Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function

Devendra Singh Sachan
Petuum Inc
sachan.devendra@gmail.com

Manzil Zaheer
Carnegie Mellon University
manzilz@cs.cmu.edu

Ruslan Salakhutdinov
Carnegie Mellon University
rsalakhu@cs.cmu.edu

Abstract

In this paper, we study bidirectional LSTM network for the task of text classification using both supervised and semi-supervised approaches. Several prior works have suggested that either complex pretraining schemes using unsupervised methods such as language modeling (Dai and Le 2015; Miyato, Dai, and Goodfellow 2016) or complicated models (Johnson and Zhang 2017) are necessary to achieve a high classification accuracy. However, we develop a training strategy that allows even a simple BiLSTM model, when trained with cross-entropy loss, to achieve competitive results compared with more complex approaches. Furthermore, in addition to cross-entropy loss, by using a combination of entropy minimization, adversarial, and virtual adversarial losses for both labeled and unlabeled data, we report state-of-the-art results for text classification task on several benchmark datasets. In particular, on the ACL-IMDB sentiment analysis and AG-News topic classification datasets, our method outperforms current approaches by a substantial margin. We also show the generality of the mixed objective function by improving the performance on relation extraction task.[1]

1 Introduction

Text classification is an important problem in natural language processing (NLP). The task is to assign a document to one or more predefined categories. It has a wide range of applications such as sentiment analysis (Pang and Lee 2008), topic categorization (Lewis et al. 2004), and email filtering (Sahami et al. 1998). Early machine learning approaches for text classification were based on the extraction of bag-of-words features followed by a supervised classifier such as naïve Bayes (McCallum and Nigam 1998) or a linear SVM (Joachims 1998). Later, better word representations were introduced, such as latent semantic analysis (Deerwester et al. 1990), skipgram (Mikolov et al. 2013), and fastText (Joulin et al. 2017), which improved classification accuracy. Recently, recurrent and convolutional neural network (Kim 2014) models were introduced to utilize the word order and grammatical structure. Many complex variations of these models have been proposed to improve the text classification accuracy, e.g. training one-hot CNN (JZ15a; Johnson and Zhang 2015a) or one-hot bidirectional LSTM (BiLSTM) network with dynamic max-pooling (JZ16; Johnson and Zhang 2016).

Current state-of-the-art approaches for text classification involve using pretrained LSTMs (DL15; Dai and Le 2015) or complex computationally intensive models (JZ17; Johnson and Zhang 2017). DL15 argued that randomly initialized LSTMs are difficult to optimize and can lead to worse performance than linear models. Therefore, to improve the performance, they proposed *pretraining* the LSTM with either a language model or a sequence auto-encoder. However, pretraining or using complicated models can be very time consuming, which is a major disadvantage and may not be always feasible. In this paper, we consider a BiLSTM classifier model similar to the one proposed by DL15 for text classification. For this simple BiLSTM model with pretrained embeddings, we propose a training strategy that can achieve accuracy competitive with the previous purely supervised models, but without the extra pretraining step. We also perform ablation studies to understand aspects of the proposed training strategy that result in an improvement.

Pretraining approaches often use extra unlabeled data in addition to the labeled data. We explore the applicability of such semi-supervised learning (SSL) in our training framework, where there is no prior pretraining step. In this regard, we propose a *mixed objective function* for SSL that can utilize both labeled and unlabeled data to obtain further improvement in classification. To summarize, our contributions are as follows:

- We show that with proper model training, using a maximum likelihood objective with a simple one-layer BiLSTM model (§2) can produce competitive accuracies,
- We propose a mixed objective function that can be applied to text classification tasks (§2),
- On seven benchmark text classification tasks, we achieve new state-of-the-art results despite having a much simpler model, minimal model tuning, and fewer parameters (§4),
- We extend our proposed mixed objective function to relation extraction task, where we achieve better F1 score on SemEval-2010 and TACRED datasets, again with a simple model and minimal model tuning (§6).

[1] https://github.com/DevSinghSachan/ssl_text_classification

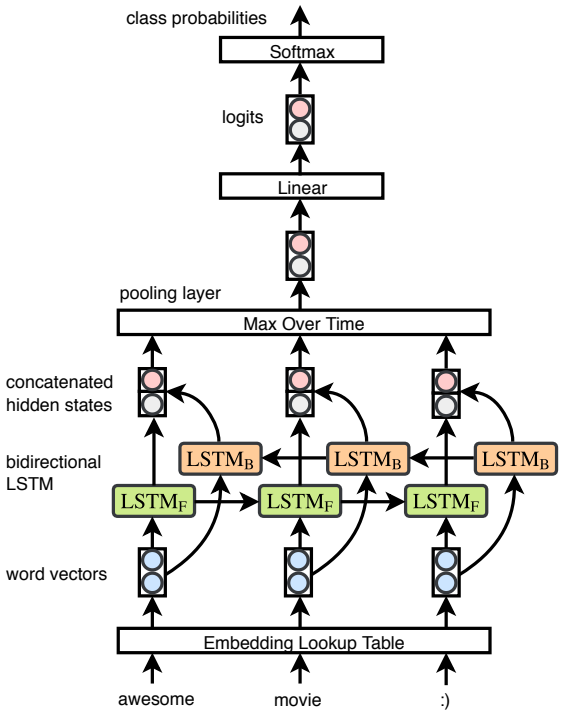

Figure 1: Text classification model architecture.

2 Methods

In this section, we will describe the model architecture, our training strategy, and our proposed mixed objective function. For mathematical notation, we will use bold lowercase to denote vectors, bold uppercase to denote matrices, and lowercase to denote scalars and individual words in a document.

