Model Blending for Text Classification

Ramit Pahwa[1] **Surya Dwivedi**[2] **Jayanta Mukhopadhyay**[3] **Sunav Choudhary**[4] **Vishwa Vinay**[4*]
[1]Adobe
[2]Oracle
[3]Indian Institute of Technology, Kharagpur
[4]Adobe Research
{ramitpahwa123,surya2191997}@gmail.com, jay@cse.iitkgp.ac.in, {schoudha,vinay}@adobe.com

Abstract

State-of-the-art for many NLP tasks are dominated by deep learning models. Effectively utilizing the signal in the sequence of words via Recurrent Neural Networks (RNNs) provides impressive accuracies, especially when using LSTM-based models. These models however have the disadvantage of longer scoring times. Convolutional Neural Networks (CNNs) on the other hand are significantly faster during inference due to their parallelism. Moreover, some recent work suggests that RNN and CNN provide complementary information when trained on the same data for various NLP tasks. In this paper, we propose a method that allows the training of very competitive CNNs by *blending* in the complementary knowledge from an LSTM Teacher model. We empirically validate the method on multiple standard text classification datasets to demonstrate that the Student CNN model can be up to 15x faster than the Teacher LSTM, while being more accurate than training the same CNN model directly from the data.

1 Introduction

Text classification is fundamental problem in Natural Language Processing (NLP) and is an essential component in applications such as sentiment analysis (Maas et al., 2011), question classification (Voorhees and Tice, 1999) and topic classification (Zhang et al., 2015). In recent year there has been a shift from traditional bag of words (BoW) model, which treats textual data as an un-ordered set of words (Wang and Manning, 2012) to more recent order sensitive models which have achieved exceptional performance across numerous NLP tasks. The challenge for textual modelling is to tackle how to capture features from various textual units such as phrases, sentences and documents. Benefiting from the sequential nature of the textual data, the recurrent neural network (RNN) in particular Long short term memory (LSTM) (Zhou et al., 2016) units have been successful in modelling the sequence in numerous NLP tasks. But it comes with its limitations as it processes the data in a sequential manner, thereby increasing the execution latency and hindering deployment on devices with strict latency requirements. The alternative deep learning architecture which in the recent years has gained popularity over LSTM is the convolution neural network (CNN) (Kim, 2014) primarily due to its parallelism, flexible receptive filter window (Wang et al., 2018) and low memory requirement while training in comparison to LSTM network. The inherent difference in the architectures of CNN and LSTM pose an interesting question of how similar the functions they represent are . A recent work by (Yin et al., 2017) shows that the knowledge gained by LSTM and CNN architectures are *complementary* in nature. Thus a natural question is whether there exists a method to incorporate this complementary knowledge among the different model families. In this paper we answer this question in affirmative and propose *model blending*, a method to incorporate the knowledge of LSTMs into CNNs during their training. The choice to blend the knowledge from LSTM into CNN and not vice-versa is motivated by the fact that CNNs are faster at test time due to their inherent parallelism. Thus, we blend the complementary inductive biases of two deep learning models, namely LSTM and CNN, resulting in a CNN model which is much more faster at test time as compared to the teacher LSTM. In summary, the contributions of this paper include:

- We propose blending into CNN using LSTM priors resulting in accurate CNN classifiers which can be upto **15x** more computationally efficient at test time and take a fraction of

the training time as compared to the teacher LSTM.

- We empirically find that using a more accurate Teacher LSTM we can improve on the performance of our Student model.

- We extensively evaluate the blending process across text classification tasks such sentiment analysis (Maas et al., 2011), question classification (Li and Roth, 2002; Voorhees and Tice, 1999) and topic classification (Zhang et al., 2015; Lehmann et al., 2015)and obtain competitive performance when compared with state of the art models.

2 Related Work

There are two streams of literature relevant to the current work. On the one hand, there are standardized text classification tasks with published works aimed at increasing the effectiveness of models on reference datasets. The current paper concentrates on flavors of text classification, but does not provide a comprehensive overview of recent work motivated by task-specific considerations. Where applicable, the choice of baselines and reference models however reflects closest state-of-the-art in terms of accuracies and performance.

