing the attention distribution. Let $\mathbf{z}_i \in \mathbb{R}^f$ denote the f-dimension vector for the i-th tag. For the j-th document, we sum up all tags associated with it:

$$\mathbf{e} = \sum_{i=1}^{n_{tags}} \mathbb{I}(i,j)\mathbf{z}_i$$

where n_{tags} is the total number of unique tags, and function $\mathbb{I}(i,j)$ returns 1 is the i-th tag is in the j-th document or 0 otherwise. We compute $\mathbf{d}$ as before (Section 3.1), and concatenate it with the summed tag vector: $\mathbf{d}' = \mathbf{d} \oplus \mathbf{e}$.

We train two versions of tdlm on the new APNEWS dataset: (1) the vanilla version that ignores the tag information; and (2) the extended version which incorporates tag information.[15] We exper-

[15]Model hyper-parameters are the same as the ones used in the language (Section 4) and topic model (Section 5) experiments.

Topic	Generated Sentences
protesters suspect gunman officers occupy gun arrests suspects shooting officer	• police say a suspect in the shooting was shot in the chest and later shot and killed by a police officer . • a police officer shot her in the chest and the man was killed . • police have said four men have been killed in a shooting in suburban london .
film awards actress comedy music actor album show nominations movie	• it 's like it 's not fair to keep a star in a light , " he says . • but james , a four-time star , is just a ⟨unk⟩. • a ⟨unk⟩ adaptation of the movie " the dark knight rises " won best picture and he was nominated for best drama for best director of " ⟨unk⟩, " which will be presented sunday night .
storm snow weather inches flooding rain service winds tornado forecasters	• temperatures are forecast to remain above freezing enough to reach a tropical storm or heaviest temperatures . • snowfall totals were one of the busiest in the country . • forecasters say tornado irene 's strong winds could ease visibility and funnel clouds of snow from snow monday to the mountains .
virus nile flu vaccine disease outbreak infected symptoms cough tested	• he says the disease was transmitted by an infected person . • ⟨unk⟩ says the man 's symptoms are spread away from the heat . • meanwhile in the ⟨unk⟩, the virus has been common in the mojave desert .

Table 8: Generated sentences for APNEWS topics.

imented with a few values for the tag vector size (f) and find that a small value works well; in the following experiments we use $f = 5$. We evaluate the models based on language model perplexity and topic model coherence, and present the results in Table 7.[16]

In terms of language model perplexity, we see a consistent improvement over different topic settings, suggesting that the incorporation of tags improves modelling. In terms of topic coherence, there is a small but encouraging improvement (with one exception).

To investigate whether the vectors learnt for these tags are meaningful, we plot the top-14 most frequent tags in Figure 3.[17] The plot seems reasonable: there are a few related tags that are close to each other, e.g. "State government" and "Government and politics"; "Crime" and "Violent Crime"; and "Social issues" and "Social affairs".

7 Discussion

Topics generated by topic models are typically interpreted by way of their top-N highest probability words. In `tdlm`, we can additionally generate sentences related to the topic, providing another way to understand the topics. To do this, we can constrain the topic vector for the language model to be the topic output vector of a particular topic (Equation (3)).

We present 4 topics from a APNEWS model ($k = 100$; LSTM size = "large") and 3 randomly generated sentences conditioned on each topic in Table 8.[18] The generated sentences highlight the content of the topics, providing another interpretable aspect for the topics. These results also reinforce that the language model is driven by topics.

8 Conclusion

We propose `tdlm`, a topically driven neural language model. `tdlm` has two components: a language model and a topic model, which are jointly trained using a neural network. We demonstrate that `tdlm` outperforms a state-of-the-art language model that incorporates larger context, and that its topics are potentially more coherent than LDA topics. We additionally propose simple extensions of `tdlm` to incorporate information such as document labels and metadata, and achieved encouraging results.

Acknowledgments

We thank Shraey Bhatia for providing an open source implementation of `ntm`, and the anonymous reviewers for their insightful comments and valuable suggestions. This work was funded in part by the Australian Research Council.

References

Nikos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the Tenth International Workshop on Computational Semantics (IWCS-10)*. Potsdam, Germany, pages 13–22.

[16] As the vanilla `tdlm` is trained on the new APNEWS dataset, the numbers are slightly different to those in Tables 3 and 4.

[17] The 5-dimensional vectors are compressed using PCA.

[18] Words are sampled with temperature = 0.75. Generation is terminated when a special end symbol is generated or when sentence length is greater than 40 words.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

BNC Consortium. 2007. The British National Corpus, version 3 (BNC XML Edition). Distributed by Oxford University Computing Services on behalf of the BNC Consortium. http://www.natcorp.ox.ac.uk/.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Proceedings of the 29th Annual Conference on Artificial Intelligence (AAAI-15)*. Austin, Texas, pages 2210–2216.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems 21 (NIPS-09)*. Vancouver, Canada, pages 288–296.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar, pages 103–111.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning and Representation Learning Workshop*. Montreal, Canada, pages 103–111.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Felix A. Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'2000)*. Como, Italy, pages 198–194.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR* abs/1410.5401.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101:5228–5235.

Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2004. Integrating topics and syntax. In *Advances in Neural Information Processing Systems 17 (NIPS-05)*. Vancouver, Canada, pages 537–544.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*. Honolulu, USA, pages 363–371.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2009. Replicated softmax: an undirected topic model. In *Advances in Neural Information Processing Systems 21 (NIPS-09)*. Vancouver, Canada, pages 1607–1614.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9:1735–1780.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2016. Document context language models. In *Proceedings of ICLR-16 Workshop, 2016*. Toulon, France.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*. Sapporo, Japan, pages 423–430.

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems 25*. pages 2708–2716.

Jey Han Lau and Timothy Baldwin. 2016. The sensitivity of topic coherence evaluation to topic cardinality. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2016)*. San Diego, USA, pages 483–487.

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 EACL (EACL 2014)*. Gothenburg, Sweden, pages 530–539.

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 (ACL HLT 2011)*. Portland, Oregon, USA, pages 142–150.

Jon D. McAuliffe and David M. Blei. 2008. Supervised topic models. In *Advances in Neural Information Processing Systems 20 (NIPS-08)*. Vancouver, Canada, pages 121–128.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*. Makuhari, Japan, pages 1045–1048.

David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*. Edinburgh, UK, pages 262–272.

David Newman, Timothy Baldwin, Lawrence Cavedon, Sarvnaz Karimi, David Martinez, and Justin Zobel. 2010a. Visualizing document collections and search results using topic mapping. *Journal of Web Semantics* 8(2–3):169–175.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010b. Automatic evaluation of topic coherence. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010)*. Los Angeles, USA, pages 100–108.

Vu Pham, Christopher Kermorvant, and Jérôme Louradour. 2013. Dropout improves recurrent neural networks for handwriting recognition. *CoRR* abs/1312.4569.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28 (NIPS-15)*. Montreal, Canada, pages 2440–2448.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2016)*. San Diego, California, pages 321–331.

Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*. Montreal, Canada, pages 1105–1112.

Li Wan, Leo Zhu, and Rob Fergus. 2012. A hybrid neural network-latent topic model. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*. La Palma, Canary Islands, pages 1287–1294.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. Berlin, Germany, pages 1319–1329.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Philadelphia, USA, pages 424–433.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated LSTM. *CoRR* abs/1508.03790.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR* abs/1409.2329.