Model Architecture

The classification model consists of an embedding layer, a bidirectional long short-term memory (BiLSTM) encoder (Hochreiter and Schmidhuber 1997; Schuster and Paliwal 1997), a max-pooling layer, a linear (fully-connected) layer, and finally a softmax layer (see Figure 1). First, the sequence of words $(w_1, \ldots, w_T)$ contained in a document are passed through an embedding layer that contains a lookup table which maps them to dense vectors $(\mathbf{v}_1, \ldots, \mathbf{v}_T,\ \mathbf{v}_t \in \mathbb{R}^d)$. Next, the forward and backward LSTM of the BiLSTM encoder processes these word vectors in the forward (left to right) and backward (right to left) directions respectively, updating corresponding hidden states at each time-step

$$\overrightarrow{\mathbf{h}_t} = \overrightarrow{\text{LSTM}_\text{F}}(\overrightarrow{\mathbf{h}_{t-1}}, \mathbf{v}_t),\ \overleftarrow{\mathbf{h}_t} = \overleftarrow{\text{LSTM}_\text{B}}(\overleftarrow{\mathbf{h}_{t-1}}, \mathbf{v}_t).$$

Next, these hidden state outputs from the forward LSTM ($\overrightarrow{\mathbf{h}_t}$) and backward LSTM ($\overleftarrow{\mathbf{h}_t}$) are concatenated at every time-step to enable encoding of information from past and future contexts respectively

$$\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t} \,||\, \overleftarrow{\mathbf{h}_t}],\ t \in [1, T],\ \mathbf{h}_t \in \mathbb{R}^n.$$

These concatenated hidden states are next fed to the pooling layer that computes the maximum value over time to obtain the feature representation of the input sequence ($\mathbf{h} \in \mathbb{R}^n$)

$$h^\ell = \max_t h_t^\ell,\ t \in [1, T],\ \forall \ell \in \{1, \ldots, n\}.$$

This max-pooling mechanism constrains the model to capture the most useful features produced by the BiLSTM encoder. Next, the linear layer applies an affine transformation to the feature vector to produce *logits* ($\mathbf{d}$)

$$\mathbf{d} = \mathbf{W}\mathbf{h} + b,\ \mathbf{d} \in \mathbb{R}^K,$$

where K is the number of classes, $\mathbf{W}$ is the weight matrix, and b is the bias. Next, these logits are normalized using the $\mathrm{softmax}$ function to give our estimated class probabilities as

$$p(y = k|\mathbf{x}; \boldsymbol{\theta}) = \frac{\exp(d_k)}{\sum_{j=1}^K \exp(d_j)},\ \forall k \in \{1, \ldots, K\},$$

where $(\mathbf{x}, y)$ is a training example and $\boldsymbol{\theta}$ denotes the model parameters. For model training, we use supervised and unsupervised loss functions, which are discussed next.

Supervised Training

Let there be m_ℓ labeled examples in the training set that are denoted as $\{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m_\ell)}, y^{(m_\ell)})\}$, where $\mathbf{x}^{(i)}$ represents a document's word sequence, $y^{(i)}$ represents class label such that $y^{(i)} \in \{1, 2, \ldots, K\}$. For supervised training of the classification model, we make use of two methodologies: *maximum likelihood* estimation and *adversarial training*, which are described next.

Maximum Likelihood (ML) This is the most widely used method to learn the parameters of neural network models from observed data for text classification task. Here, we minimize the average cross-entropy loss between the estimated class probability and the ground truth class label for all training examples

$$L_{ML}(\boldsymbol{\theta}) = \frac{-1}{m_\ell} \sum_{i=1}^{m_\ell} \sum_{k=1}^{K} \mathbb{1}(y^{(i)} = k) \log p(y^{(i)} = k|\mathbf{x}^{(i)}; \boldsymbol{\theta}),$$

where $\mathbb{1}(;)$ is an indicator function.

Adversarial Training (AT) Adversarial examples are created from inputs by small perturbations to mislead the machine learning algorithm. The objective of adversarial training is to construct and give as input adversarial examples during model training procedure to make the model more robust to adversarial noise and thereby improving its generalization ability (Goodfellow, Shlens, and Szegedy 2014).

In this work, we make adversarial perturbations to the input word embeddings ($\mathbf{v} = [\mathbf{v}_1, \ldots, \mathbf{v}_T]$) (MDG16, Miyato, Dai, and Goodfellow 2016). These perturbations ($\mathbf{r}_{at}$) are estimated by linearizing the supervised cross-entropy loss around the input word embeddings. Specifically, to get the adversarial embedding ($\mathbf{v}^*$) corresponding to $\mathbf{v}$, we use the L_2 norm of the training loss gradient ($\mathbf{g}$) that is computed by backpropagation using the current model parameters ($\hat{\boldsymbol{\theta}}$)

$$\mathbf{r}_{at} = \epsilon\mathbf{g}/\|\mathbf{g}\|_2,\ \text{where } \mathbf{g} = -\nabla_\mathbf{v} \log p(y = k|\mathbf{v}; \hat{\boldsymbol{\theta}})$$
$$\mathbf{v}^* = \mathbf{v} + \mathbf{r}_{adv}$$

where, k is the correct class label, ϵ is a hyperparameter that controls the magnitude of the perturbation. We apply adversarial loss to only the labeled data. It is defined as

$$L_{AT}(\boldsymbol{\theta}) = \frac{-1}{m_\ell} \sum_{i=1}^{m_\ell} \sum_{k=1}^{K} \mathbb{1}(y^{(i)} = k) \log p(y^{(i)} = k | \mathbf{v}^{*(i)}; \hat{\boldsymbol{\theta}}).$$

Unsupervised Training

In this paper, in addition to supervised training, we also experiment with two unsupervised methodologies: *entropy minimization* and *virtual adversarial training*. These loss functions when incorporated into the objective function act as effective regularizers during model training. To describe them, we assume that there exists an additional m_u unlabeled examples in the dataset $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m_u)}\}$.

