News Classification Model Final Report

Xiaobo Zhang, Yuhao Zhang
Stanford CS230 Students
{bobz, njzyh}@stanford.edu
Github Link: https://github.com/bobzhang199102/news_classifier_with_birnn.git

Abstract

This report presents a design of a news classification system based on various ML models. We experimented Bidirectionally with RNN and MT-DNN with different sets of hyperparameter tunings. The bidirectional RNN was implemented with Tensorflow on AWS ML platform and the MT-DNN is using Pytorch within a Docker environment. The results show an accuracy of 0.6028 for bidirectional RNN and an accuracy of 0.3349 for MT-DNN.

1 Introduction

Newsfeed applications are growing exponentially in the big data area. Even though there are a lot of news to explore, there are challenges to find news that meet the users' interest. One of the major tasks for those intelligent applications is to recommend personalized feeds to users. The core of a recommendation system is usually a ML-based model that categorizes news feeds based on its content and metadata to make recommendations based on a user activities. This project proposes a news content classifier that classifies text news content that sets predefined labels. The input to the system is text format news content (a combination of title and content) and the output is one of the predefined labels of the news categories.

2 Related Work

The exploration of text classification has a long history. The challenges residing inside the text comprehension is the key to understanding our language and demonstrate the potential of machine learning. This research domain has been gaining increased popularity recently [1, 2] due to many newly developed algorithms, tools, and techniques.

Many previous research and algorithms have been evaluated for similar tasks [3, 4, 5]. Previous findings have created a solid foundation, enabling us to apply some newly invented algorithms, such as MT-DNN [7], to solve text classification problems. We also benefit from more powerful tools, like Tensorflow [8] and more comprehensive word embedding techniques.

In this project, we apply both the traditional bidirectional RNN [6] and the pioneering MT-DNN [7] to the news classification problem. We used the GloVe [9] embedding for the RNN and BERT [10] for MT-DNN. Their performance is analyzed.

3 Dataset and Features

The dataset is General News Category Dataset from Kaggle. The Json file contains 202,372 records and a total of 41 distinct categories. Since we have a relatively large amount of data and the median amount

of categories, we used 10% of the data for validation and 10% for the test. Each record contains the following attributes:

- category: Category article belongs to
- headline: Headline of the article
- authors: Person authored the article
- link: Link to the post
- short_description: Short description of the article
- date: Date the article was published

The following is an example:
{"category": "CRIME","headline": "There Were 2 Mass Shootings In Texas Last Week, But Only 1 On TV", "authors": "Melissa Jeltsen", "link":"https://www.huffingtonpost.com/entry/texas-amanda-painter-mass-shooting_us_5b081ab4e4b0802d69caad89", "short_description": "She left her husband. He killed their children. Just another day in America.", "date": "2018-05-26"}

For bidirectional RNN with attention, we use the following mechanisms for preprocessing of the data:

- Gensim for vector space modeling.
- NLTK for text data processing.
- GloVe for word embedding.

For MT-DNN, we apply the author published code (https://github.com/namisan/mt-dnn) to our dataset. We still use most of the above libraries to preprocess our data but the code training model uses BERT instead.

4 Methods

4.1 Bidirectional RNN

Format and Clean Data: we connected “title” and “short_description” together because we usually see the title carries equally important information as a description. The connected texts are tokenized by NLTK library. Then we filtered out unusually short ones (less than 5 words) from the results to clean up abnormal data. We use Keras pad_sequences to pad all sequences to 50 for the batch process. We also clean up some non-alphabetical characters, converted sentences to lower case, etc. to ease the training.
Generate Integer Label from Category Attribute: to make the training easier, we mapped each category to an integer value to simplify the output evaluation.
Split Dev, Validation and Test Dataset: we use the sklearn train_test_split function to split the dataset to 90% dev and 10% validation. Note that the test portion was already manually removed previously.
Embedding Layer: we use the pre-trained GloVe for word embedding. It provides a dense vector representation for each word so that similar words are treated similarly instead of spreading over a sparse matrix.
Bidirectional RNN Layer: we use two LSTMs to handle the text from beginning to the end and the reverse. Each of the LSTM is also wrapped with a dropout layer to prevent overfitting. The softmax output of the RNN layer goes through an attention layer to help the model understand the context of each component.
Fully Connected Layer: the output layer is a fully connected layer to produce one of the 41 categories of the news content.