More relevant to the work here are methods that have general applicability, but for whom empirical evidence is provided on NLP specific tasks and datasets. These could be intuitions about model choices or training regimes. As an example, unsupervised data augmentation methods combined with Bi-directional Transformer Networks (Devlin et al., 2018) provide the best performance on a number of NLP tasks - sentiment classification on IMDB movie reviews , topic classification on DBpedia (Lehmann et al., 2015), etc.

In this spirit, flavors of RNNs have extensively been used in NLP tasks because of their strong performance in processing sequential data. Of these, LSTM units (Hochreiter and Schmidhuber, 1997) have proved to be successful in a variety of language modelling tasks - classification, machine translation and text generation. These RNN variants, and specifically LSTM-based models are currently the most effective in many NLP applications. They represent a difficult to beat benchmark on topic classification on AG-News dataset, question classification on TREC-6, and related tasks (Sachan et al., 2018; Howard and Ruder, 2018; Zhou et al., 2016; Wang, 2018). This is an area of active work, for example, AWD-LSTM introduced by (Merity et al., 2017), utilizes effective regularization to outperform other deep recurrent models.

Although we achieve a gain performance but this comes at the cost of increased training and execution times. The alternative are CNNs (Collobert et al., 2011), a model class more common for other modalities like images. They utilize convolution filters on sliding windows for a text sequence and applied a convolution operation to capture the most useful local features. Again, there is active work aimed at increasing the efficacy of these CNN models.

For example, (Kim, 2014) adopted multiple filters with different window sizes to extract multi-scale convolutional features for text classification. Similarly, to capture word relations of varying sizes, (Kalchbrenner et al., 2014) proposed a dynamic k-max pooling mechanism.

Recently there have been comparisons between RNN and CNN architectures as to which is better suited for sequence modelling. (Bai et al., 2018) conduct a systematic evaluation of generic convolutional and recurrent architectures for sequence modeling. The key idea they posit is that CNN should be regarded as a natural starting point for sequence modeling tasks due to various advantages such as parallelism and less training time as compared to RNN's. Other recent works have aimed to combine aspects of RNN and CNN architectures. The authors of (Zhou et al., 2015) propose the stacking of CNN and LSTM in a unified architecture for semantic sentence modeling.

Closest to the current work is (Geras et al., 2015), where the authors show that even more accurate CNNs can be trained under the guidance of LSTMs for certain speech recognition tasks as the two architecture have complementary inductive biases (Hinton et al., 2015). We are also motivted by recent work by (Yin et al., 2017) that shows that for NLP tasks, the knowledge gained by LSTM and CNN architectures are complementary in nature. Thus, the method due to (Geras et al., 2015) can also be extended to NLP tasks, and this is the focus of the current paper.

3 Method

3.1 Model Blending

As mentioned in the previous section, there has been some recent work which suggests that RNNs and CNNs provide complementary information when trained on the same data. RNN computes a weighted combination of all words in the sentence, while the CNN extracts the most informative ngrams for the relation and only considers their resulting activations. Given this, methods to improve the performance of one architecture, by incorporating the complementary knowledge of the other architecture are of interest.

One naive way to do this is ensembling, i.e. by combining the predictions of the two models as follows:

$$p_{ensemble}(c|s) = \gamma p_{RNN}(c|s) + (1-\gamma) p_{CNN}(c|s) \quad (1)$$

Here, $p_*(c|s)$ is the probability of a particular class (since we focus here on classification tasks) represented by c given the sentence s. While this potentially increases accuracy if the RNN and CNN models provide orthogonal information (as we expect them to in our case), there are downsides. Mainly, the ensemble is inefficient at test time as it requires us to hold both the models (RNN and CNN) in memory and score using them both.

We propose an alternative which is inspired from knowledge distillation (Hinton et al., 2015). To transfer the knowledge of one model (Teacher) into another (Student), we can use a training objective that combines the loss on the hard labels from the training data with a loss function which penalizes deviation from predictions of the teacher. That is, we optimize:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{soft} + (1-\lambda) \cdot \mathcal{L}_{hard} \quad (2)$$

$$\mathcal{L}_{soft} = H(p_T(c_i|s), q_S(c_i|s)) = -\sum_i p_T(c_i|s) \cdot \log q_S(c_i|s) \quad (3)$$

$$\mathcal{L}_{hard} = H\big(q_S(c_i|s), c_i^{true}\big) = -\sum_i q_S(c_i|s) \cdot \log c_i^{true} \quad (4)$$