Entropy Minimization (EM) In addition to supervised cross-entropy loss, we also minimize the conditional entropy of the estimated class probabilities (Grandvalet and Bengio 2004; Miyato et al. 2018). This can also be interpreted as a special case of the missing label problem where the probability $p(y^{(i)} = k | \mathbf{x}^{(i)}; \boldsymbol{\theta})$ signifies a soft assignment of the i^{th} example to label k (*i.e.* soft clustering). Entropy minimization loss is applied in an unsupervised manner to both the labeled and unlabeled data.

$$L_{EM}(\boldsymbol{\theta}) = \frac{-1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} p(y^{(i)} = k | \mathbf{x}^{(i)}) \log p(y^{(i)} = k | \mathbf{x}^{(i)}),$$

where $m = m_\ell + m_u$ and dependence on $\boldsymbol{\theta}$ is suppressed.

Virtual Adversarial Training (VAT) As opposed to minimizing the cross-entropy loss of the adversarial examples in AT, in VAT, we minimize the KL divergence between $p(\mathbf{v})$ and $p(\mathbf{v}^*)$, where $\mathbf{v}^* = \mathbf{v} + \mathbf{r}_{vat}$. The motivation of using KL divergence as an additional loss term in the objective function is that it tends to make the loss surface smooth at the current example (Miyato et al. 2018). Also, computing the VAT loss doesn't require class labels, so it can be applied to unlabeled data as well. In this work, we follow the approach proposed by MDG16 that makes use of the second-order Taylor expansion of distance followed by power iteration method to approximate the virtual adversarial perturbation. First, an i.i.d. random unit vector is sampled for every example from the Normal distribution ($\mathbf{d}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \in \mathbb{R}^d$) and then adversarial perturbation computed as $\xi \mathbf{d}^{(i)}$ is added to the word embeddings, where ξ is a hyperparameter

$$\mathbf{v}'^{(i)} = \mathbf{v}^{(i)} + \xi \mathbf{d}^{(i)}.$$

Dataset	Train	Test	K	ℓ
ACL-IMDB	25,000	25,000	2	268
Elec	25,000	25,000	2	125
AG-News	120,000	7,600	4	46
DBpedia	560,000	70,000	14	56
RCV1	15,564	49,821	51	120
IMDB	1,490,000	1,070,000	5	280
Arxiv	664,000	443,000	127	153

Table 1: Summary statistics for text classification datasets; K = number of classes; ℓ = average length of a document.

Next, the gradient is estimated from the KL divergence as:

$$\mathbf{g} = \nabla_{\mathbf{v}'} D_{KL}(p(. \mid \mathbf{v}^{(i)}; \hat{\boldsymbol{\theta}}) \parallel p(. \mid \mathbf{v}'^{(i)}; \hat{\boldsymbol{\theta}})).$$

Virtual adversarial perturbation ($\mathbf{r}_{vadv}$) is generated using the L_2 norm of the gradient and added to the word embeddings

$$\mathbf{r}_{vat}^{(i)} = \epsilon \mathbf{g} / \|\mathbf{g}\|_2$$
$$\mathbf{v}^{*(i)} = \mathbf{v}^{(i)} + \mathbf{r}_{vat}^{(i)} .$$

Lastly, virtual adversarial loss can be computed from both the labeled and unlabeled data as:

$$L_{VAT}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} D_{KL}(p(. \mid \mathbf{v}^{(i)}; \boldsymbol{\theta}) \parallel p(. \mid \mathbf{v}^{*(i)}; \boldsymbol{\theta})),$$

where $m = m_\ell + m_u$.

Mixed Objective Function

Our proposed mixed objective function combines the above described supervised and unsupervised loss functions using λ_{ML}, λ_{AT}, λ_{EM}, and λ_{VAT} as hyperparameters

$$L_{MIXED} = \lambda_{ML} L_{ML} + \lambda_{AT} L_{AT} + \lambda_{EM} L_{EM} + \lambda_{VAT} L_{VAT}.$$

Training Strategy

We note that there are three main considerations to be taken into account when training the text classifier. First, the knowledge of the entire sequence is essential for good classification performance. Thus, commonly used practice of truncated backpropagation through time (Werbos 1988) is a key limiting factor. One should perform gradient update for the entire text sequence. To prevent out of memory issues that can result from longer sequences, we propose to use *dynamic batch size* that consists of fixed number of total words per mini-batch. Second, not only we need to use pretrained word embeddings, but we need to finetune them for the specific task. Lastly, we should use a larger vocabulary size and not limit to only high-frequency words. This is because rare or tail words are often strong indicators of the class.

3 Experiments

Dataset Description

In this work, we experiment with seven datasets that are summarized in Table 1. *ACL-IMDB* (Maas et al. 2011) and

Dataset	BSize	Vocab	ϵ
ACL-IMDB	3,000	80,000	5
Elec	2,000	40,000	2
AG-News	2,000	75,000	1
DBpedia	7,500	50,000	1
RCV1	2,000	100,000	2
IMDB	15,000	150,000	5
Arxiv	8,000	100,000	1

Table 2: Dataset-specific hyperparameters; BSize = number of tokens in a minibatch; Vocab = number of words present in vocabulary; ϵ = adversarial perturbation.

