Ensemble Deep Learning for Multiclass Text Classification

Vaibhav Jain
DTU/2K16/SE/087

We studied the performance of an ensemble method called stacking on the performance of deep neural network (DNN) architectures in multiclass text classification. We considered two DNN variants: Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). We also studied the effect of the dynamicity of pre-trained word embeddings on these architectures.

Introduction

Supervised multiclass text classification can be formally defined as the problem of assigning a class label to a sequence of words, i.e. a document. We are given a set of training examples (x_i, y_i) where $x_i \in \mathbb{X}$ (the *document space*) and $y_i \in \mathbb{C}$ ($\mathbb{C} = \{c_1, c_2, ..., c_n\}$ is the set of classes). We then aim to learn a classifier $f: \mathbb{X} \rightarrow \mathbb{C}$ which can assign a class to unseen documents (Manning et al., 2008).

The simplest approach to text classification is to use a *bag-of-words* model approach in which the word frequencies are passed as input to linear classifiers such as Naïve Bayes or Support Vector Machine (SVM). However this approach suffers from the lack of word order and grammatical structure, along with data sparsity in case of a small training set (Sachan et al., 2018). Neural network models such as *Convolutional Neural Network* (CNN) and *Long Short-Term Memory* (LSTM) have recently shown good results (Kim, 2014; Johnson and Zhang, 2016), yet they haven't achieved the same success they did in other fields such as computer vision (Conneau et al., 2016).

Deep neural network architectures usually suffer from overfitting due to their high representational power (Srivastava et al., 2014). A popular technique to resolve this is *emsemble learning*, which aims at learning a *meta-classifier* by combining several classifiers (called *base-level classifiers*) (Sikora and Al-Laymoun, 2014). We use a particular technique called *stacking*, which uses the output generated by the base-level classifiers as the input to the meta-classifier. The meta-classifier combines the individual outputs to generate the final output. It has been observed that the meta-classifier usually performs better than all of the base-level classifiers. (Dzeroski and Zenko, 2004)

In the present work, we have made use of the "News Category Dataset" (Misra, 2018), which is a collection of the headlines and short description of 2 lakh Huffpost articles, categorized into 40* classes. We learned four different base-learners: CNN-static, CNN-dynamic, LSTM-static, and LSTM-dynamic. The static and dynamic models differ only by the trainability of the word embeddings. We then combined these base-learners into an artificial neural network based meta-classifier.

Data Preprocessing

We converted each number in the training corpus to a "num" tag, and then performed *stop word* removal and *lemmatization* using the spaCy library (Honnibal and Montani, 2017).

Stop Word Removal

Stop words refer to extremely common words that serve little value in the classification process. (Manning et al., 2008) Examples include "in", "they", "which", etc. Stop words are generally removed in NLP applications. We have used the stop words list provided by spaCy which includes 305 such words.

Lemmatization

In a natural language like English, words may have various inflected forms and derivationally-related words. *Lemmatization* refers to converting such words into their *canonical* form called *lemma*. Lemmatization relies on complete morphological analysis and finds the lemma of a word on the basis of its intended meaning. This involves identifying the part of speech and the context. Lemmatization differs from *Stemming* which removes commonly used prefixes and suffixes from a word, and can result into wrong results (Manning et al., 2008).

Model

Word Embeddings

We have word embeddings of dimension 200 as the starting layer for all of our base-classifiers. We used pre-trained GloVe (Pennington et al., 2014) embeddings which has a vocabulary size of 4 lakh, and was trained on 5 billion tokens from the 2014 Wikipedia dump and the English Gigaword Fifth Edition. They were used to initialize the embedding matrix of both static and dynamic models, however the dynamic models were allowed to make changes to them during backpropagation whereas the static models weren't.

*The dataset originally contains 41 classes; however we merged two classes "Worldpost" and "The Worldpost" as they are completely similar.

The embeddings for out-of-vocabulary words were initialized in such a way that their mean and standard deviation is equal to that of the overall GloVe embedding matrix.