Calculate Loss: after each epoch iteration, we calculate the loss with tensorflow sparse_softmax_cross_entropy_with_logits loss function. We also apply Adam optimizer to minimize the loss.

Hyperparameters: Different values of hyperparameters have been tried to achieve the best accuracy in the models and the best hyperparameters are as follows:

- Learning Rate: 1e-3
- The number of units in the LSTM cell: 100
- Embedding size: 300
- Dropout: 0.8

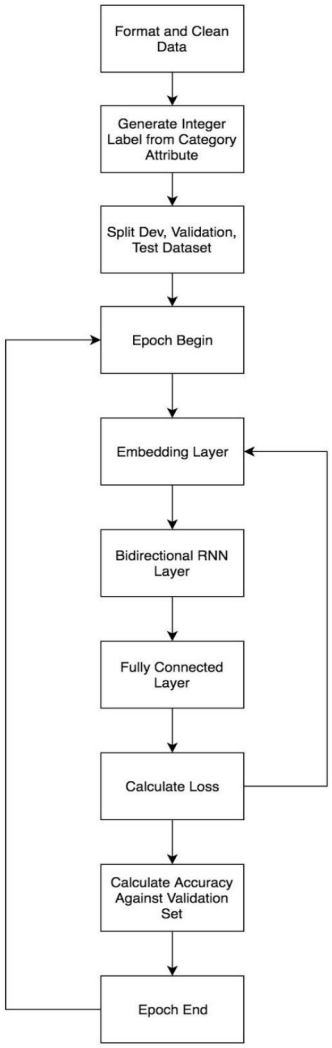

Figure 1. Bidirectional RNN with attention implementation

4.2 MT-DNN

MT-DNN is a novel NLP model leveraging BERT [10]. They are both trained via pretraining and fine-tuning stages. The key difference is that MT-DNN uses Multitask Learning in the fine-tuning stage with four types of task-specific layers. Those layers perform various NLP tasks and combine the results to a final prediction. They all share the same lower level layers, which are responsible for major embedding tasks.

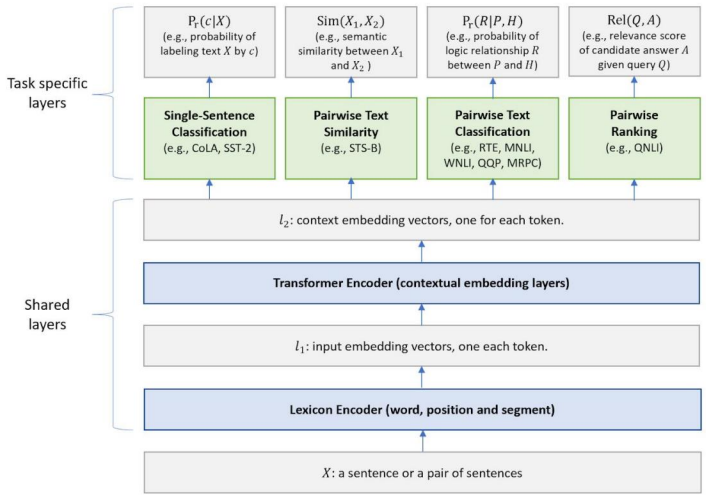

Figure 2. This diagram is from [7] to show the architecture of MT-DNN.

4.2.1 Implementation

For most of the parts, we apply the author published code (https://github.com/namisan/mt-dnn) to our dataset, with the following minor modifications:

1. We replaced the train and test dataset with our own inputs and labels, which uses the same mechanism in the previous bidirectional RNN approach to clean up and filter data.
2. We added one more fully connected layer to convert the output to the desired 41 category integers.

We tried to use the same hyperparameters introduced in the original paper [7], which reflected a learning rate of 5e-5, a batch size of 32, a maximum number of epochs of 5, a dropout rate of 0.1, a clipped gradient norm within 1. However, we ran the model in docker on our personal Macbook. We had to turn off the GPU acceleration option and we had to downsize the attention layer size from the default of 128 to 32. The max sequence length was also reduced from the default 512 to 64 given that the news data was typically short.

5 Experiments/Results/Discussion

5.1 Bidirectional RNN

After tuning of hyperparameters, the best accuracy achieved for the Bidirectional RNN model was 0.5453.