Model	ACL-IMDB	Elec	AG-News	DBpedia	RCV1	IMDB	Arxiv
Linear Model (TFIDF + SVM)	9.51	9.16	7.64	1.31	10.68	40.00	34.81
Vanilla LSTM [Dai and Le (2015)]	10.00	–	–	13.64	–	–	–
DocVec [Le and Mikolov (2014)]	11.40	8.95	8.00	2.34	–	44.10	37.40
FastText [Joulin et al. (2017)]	11.34	–	7.50	1.40	–	42.13	33.23
CNN [Kim (2014)]	9.17	8.03	5.92	0.98	10.44	49.53	34.21
oh-CNN [Johnson and Zhang (2015b; 2017)]	7.67	**7.14**	6.88	**0.88**	9.17	38.15	35.89
char-CNN [Zhang, Zhao, and LeCun (2015)]	–	–	9.51	1.55	–	–	–
L_{ML} [Our Method]	**6.43**	7.40	**5.62**	0.91	**7.78**	**35.64**	**31.76**

Table 3: Error rates (%) when the model is trained using L_{ML} and comparison with previous best supervised methods.

Model	ACL-IMDB	Elec	AG-News	DBpedia	RCV1	IMDB	Arxiv
SA-LSTM [Dai and Le (2015)]	7.24	–	–	1.19	7.40	–	–
LSTM [Miyato, Dai, and Goodfellow (2016)]	5.91	5.40	6.78	0.76	6.68	35.85	30.97
oh-LSTM [Johnson and Zhang (2016; 2017)]	5.94	5.55	6.57	0.84	7.15	37.56	31.17
ULMFit [Howard and Ruder (2018)]	4.60	–	5.01	0.80	–	–	–
L_{MIXED} [Our Method]	**4.32**	**5.24**	**4.95**	**0.70**	**6.23**	**34.04**	**29.95**

Table 4: Error rates (%) when the model is trained using L_{MIXED} and comparison with previous best semi-supervised methods.

Elec (JZ15a) datasets are widely used for binary sentiment classification of movie reviews and Amazon product reviews respectively while *AG-News*, *DBpedia* (Zhang, Zhao, and LeCun 2015), and *RCV1* (Lewis et al. 2004) are for topic classification of news articles, Wikipedia, and Reuters corpus respectively. For the RCV1 dataset, we perform multiclass topic classification based on the second-level topics and construct its training, dev, and test splits in accordance with JZ15a. To show that our proposed method also scales to larger datasets and categories, we also experiment with large-scale datasets of *IMDB* reviews and *Arxiv* abstracts that are used for fine-grained sentiment- and topic classification respectively (Sachan, Zaheer, and Salakhutdinov 2018). We preprocess all the datasets by converting the text to lowercase and treat all punctuations as separate tokens.

Implementation Details

All of our models were implemented in PyTorch framework (Paszke et al. 2017) and were trained on a single GPU. Our experimental setup is common for all the datasets unless specified otherwise. We use 300D pretrained vectors to initialize the embedding layer. We learn embeddings for the ACL-IMDB, RCV1, IMDB, and Arxiv datasets using `word2vec` (Mikolov et al. 2013) and use `fastText` pretrained embeddings[2] (Mikolov et al. 2018) for all the other datasets. We use one-layer BiLSTM of size 512D. For regularization, we apply dropout ($p_{drop} = 0.5$) to word embeddings and to LSTM's hidden states. We also use the word dropout strategy in which we randomly set a word to be "UNK" with a probability $p_w = 0.1$. For training, we employ SGD using *Adam* optimizer (Kingma and Ba 2014) (*learning rate* $= 10^{-3}$, $\beta_1 = 0$, $\beta_2 = 0.98$, $\epsilon_{adam} = 10^{-8}$) with an exponential learning rate decay scheme. We perform gradient clipping by having a maximum L_2 norm of 1. For training, we backpropagate through time over entire sequence, i.e. we did not truncate sequence. This differs from DL15 where they perform truncated backpropagation through time for 400 time-steps from the end of a sequence.

For semi-supervised training, we experiment with all the objective functions described in §2. For L_{MIXED}, we include all the constituent terms with λ_{ML}, λ_{AT}, λ_{EM}, λ_{VAT} set to 1 and $\xi = 0.1$. We want to emphasize that in contrast with MDG16, we do not perform *embedding layer normalization* during AT or VAT objectives, as by including it, we noticed a drop in accuracy during our initial experiments. We select the hyperparameters such as dynamic batch size, vocabulary size, and adversarial perturbation (ϵ) by cross-validation on the development set. We mention these dataset-specific hyperparameters in Table 2. For supervised experiments (L_{ML}), we perform training till 20 epochs and for semi-supervised experiments (L_{MIXED}), training is done till 50 epochs. For ACL-IMDB, Elec, RCV1, IMDB, and Arxiv datasets, we use training and test set as unlabeled data, while for AG-News and DBpedia datasets as their test sets are small, we use only the training set as unlabeled data. During training, we keep the batch size of the unlabeled data the same as that of the labeled data.

4 Results

In this section, we report the classification accuracy on the test set and perform ablation studies for both supervised and semi-supervised training.

Maximum Likelihood Training

In Table 3, we present the error rates of our method and the previous best-published models when training is done using only the maximum likelihood objective (L_{ML}). We observe that our model that consists of one-layer BiLSTM and pre-

[2]These embeddings were trained on data containing 600B tokens from Common Crawl (*crawl-300d-2M.vec*).

N	*Embedding*	p_w	*BSize*	H	*Vocab*	ACL-IMDB
1	*Finetune*	0.1	3,000	512	80,000	6.43
2						6.47
	Random					7.64
	Static					8.17
		0.0				6.57
			1,000			6.98
				256		6.67
				1,024		7.05
					30,000	7.78
No text preprocessing						8.80

Table 5: Error rates (%) of variations on the BiLSTM model trained using L_{ML} on the ACL-IMDB dataset. Unlisted values are identical to those of the first row; N = number of BiLSTM layers; H = LSTM hidden size.