Convolutional Neural Networks

The model architecture for the CNN classifiers is partially inspired from Kim (2014). We used 256 one-dimensional convolutional kernels of size 2, 3, and 5 each. Xavier uniform initializer was used to initialize the kernel weights. We then employed *batch normalization* in order to regularize the model. This step applies a transformation to change the mean and standard deviation of the layer outputs to zero and one respectively for each batch. We then applied *rectified linear unit* (ReLU) as the activation function, followed by global max-pooling. We also tried global average-pooling under the influence of Lin et al. (2014), but observed extreme overfitting.

We employed *dropout* (Srivastava et al., 2014) with a 0.5 keeping rate for regularization purposes, and then the outputs are passed to a dense layer of size 512 and ReLU activation. We then applied the output layer with softmax activation.

Long Short-Term Memory

We used two LSTM layers of output dimensionality 500 and 200 each. The first layer returns the generated sequences for all units, whereas the second one returns the sequence generated by the last unit only. Each LSTM layer is followed by a dropout layer with 0.5 keep rate. *Hyperbolic tangent* (tanh) was used as the activation function. Xavier uniform and orthogonal initializers were used to initialize the kernel and the recurrent kernel weights respectively.

Meta-classifer

The extended part of the meta-classifier is a neural network with a single hidden-layer of size 100 and ReLU activation. The output layer has softmax activation.

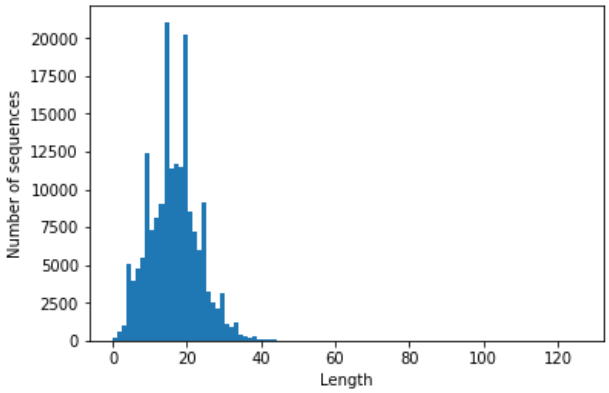
Experimental Setup

We used Keras for implementation purposes.[†] We divided our dataset into training set, holdout set, and test set in the ratio of 7:2:1.

Sequence length

After the data preprocessing steps and tokenization, the distribution of the sequence lengths is as shown in figure 1. The maximum sequence length is 126, however we did not find it suitable to build models to accommodate such large sequence lengths as most of the sequences are much smaller. Therefore, we kept the sequence length as 50 and post-truncated any sequence larger than that. Truncation was not required for more than 99.88% of the sequences.

Figure 1. Sequence length histogram



Padding

For sequences smaller than the defined sequence length, we padded them with the word embedding corresponding to "unknown word". This embedding was initialized and trained in a manner similar to that of out-of-vocabulary words. We used post-padding for CNNs and pre-padding for LSTMs.

Training

We trained the base-level classifiers using the training set. We then generated their predictions on the holdout set, and used them as input to the meta-classifier. The meta classifier, along with the base-level classifiers, were then tested on the test set. All models were trained by optimizing for the categorical crossentropy through the *Adam* optimizer with a mini-batch size of 128.

Results and Discussion

We have used accuracy and top-3 accuracy as our evaluation metrics. The evaluation results generated by our study have been summarized in Table 1.