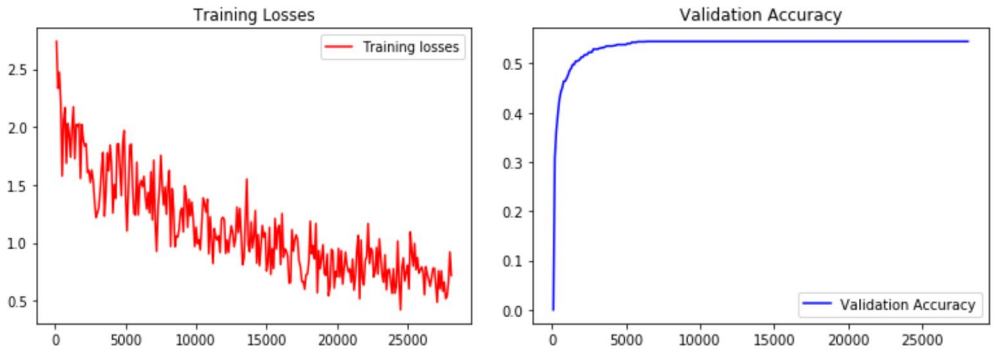

Figure 3. Training Losses and Validation Accuracy of Bidirectional RNN.

5.2 Bidirectional RNN with Attention Results

After the tuning of hyperparameters, the best accuracy achieved for the Bidirectional RNN model was 0.6028.

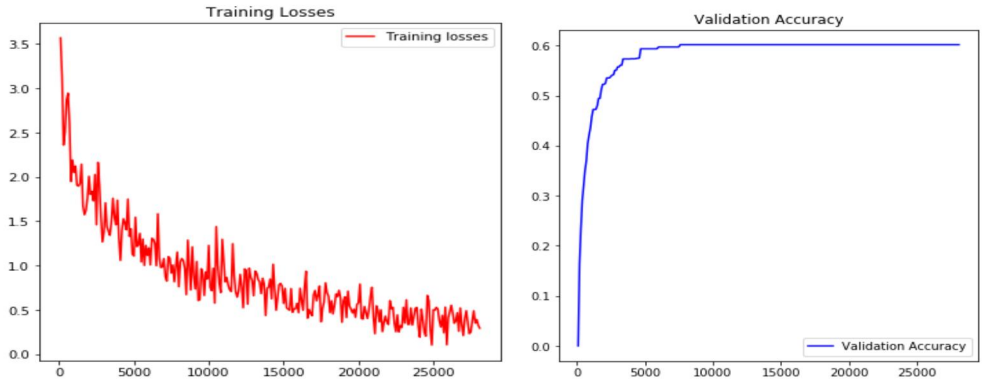

Figure 4. Training Losses and Validation Accuracy of Bidirectional RNN with Attention

5.3 MT-DNN Results

Due to the resource limitation and docker virtual machine limitation mentioned above, we had to run the model in a degraded mode. The best accuracy in test dataset was 0.3349 and 0.3760 when training the data set. Most of the wrong results were concentrated on Crime related news.

5.4 Discussion

The Bidirectional RNN with Attention has better performance.

MT-DNN wrong predictions concentrate in the Crime category. The main reason might be that there was more special terminology in the Crime category than other categories.

6 Conclusion/Future Work

The Bidirectional RNN with Attention can achieve good accuracy, which is better than Bidirectional RNN. In addition, when more data was used for training the accuracy increased. Hyperparameters significantly impacted the accuracy of the models, so tuning hyperparameters was essential.

We did not have enough time and computing resource to explore MT-DNN in greater depth. However, this is a novel and interesting algorithm. If we had time to migrate the original codes from local docker to AWS platform, we should have been able to run the algorithm with the same parameters in the paper and compare the algorithm with Bidirectional RNN fairly.

7 Contributions

Both team members have equal contribution to the project. Each person joined in the entire process of the project.

Reference

[1] Hingmire, S.; Chougule, S.; Palshikar, G. K.; and Chakraborti, S. 2013. Document classification by topic labeling. In SIGIR, 877–880.

[2] Lewis, D. D. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In SIGIR, 37–50.

[3] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[4] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329, 2014.

[5] I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. In ICML, 2011.

[6] Schuster, M., and Paliwal, K. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing 45, 11 (Nov. 1997), 2673–2681.

[7] Liu, X., He, P., Chen, W., and Gao, J. Multi-task deep neural networks for natural language understanding. CoRR, abs/1901.11504, 2019.

[8] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: A system for large-scale machine learning. Tech. rep., Google Brain, 2016. arXiv preprint.

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation

[10] R. Jeffrey Pennington and C. Manning. Glove: Global vectors for word representation. 2014.

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