L	U	λ_{ML}	λ_{AT}	λ_{EM}	λ_{VAT}	ACL-IMDB
1	0	1	0	0	0	6.43
1	0	1	1	0	0	5.96
1	0	1	0	1	0	6.46
1	0	1	0	0	1	5.98
1	0	1	1	1	1	5.68
1	1	1	0	1	0	5.78
1	1	1	0	0	1	5.52
1	1	1	0	1	1	4.47
1	1	1	1	1	1	4.32

Table 6: Error rates (%) for ablation study on the importance of hyperparameters when the BiLSTM model is trained using L_{MIXED} objective; L = was labeled data used? U = was unlabeled data used?

trained embedding weights achieves a very competitive performance on all the datasets compared with the more complex approaches such as one-hot LSTM (JZ16) or pyramidal CNN (JZ17). Specifically, for ACL-IMDB, AG-News, IMDB, and Arxiv datasets, we report much better results than earlier methods. Thus, our proposed model and training strategy enjoy the following advantages: (a) it is very easy to implement using current deep learning frameworks; (b) it requires much less training time and GPU memory compared with other complicated models; (c) it entirely avoids complex initialization strategies such as pretraining the LSTM weights using a language model; (d) Our results can serve as strong baselines when developing more advanced task-specific models.

To know the importance of various components in the model and training regimen, we perform ablation studies using the ACL-IMDB dataset (see Table 5). We verify that good performance of our model mostly results from finetuning the pretrained embeddings, using a larger vocabulary size, and using a carefully preprocessed dataset. We also see that excluding word dropout, smaller-sized LSTM, and lowering the batch size causes a slight drop in performance while using static pretrained or randomly initialized embeddings or smaller vocabulary size can cause a large drop.

Semi-Supervised Training

For our next set of experiments, we perform training using L_{MIXED} objective whose results are shown in Table 4. We observe that the mixed objective improves over maximum likelihood objective and achieves state-of-the-art results on all the seven datasets. Specifically, on the widely used ACL-IMDB dataset, there is a substantial reduction of 26.9% in relative error compared with the previous best-published model of JZ16, which was substantially more complex as they use one-hot encodings of words along with a lot of additional features such as multi-view region embeddings from CNNs and LSTMs. We also want to highlight that, although the model of MDG16 also experiments with adversarial and virtual adversarial training, our approach performs much better compared with them due to our improved training strategy and the use of L_{EM} objective. Similarly, for the benchmark AG-News dataset, we observe relative error reduction of 26.6% compared with previous state-of-the-art model of JZ17 who use a very deep pyramidal-CNN along with region embeddings. Even on the Elec, DBPedia, and RCV1 datasets, our results present significant improvements over the previous best semi-supervised results. L_{MIXED} objective also scales well to the dataset sizes, as on the large datasets of IMDB and Arxiv, it outperforms the above mentioned previous approaches by a substantial margin. We note here that the approach of Howard and Ruder (2018) is not directly comparable with our results as they use a three-layer LSTM model. We discuss the effect of model size in §5.

Next, we perform ablation studies when the model is trained using L_{MIXED} on the ACL-IMDB dataset and analyze the contributions of the different component terms present in the objective (see Table 6). First, we observe that when the model is trained using L_{MIXED} objective on both the labeled and unlabeled data, the accuracy on ACL-IMDB drastically improves by 33% compared with using only the L_{ML} objective. Second, we also observe that when trained only on labeled data the inclusion of L_{AT} and L_{VAT} can also significantly improve the performance. However, L_{EM} alone doesn't lead to any significant gains. Furthermore, when L_{MIXED} is trained with only labeled data, we see 12% relative increase in accuracy. Finally, when we add unlabeled data to both L_{VAT} and L_{EM}, we see consistent improvements, thus suggesting that these objective functions complement each other and together improve the overall performance.

5 Analysis

Effect on Word Embeddings

To understand the effect on word embeddings due to training using L_{ML} and L_{MIXED} objectives, we show the top-10 closest words for the query word "*good*" based on their cosine similarity in Table 7, where the word embeddings were extracted from the models trained on ACL-IMDB dataset. We see that for static embeddings, the closest words have a mix of both positive ('*great*', '*decent*', '*nice*') and negative sentiments ('*bad*', '*but*'). This can be understood as they are syntactically similar adjectives. When these embeddings are

query word: good		
word2vec	**ML Objective**	**Mixed Objective**
great (0.64)	great (0.66)	funny (0.73)
really (0.61)	nice (0.62)	well-acted (0.72)
decent (0.60)	decent (0.58)	interesting (0.66)
nice (0.59)	entertaining (0.56)	fine (0.65)
ok (0.57)	overall (0.55)	nice (0.65)
but (0.56)	really (0.55)	thought-provoking (0.63)
pretty (0.56)	liked (0.54)	decent (0.62)
overall (0.55)	lot (0.52)	worth (0.60)
bad (0.54)	enjoyable (0.52)	recommend (0.60)
movie (0.53)	fun (0.51)	recommendable (0.60)

Table 7: Top-10 nearest neighbors according to cosine similarity (shown in parentheses) for the word "*good*" computed in the embedding space. "*word2vec*" refers to the static embeddings i.e. not finetuned during training.

finetuned using the L_{ML} objective, the network learns more meaningful representations and accommodates more positive sentiment words close to the query word '*good*'. Moreover, when trained using the L_{MIXED} objective, we see that those words that have a very high correlation with the class label (*positive sentiment* class in this case) are clustered together in the embedding space. Our hypothesis is that this factor also contributes to an increase in the overall classification accuracy.

Model Regularization Effect

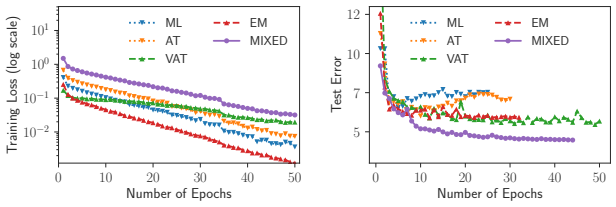

Figure 2: (a) Training loss vs. epochs on ACL-IMDB; (b) Test error vs. epochs on ACL-IMDB.