Here, the loss is for one data point (sentence s), with p_T and q_S being the output of the Teacher and Student models respectively. $\mathcal{L}_{soft}$ is the cross-entropy difference between the predictions of teacher(soft labels) and the student. $\mathcal{L}_{hard}$ is the cross-entropy difference between the ground truth(hard labels) and the predictions of the student. The coefficient $\lambda \in [0, 1]$ controls the weights of the error on soft and hard labels. Thus, the final training objective is:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{soft} + (1-\lambda) \cdot \mathcal{L}_{hard} \quad (5)$$

$$\mathcal{L}_{soft} = H(p_{RNN}(c|s_i), q_{CNN}(c|s_i)) = -\sum_i p_{RNN}(c_i|s) \cdot \log q_{CNN}(c_i|s) \quad (6)$$

$$\mathcal{L}_{hard} = H\big(q_{CNN}(c_i|s), c_i^{true}\big) = -\sum_i q_{CNN}(c_i|s) \cdot \log c_i^{true} \quad (7)$$

In principle we can use both the combinations, i.e., RNN as Teacher and CNN as Student and vice-versa, but having the CNN as student would be beneficial in terms of inference time. This is due to the fact that parallel computation is possible for CNNs but not for RNNs. Thus in this work, we chose to evaluate the utility of distilling the knowledge of RNN Teachers into CNN Students. As indicated in (Geras et al., 2015), since the Teacher and Student may not differ in terms of model capacity (number of parameters), this method is referred to as *model blending* rather than distillation.

3.2 Teacher: RNN

We experiment using two LSTM networks as the teacher. The first, i.e. Bi-Directional LSTM is a tough baseline to break for most NLP tasks.The sentence $S = (x_1, x_2, ..., x_m)$, where m is the length of the sentence is taken as input and then we use glove embedding (Pennington et al., 2014) of size 300 to embed the words of the sentence.

At each time step, the word embedding $e(x_t)$ at that time step is passed through the forward and backward LSTM units to produce two hidden states $(\overrightarrow{h_t}, \overleftarrow{h_t})$ respectively. The last hidden state of the forward $\overrightarrow{h_T}$ and backward network $\overleftarrow{h_T}$ are concatenated and passed into a fully connected network to predict the outputs. This forms a baseline RNN teacher for our experiments. Detailed equations are as under:

$$\overrightarrow{h_t} = f_1(\overrightarrow{h_{t-1}}, e(x_t)) \quad (8)$$

$$\overleftarrow{h_t} = f_2(\overleftarrow{h_{t-1}}, e(x_t)) \quad (9)$$

$$p(y|S) = \sigma(W[\overrightarrow{h_T}, \overleftarrow{h_T}] + b) \quad (10)$$

f_1 and f_2 are the functions as in a LSTM unit.

The second LSTM network we use is the AWD-LSTM (Merity et al., 2017) architecture and employ the same training regime mentioned in (Howard and Ruder, 2018) namely discriminative fine-tuning, gradual unfreezing of layers and slanted triangular learning rates which achieve state of the art performance for these tasks. This ensures that our Teacher models represent current best practices in terms of producing very effective models tailored for specific tasks. Our experiments show that these Teacher models exhibit performance levels that the Student models are unable to beat. However, the Student models - whose advantages are described next - are able to bridge the accuracy gap.

3.3 Student: CNN

We use the CNN architecture for text classification due to (Kim, 2014) as the student. Each word x_i in the input is embedded into a k dimensional word vector $e(x_i)$. A sentence of length n which is padded when necessary is represented by a kXn matrix as:

$$S_{1:n} = [e(x_1), e(x_2), ..e(x_n)] \quad (11)$$

$S_{i:j}$ represents the sub-matrix $[e(x_i), e(x_{i+1}), ..e(x_j)]$ A convolution involves a filter $\boldsymbol{w}$ of dimension hXk applied to a window of h words to produce a feature. A feature f_i is generated from $S_{i:i+h-1}$ as follows:

$$f_i = g(\boldsymbol{w}.S_{i:i+h-1} + b) \quad (12)$$

where b is the bias term and g is any non linear function such as $tanh$. This filter is applied to each possible window to produce a feature map:

$$\boldsymbol{f} = [f_1, f_2, ...f_{n-h+1}] \quad (13)$$

Thereafter, max over time pooling is applied to obtain the most important feature f', i.e. $f' = max(\boldsymbol{f})$. In this way the model uses multiple convolutional filters to obtain multiple features. These features are then passed to a fully connected layer followed by softmax to predict the probability of the classes. More details regarding the hyperparameters used in experiment are mentioned in the experiment 4.3.2.