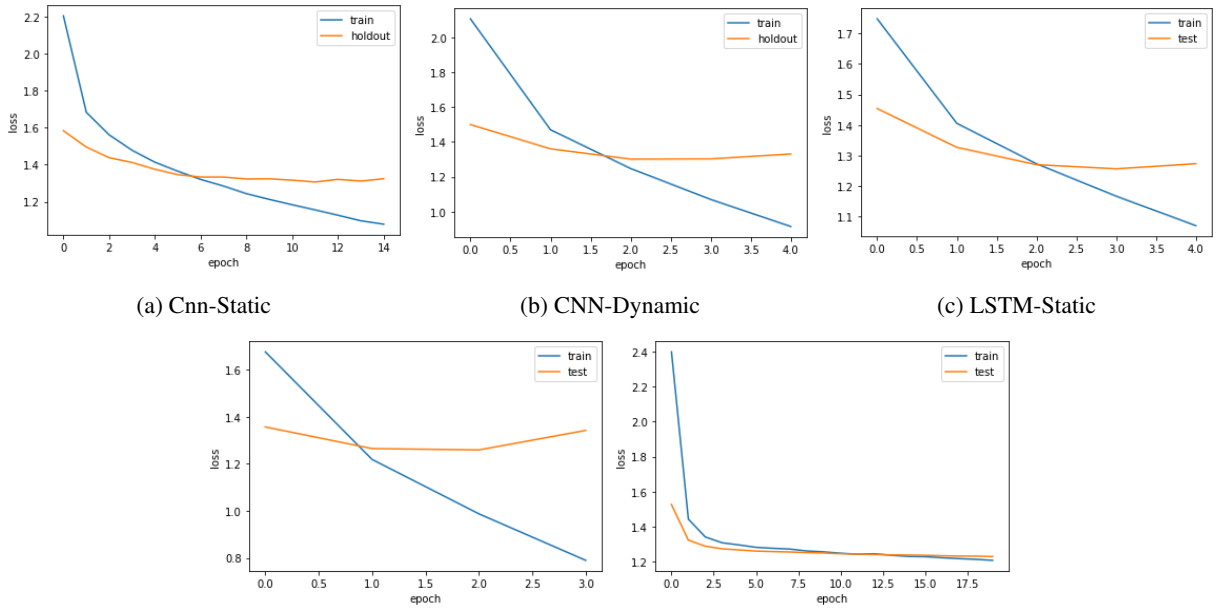
We also made the following observations during the study:

1. Ensemble learning leads to a considerable increase in the accuracy, but not in top-3 accuracy. It is to be tested if specifically designed top-k loss functions (Berrada et al., 2018) can lead to better top-3 accuracy results.

2. We had to apply early-stopping in the case of CNN-Dynamic, LSTM-Static, and LSTM-Dynamic. The learning curves for all base-level and meta classifiers can be seen in figure 2.

[†]The implementation code has been released on Kaggle under an open source licence. The code is divided into two parts: Stopwords removal and Lemmatization, and Ensemble Deep Learning Model.

Figure 2. Learning curves



(a) Cnn-Static (b) CNN-Dynamic (c) LSTM-Static

(d) LSTM-Dynamic (e) Meta-classifier

Table 1
Evaluation results

Model	Accuracy	Top-3 Accuracy
Convolutional Neural Networks		
CNN-Static	62.20%	84.57%
CNN-Dynamic	63.32%	84.63%
Long Short-Term Memory		
LSTM-Static	65.23%	**85.95%**
LSTM-Dynamic	64.57%	85.31%
Ensemble learning		
Meta-classifier	**67.24%**	85.82%

3. Dynamic model performed better than static ones in the case of CNNs but not in LSTMs.

4. We also tried post-padding for LSTMs but that resulted in degraded performance. This was expected as we use the sequence generated by the last LSTM unit as the input to the dense layer. Post-padding may *"flush out"* the hidden state of the network.

Conclusion

We observed that ensemble deep learning methods can outperform popular DNN models in text classification. Our model can be further improved upon through hyperparameter tuning. We believe that future work should be focused upon prevention of overfitting in the base-level classifiers.

References

Berrada, L., Zisserman, A., and Kumar, M. P. (2018). Smooth loss functions for deep top-k classification. *CoRR*, abs/1802.07595.

Conneau, A., Schwenk, H., Barrault, L., and LeCun, Y. (2016). Very deep convolutional networks for natural language processing. *CoRR*, abs/1606.01781.

Dzeroski, S. and Zenko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54:255–273.

Honnibal, M. and Montani, I. (2017). spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Johnson, R. and Zhang, T. (2016). Supervised and semi-supervised text categorization using lstm for region embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 526–534. JMLR.org.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Lin, M., Chen, Q., and Yan, S. (2014). Network in network. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Misra, R. (2018). News category dataset.

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

Sachan, D. S., Zaheer, M., and Salakhutdinov, R. (2018). Investigating the working of text classifiers. *CoRR*, abs/1801.06261.

Sikora, R. and Al-Laymoun, O. (2014). *A modified stacking ensemble machine learning algorithm using genetic algorithms*, pages 43–53.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.