Figure 2a and Figure 2b show the moving average training loss and test error respectively versus the number of epochs on the ACL-IMDB dataset with the L_{ML}, L_{AT}, L_{VAT}, L_{EM}, and L_{MIXED} objectives. We can see that L_{ML} begins to overfit after 5 epochs, L_{AT} overfits after 10 epochs while L_{MIXED}, L_{EM}, and L_{VAT} don't overfit much and thus achieve better generalization than the L_{ML} and L_{AT} objectives (see in Figure 2b). Moreover, as L_{MIXED} and L_{VAT} objectives can use unlabeled data, their training loss decays gradually. Thus, L_{MIXED} objective while being very effective in performance is also a very robust model regularizer. On the other hand, from Figure 2b, we can see that L_{VAT}, L_{EM}, and L_{MIXED} take a long time to converge compared with L_{ML} and L_{AT} and are thus quite slow to train. In our experiments, one epoch of L_{MIXED} takes around 20m on GeForce GTX 1080 GPU and it requires roughly 45 epochs to converge. This is considerably slower than L_{ML} objective where each epoch takes around 3m and the overall convergence time is thus 15m for 5 epochs.

Varying Data Size

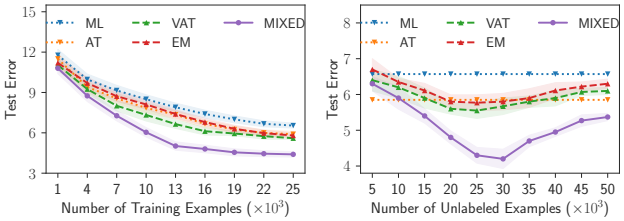

Figure 3: Test Error on ACL-IMDB vs. (a) number of training examples (b) increasing number of unlabeled examples.

In this setup, we first analyze the test error on ACL-IMDB dataset by feeding the model trained with different objective functions (§2) with an increasing number of training examples (learning curve; see Figure 3a). We observe that all the objective functions converge to lower error rates when training data is increased. We also see that mixed objective model is always optimal (achieves lower test error rate) for any setting of the number of training examples.

Next, we analyze the test error on ACL-IMDB dataset by varying the amount of unlabeled data. For this experiment, we use additional 50,000 reviews provided with the ACL-IMDB dataset and its 25,000 reviews from test set as unlabeled data. We evaluate the performance of each objective function by linearly increasing the amounts of unlabeled data (see Figure 3b). Initially, increasing the amount of unlabeled data tends to improve the performance of L_{VAT}, L_{EM}, and L_{MIXED}. However, we observe that their performance saturates once 25,000 unlabeled examples are available. Furthermore, as the amount of unlabeled data increases, the performance tends to degrade sharply. As ACL-IMDB training set also consists of 25,000 examples, from this observation, it can be assumed that to obtain the best performance using L_{MIXED}, the size of unlabeled and labeled dataset should be roughly the same. We also note that as L_{ML} and L_{AT} are supervised approaches, their performance remains unaffected.

Varying Model Size

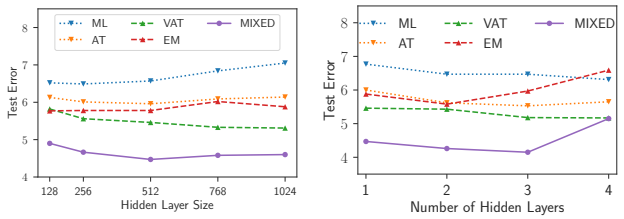

Figure 4: Test error on ACL-IMDB vs. (a) hidden layer size of LSTMs in the BiLSTM with one-hidden layer; (b) number of BiLSTM layers where each LSTM has size of 512.

We note that prior work in the supervised text classification task has used smaller-sized LSTMs *i.e.* models with

Sentiment	Text
	Mixed Objective Training (L_{MIXED})
Negative	This movie is so over-the-top as to be a borderline comedy. Laws of physics are broken. Things explode for no good reason. Great movie to sit down with a six-pack and enjoy. Do not - I repeat DO NOT see this movie sober. You will die horrible death!
	Entropy Minimization Training (L_{EM})
Positive	Coming from Oz I probably shouldn't say it but I find a lot of the local movies lacking that cohesive flow with a weak storyline. This comedy lacks in nothing. Great story, no overacting, no melodrama, just brilliant comedy as we know Oz can do it. Do yourself a favour and laugh till you drop.
	Virtual Adversarial Training (L_{VAT})
Negative	The plot seemed to be interesting, but this film is a great dissapointment. Bad actors, a camera moving like in the hands of an amateur. If there was C-movies, this would be a perfect example. A plus for a nice DVD cover though and a great looking female actor.

Table 8: Examples from ACL-IMDB dataset for sentiment classification task that are correctly classified by the method indicated directly above it and incorrectly classified by all the other methods.

hidden state sizes at most 512 units because larger models didn't give accuracy gains (DL15, JZ16). This is also consistent with our observation, as we find that that supervised approaches (L_{ML}) do not benefit much from increasing the model size. However, when using additional loss functions such as the mixed objective, accuracy scales much better with model size (see Figure 4a). Further, we also observe accuracy gains for all methods upon increasing the number of layers in the model (see Figure 4b). Specifically, the error rate of L_{MIXED} objective improves to 4.15% when using a three-layer deep model. This suggests that larger-sized semi-supervised methods can lead to the development of more accurate models for text classification task. However, a four-layer model hurts the L_{MIXED} objective's performance due to the training instability of L_{EM} method.