4 Experiments

In this section, we conduct a thorough empirical validation of the method previously described. We cover a range of text classification tasks represented by their corresponding benchmark datasets which are described next.

4.1 Datasets

The proposed model is tested on four datasets. The Summary statistics for these are provided in Table 2.

1. `IMDB dataset`[1]:Sentiment Classification Dataset (Maas et al., 2011). This is a large scale dataset for binary sentiment classification (positive and negative) of movie reviews.

2. `TREC-6`[2]: Question Classification Dataset (Li and Roth, 2002). This task involves classification of question into 6 question types (abbreviation, description, entity, human, location, numeric value).

3. `AG-News`[3]: Topic Classification Dataset (Zhang et al., 2015). This task involves classification of news articles,we choose the 4 largest classes from this corpus to construct our dataset, using only the title and description fields.

4. `DBpedia`:Topic Classification Dataset (Lehmann et al., 2015; Zhang et al., 2015).This involves classification of Wikipedia articles in 14 categories.

4.2 Pre-processing

We utilize same pre-processing techniques utilized in (Johnson and Zhang, 2017; Howard and Ruder, 2018). We add special tokens for uppercase letters, following which we convert the uppercase letters to lowercase for consistency - this allows us to explicitly incorporate the effect of capitalization. We also add tokens for repetitions, along with special tokens to indicate the *start* and *end* of document. This allows us to capture aspects from the document that are pertinent to text categorization.

4.3 Experimental Details

The proposed workflow has constituent components that can be evaluated individually.

[1] https://ai.stanford.edu/~amaas/data/sentiment/

[2] http://cogcomp.org/Data/QA/QC/

[3] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

Table 1: Dataset summary statistics. K: Number of classes. l: Average length of a document in the training set

Dataset	IMDB	TREC-6	AG	DBPedia
Training	25k	5452	120k	560k
Testing	25k	500	7600	70k
K	2	6	4	14
l	234	11	45	55

4.3.1 Embedding

We perform two sets of experiments. For the first experiment we adopted the fixed size word vectors from (Pennington et al., 2014). In particular, our experiments utilize the GloVe embedding [4] trained on 6 billion tokens of Wikipedia 2014 and Gigaword 5. For the words which are absent in GloVe we average the vector representations of 8 words around the word in training dataset as its word vector as suggested by (Wang and Jiang, 2015). The word embedding are then fine-tuned during training specific to task to boost performance. The second set of experiments, instead of using fixed length word vectors, we pre-train the model (referred to as the *language model*) on Wikitext-103 consisting of 28,595 pre-processed Wikipedia articles and 103 million words utilizing *Discriminative learning rates* and *Gradual unfreezing of layers* introduced by (Howard and Ruder, 2018) to encode the document and then we perform task specific fine-tuning to further improve the classification performance. This is analogous to training/fine-tuning model in Computer vision, where we provide a warm start to the model by pre-training the model on much larger dataset.

4.3.2 Training Setting

We train the state of the art AWD-LSTM Teacher with an embedding size of 400, 3 layers, 1150 hidden activations per layer, and a BPTT batch size of 70. We apply different dropout to different types as suggested by (Merity et al., 2017). The student CNN network has filter sizes of 3, 4 and 5. We pad the documents which have length less the 3, so that convolution operation is possible over the sentence. The models were implemented in PyTorch and were adapted from prior works (Howard and Ruder, 2018; Kim, 2014). The CNN trained using the proposed method outperforms the CNN architecture and the teacher network and is 15x more computationally efficient from its teacher. The experiments were first performed on IMDB dataset for sentiment analysis to tune γ and λ hyperparameter to their optimal values of 0.3 and 0.5 we then validate these parameter using the TREC-6 data set for question classification.