Effect on Prediction Probabilities

To study the behavior of different methods, we plot a histogram of the prediction probabilities for both the correct (Figure 5a) and incorrect (Figure 5b) predictions. We observe that for correct predictions all the methods especially L_{EM} and L_{MIXED} have very sharp and confident distribution of class probabilities. However, for incorrect predictions only L_{EM} has sharp peaks while L_{VAT}, L_{AT}, and L_{ML} encourage the model to learn a smoother distribution.

Ensemble Approach vs Mixed Objective

To understand if the above objectives have complementary strengths, we combine their predicted probabilities with a linear interpolation strategy. Given the output probability for a class k as $p(y = k|\mathbf{x})$, the interpolated probability $p_I(y = k|\mathbf{x})$ is calculated as:

$$p_I = \alpha_{ML} p_{ML} + \alpha_{AT} p_{AT} + \alpha_{VAT} p_{VAT} + \alpha_{EM} p_{EM} \mid \sum \alpha_i = 1,$$

where $\alpha_i \in [0, 1]$ and is chosen based on grid search. This simple interpolation technique results in an improved error rate of 5.2%. However, the error rate of our proposed mixed objective function is substantially lower (4.3%) thus highlighting the importance of performing joint training of the model based on different objective functions.

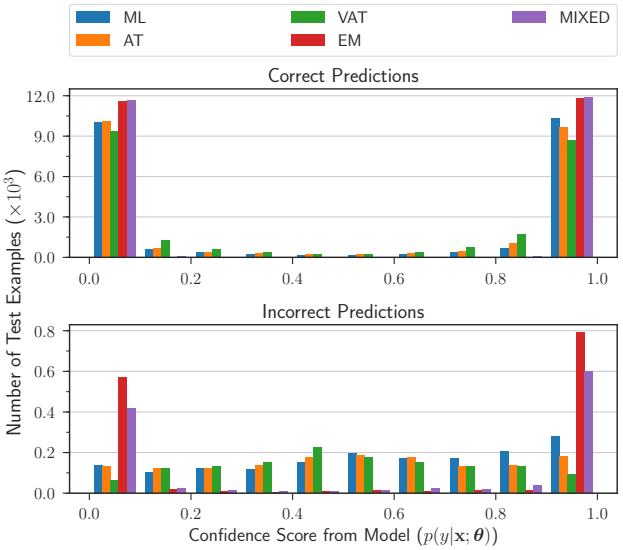

Figure 5: Histogram of prediction probabilities on ACL-IMDB test set for the case of (a) correct predictions; (b) incorrect predictions.

Example Predictions

In Table 8, we show some example movie reviews from the test set of ACL-IMDB dataset that are correctly classified by the highlighted method and incorrectly by all the remaining methods. We observe that methods such as L_{MIXED}, L_{EM}, and L_{VAT} that are based on unsupervised training are able to correctly classify difficult instances in which the overall sentiment is determined by the entire sentence structure. This illustrates the ability of these methods to learn complex long-range dependencies.

6 Relation Extraction

To evaluate if the mixed objective function can generalize to other tasks we also perform experiments on relation extraction (RE) task. In this, the objective is to identify if a

predefined semantic "*relation category*" exists (or doesn't exist) between a pair of *subject* and *object* entities present in text. This task also presents specific challenges: the linguistics *coverage problem* due to the lack of all possible training examples of a relation class and longer text span between the entities in a sentence.

For the RE task, we use a position-aware attention model (Zhang et al. 2017) that consists of a word embedding, position embedding, LSTM, attention layer, linear layer, and softmax layer. We augment the word embeddings by concatenating them with POS tag and NER category embeddings which is then fed to the LSTM to get hidden states for each word. The position embeddings for a word is derived based on the relative distance of the current word from the subject and object entities. Next, the attention layer computes the final sentence representation by focusing on both the hidden states and position embeddings. Finally, sentence representation is fed to the linear layer followed by a softmax layer for relation classification.

We perform experiments using our proposed mixed objective function on two RE datasets: *TACRED* and *SemEval-2010 Task 8* whose statistics are shown in Table 9. The POS and NER tags are computed using the Stanford CoreNLP toolkit.[3] Following standard convention, we report the micro-averaged F_1 score on TACRED and official macro-averaged F_1 score on SemEval datasets. We performed only a small number of experiments to search for the hyper-parameter values of dropout, embedding size, hidden layer size, and learning rate on the development set, all other parameters remained the same as the positional attention model of Zhang et al. (2017).[4] Our results in Table 10 show that when trained with mixed objective function, our model performs quite well, producing better results than all previously reported models despite the lack of complex task-specific hyper-parameter tuning.

Dataset	Train	Dev	Test	% Neg	K	ℓ
TACRED	68,124	22,631	15,509	79.5%	42	36
SemEval	8,000	–	2,717	17.4%	19	19

Table 9: Summary statistics for RE datasets; % Neg = percentage of examples with class label of "*no relation*" between entities; K = number of classes including the *no relation* class; ℓ = average length of a sentence in the dataset.

	TACRED			SemEval-2010		
Model	**P**	**R**	**F_1**	**P**	**R**	**F_1**
CNN-PE[a]	**70.3**	54.2	61.2	82.1	83.1	82.5
SDP-LSTM[b]	66.3	52.7	58.7	–	–	83.7
PA-LSTM[c]	65.7	64.5	65.1	–	–	82.7
Our Method	66.4	**67.3**	**66.8**	**83.5**	**84.8**	**84.1**

Table 10: Model performance on TACRED and SemEval datasets; P = Precision; R = Recall; [a]Santos, Xiang, and Zhou (2015); [b]Xu et al. (2015); [c]Zhang et al. (2017). For fair comparison, we only report results from models that don't use ensembling or ranking-based approaches.

7 Related Work

Neural Network Methods. Neural network models for NLP have yielded impressive results on several benchmark tasks (Collobert et al. 2011; Peters et al. 2018). To learn document features for text classification task, several methods have been proposed— Kim (2014) uses 1D CNNs, Lai et al. (2015) uses a simple bidirectional recurrent CNN with max-pooling, Zhou et al. (2016) applies 2D max-pooling on top of BiLSTMs, Zhou et al. (2015) investigates a joint CNN-LSTM model, and JZ15a, JZ16, JZ17 apply CNNs, LSTMs, and pyramidal CNNs respectively to one-hot encoding of word sequences. An alternative approach is to first learn sentence representations followed by combining them to learn document features. To do this, Tang, Qin, and Liu (2015) first apply a CNN or LSTM followed by a gated RNN while Yang et al. (2016) learn the sentence and document features in a hierarchical manner using a self-attention mechanism.

Semi-Supervised Learning. SSL approaches can be broadly categorized into three types: multi-view, data augmentation, and transfer learning. First, under multi-view learning, the objective is to use multiple views of both the labeled and unlabeled data to train the model. These multiple views can be obtained either from raw text (Blum and Mitchell 1998) or from the features (JZ15b). Second, under data augmentation, as the name implies, involves pseudo-augmenting either the features or the labels. For text classification, Nigam et al. (2000) performed semi-supervised training using naïve Bayes and expectation-maximization algorithms and demonstrated substantial improvements in performance. MDG16 compute embedding perturbations using adversarial and virtual adversarial approaches to improve model training. Third, under transfer learning, the approach of initializing the task-specific model weights by pretrained weights from an auxiliary task is a widely used strategy that has shown to improve the performance in tasks such as text classification (Dai and Le 2015; Howard and Ruder 2018), question-answering (Devlin et al. 2018), and machine translation (Ramachandran, Liu, and Le 2017; Qi et al. 2018).

8 Conclusion

We show that a simple BiLSTM model using maximum likelihood training can result in a competitive performance on text classification tasks without the need for an additional pretraining step. Also, in addition to maximum likelihood, using a combination of entropy minimization, adversarial, and virtual adversarial training, we report state-of-the-art results on several text classification datasets. This mixed objective function also generalizes well to other tasks such as relation extraction where it outperforms current best models.

[3]https://stanfordnlp.github.io/CoreNLP/

[4]https://github.com/yuhaozhang/tacred-relation

References

Blum, A., and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *JMLR*.

Dai, A. M., and Le, Q. V. 2015. Semi-supervised sequence learning. In *NIPS*.

Deerwester, S.; Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; and Harshman, R. 1990. Indexing by latent semantic analysis. *JAIST*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *Computing Research Repository* arXiv:1810.04805.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *Computing Research Repository* arXiv:1412.6572.

Grandvalet, Y., and Bengio, Y. 2004. Semi-supervised learning by entropy minimization. In *NIPS*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation*.

Howard, J., and Ruder, S. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *ECML*.

Johnson, R., and Zhang, T. 2015a. Effective use of word order for text categorization with convolutional neural networks. In *NAACL*.

Johnson, R., and Zhang, T. 2015b. Semi-supervised convolutional neural networks for text categorization via region embedding. In *NIPS*.

Johnson, R., and Zhang, T. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *ICML*.

Johnson, R., and Zhang, T. 2017. Deep pyramid convolutional neural networks for text categorization. In *ACL*.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of tricks for efficient text classification. In *EACL*.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *Computing Research Repository* arXiv:1412.6980.

Lai, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*.

Lewis, D. D.; Yang, Y.; Rose, T. G.; and Li, F. 2004. Rcv1: A new benchmark collection for text categorization research. *JMLR*.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *ACL*.

McCallum, A., and Nigam, K. 1998. A comparison of event models for naive bayes text classification. In *Workshop on Learning for Text Categorization, AAAI*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Mikolov, T.; Grave, E.; Bojanowski, P.; Puhrsch, C.; and Joulin, A. 2018. Advances in pre-training distributed word representations. In *LREC*.

Miyato, T.; Maeda, S.-i.; Koyama, M.; and Ishii, S. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE TPAMI*.

Miyato, T.; Dai, A. M.; and Goodfellow, I. 2016. Adversarial training methods for semi-supervised text classification. *Computing Research Repository* arXiv:1605.07725.

Nigam, K.; McCallum, A. K.; Thrun, S.; and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*.

Pang, B., and Lee, L. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *NAACL*.

Qi, Y.; Sachan, D.; Felix, M.; Padmanabhan, S.; and Neubig, G. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *NAACL*.

Ramachandran, P.; Liu, P.; and Le, Q. 2017. Unsupervised pretraining for sequence to sequence learning. In *EMNLP*.

Sachan, D. S.; Zaheer, M.; and Salakhutdinov, R. 2018. Investigating the working of text classifiers. In *COLING*.

Sahami, M.; Dumais, S.; Heckerman, D.; and Horvitz, E. 1998. A bayesian approach to filtering junk E-mail. In *Workshop on Learning for Text Categorization, AAAI*.

Santos, C. d.; Xiang, B.; and Zhou, B. 2015. Classifying relations by ranking with convolutional neural networks. In *ACL*.

Schuster, M., and Paliwal, K. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*

Tang, D.; Qin, B.; and Liu, T. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*.

Werbos, P. J. 1988. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*.

Xu, Y.; Mou, L.; Li, G.; Chen, Y.; Peng, H.; and Jin, Z. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *NAACL*.

Zhang, Y.; Zhong, V.; Chen, D.; Angeli, G.; and Manning, C. D. 2017. Position-aware attention and supervised data improve slot filling. In *EMNLP*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Zhou, C.; Sun, C.; Liu, Z.; and Lau, F. C. M. 2015. A C-LSTM neural network for text classification. *Computing Research Repository* arXiv:1511.08630.

Zhou, P.; Qi, Z.; Zheng, S.; Xu, J.; Bao, H.; and Xu, B. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *COLING*.