For all our experiments we use Adam with default setting $\beta_1 = 0.9$ and $\beta_2 = 0.99$ with a batch size of 64 and we follow cyclic learning rate update. We train our model until we observe a drop in the performance.

4.4 Baselines

We compare our method to several popular models in our experiments :

1. *fastText* : Linear classifiers are often considered as strong baselines for text classification problems (Joulin et al., 2016). Here we generate bi-grams of the sentence and embed the original words as well as bi-grams into 100 dimensional fixed vector which then use for classification.

2. *CNN* : We use CNN architecture inspired by (Kim, 2014) and train it using various teacher which are recurrent in nature. We further compare our CNN trained through blending process to other convolutional neural networks viz.character level CNN (Zhang et al., 2015) referred to as c-CNN and very Deep CNN network (Conneau et al., 2016) referred to as b-CNN.

3. *RNN*: We use AWD-LSTM (Merity et al., 2017), Bi-Directional LSTM (Bi-LSTM) (Zhou et al., 2016) as our teacher model. We empirically validate that by using a better performing teacher model we can improve the performance of our student network. We leave exploration of the more efficient teacher network for different tasks to future work, but expect that they would boost performance.

5 Results

5.1 Overall Performance

Through our experiments shallow networks lead to only a slight drop in performance over deeper alternatives. In this scenario, our choice of shallow CNN networks is primarily driven by two fac-

[4] https://nlp.stanford.edu/projects/glove/

Table 2: Classification results on several standard benchmarks. each column represents accuracy of classification which has been averaged over three random seeds.

Model		IMDB	TREC-6	AG	DBPedia
Linear	Fasttext (Joulin et al., 2016)	85.22	88.2	92.6	98.5
RNN	LSTM	82.77	90.2	82.77	-
	Bi-LSTM (Zhou et al., 2016)	84.11	93.0	93.0	99.12
	AWD-LSTM (Merity et al., 2017)	95.4	96.4	95.0	99.20
CNN	a-CNN (Kim, 2014)	88.22	92.0	91.6	98.6
	b-CNN (Conneau et al., 2016)	-	93.0	91.3	98.4
	c-CNN (Zhang et al., 2015)	-	93.2	90.49	98.3
Ensemble	CNN + LSTM	88.63	93.15	92.0	99.0
	CNN + AWD-LSTM	95.4	96.7	95.0	99.23
Blended*	CNN (T:LSTM)	87.23	91.08	92.3	99.14
	CNN (T:AWD-LSTM)	93.2	92.34	92.5	99.1

Table 3: Comparison on the execution time of the methods is execution time compared to the our proposed CNN training using RNN teacher on IMDB dataset.

Model		Execution Time
Linear	Fasttext	1.1x
RNN	LSTM (Tai et al., 2015)	5.55x
	Bi-LSTM	5.64x
	AWD-LSTM	15.5x
CNN	a-CNN	1.0x
	b-CNN	4.3x
Ensemble	a-CNN + LSTM	6.2x
	a-CNN + AWD-LSTM	16.7x
Blended*	a-CNN(T:LSTM)	1.0x
	a-CNN (T:AWD-LSTM)	1.0x

tors: *interpretablility* and *ease of deployment* to low resource devices.

The CNN trained through the proposed method is comparable to its Teacher LSTM network. However, they are significantly faster at test time thereby making deployment to resource constrained environments (e.g. mobile devices) more feasible. We are therefore validating the work of (Hinton et al., 2015) where *'dark knowledge'* from a trained complex network can be transferred to a much simpler model which is computationally efficient at inference/scoring time. We have shown that this is possible not only amongst the same family of models but also across model families - i.e., distillation of learnt knowledge from recurrent to convolutional models.

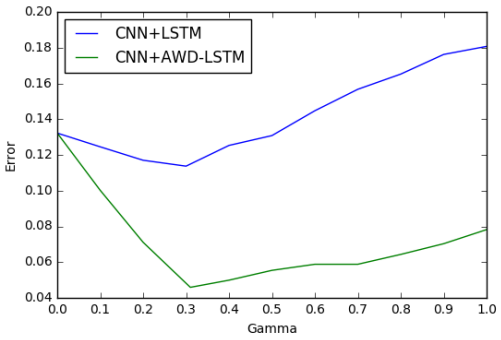

Figure 1: Error as function of γ

5.2 Model combinations

We evaluate two methods aimed at leveraging the two alternative model families. The first approach we use is to combine the two models into an ensemble. The results in 1 indicate that it is possible to identify a value of the mixing parameter γ that provides improvements over either extreme. The second method is to utilize the Teacher-Student workflow which, as shown in Table 2, provides better improvements than emsembling. We show in 2 that the trade-off hyperparameter λ allows controlling the effect of training the Student alone versus re-using the knowledge from the Teacher.

5.3 Insights

With respect to the baselines utilized in the current paper, multiple comparisons are informative:

- Methods specifically optimized for speed (here Fasttext) might give up too much accu-

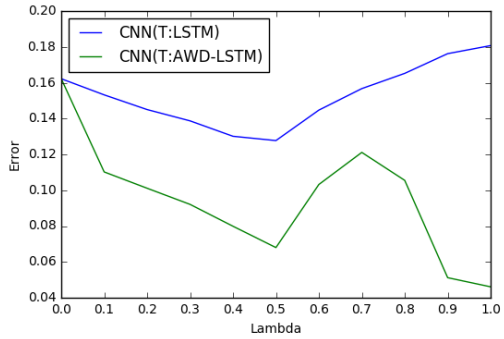

Figure 2: Error as function of λ

racy, whereas the pursuit of increased performance leads to efficiency hits (here the RNN variants)

- The CNN models (3 options were evaluated in Table 2) have speed benefits. Within this model class, increased model complexity might provide better accuracies. The slowest evaluated CNN model ("b-CNN") has lower scoring times than the fastest RNN model (a vanilla LSTM-based network)

- We also provide LSTM, Bi-LSTM & AWD-LSTM alternatives as representatives of the RNN family. These provide increasing accuracies but unfortunately at the cost of scoring time. Note that on the settings evaluated here, these RNN models are often significantly better than the CNN cousins

- Though the CNN & RNN model classes are potentially diverse, ensembling as a method to leverage them both does not necessarily bring the expected accuracy improvements. It also suffers from even lower scoring times

- The Blended CNNs by definition would have equivalent scoring times to the equivalent source Student models. They do however provide accuracy benefits over the Student, but they are currently unable to reach the levels of the best Teacher models

While the work described here applies blending to text classification, we believe that the proposed training workflow can be applied to improve performance of any model class pairs that represent different points on the accuracy and efficiency trade-off. This is best represented in our experimental results by the improved performance of shallow CNNs by incorporating LSTM priors in the training objective.

Note that while we highlight the scoring time benefits of the proposed workflow, the Teacher-Student pipeline leads to a significantly increased training time. This is because we have to first fit the Teacher before transferring the learnt knowledge to the Student. While the benefits of following this methodology are present in the results provided, investigating alternative regimes that also address training time considerations are of interest and will be the subject of future work.

6 Conclusion

In this paper, we present a method for training accurate CNNs for text classification tasks by using state-of-the-art LSTM priors to guide the training process. Following recent work in related fields, we refer to the LSTM reference models as "Teachers" whose knowledge is to be transferred into the "Student" CNN models. In addition to the normal loss-function, i.e. the difference between the ground-truth (hard labels) and the predication of the model, we include a term that penalizes the model according to the difference between prediction of the teacher LSTM(soft-targets) and the prediction of the model. The resulting model performs comparable to the benchmark models in text classification on four benchmark datasets and is significantly faster at inference time. Exploring methods that allow the training of models that are more effective accuracy and efficiency trade-offs will be the subject of future work.

References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

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(Aug):2493–2537.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Krzysztof J Geras, Abdel-rahman Mohamed, Rich Caruana, Gregor Urban, Shengjie Wang, Ozlem Aslan, Matthai Philipose, Matthew Richardson, and Charles Sutton. 2015. Blending lstms into cnns. *arXiv preprint arXiv:1511.06433*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

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*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Devendra Singh Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. 2018. Revisiting lstm networks for semi-supervised text classification via mixed objective function.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Ellen M Voorhees and Dawn M Tice. 1999. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82.

Baoxin Wang. 2018. Disconnected recurrent neural networks for text categorization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2311–2320.

Shiyao Wang, Minlie Huang, and Zhidong Deng. 2018. Densely connected cnn with multi-scale feature attention for text classification.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.