Spam Classifier with CNN and Embedding

Haikang Deng, Yicheng Zou

May 2, 2021

1 Introduction

As one of the most classical tasks in machine learning, spam classification has been explored in decades with the help of Naïve Bayes and Logistic Regression model. Enlightened by the emergence of neural networks, we are motivated to revisit the spam classification problem with a new approach, combining the power of convolutional neural network and word embedding. Simply put, our model consists of three major parts, a word embedding, a convolutional layer, and a fully connected linear layer. As a result, our model achieves a 98.88% accuracy which is slightly higher than the existed linear classifiers. In addition, the model generates a spam specific word embedding of dimension 100 which can be generalized to other related tasks. A graphical principal component analysis is presented which shows certain patterns of the word embedding. In the following sections, we will demonstrate model's construction and analysis in detail.

2 Model

Here we propose a convolutional neural network to conduct spam-ham classification, which not only trains the weights of convolution filters, but also trains the word embedding that could be later used for spam detection on outer tasks.

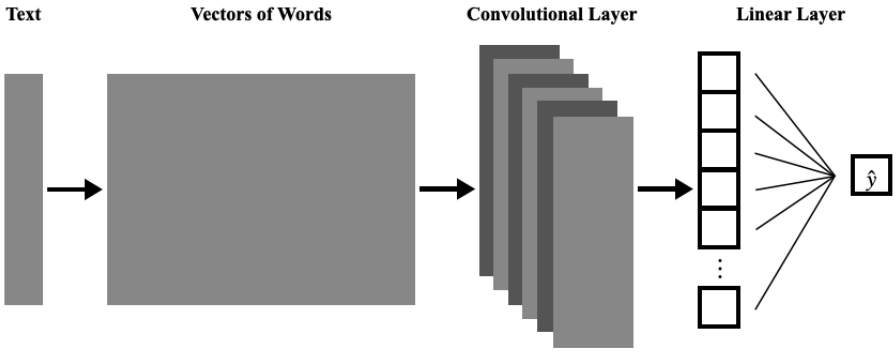

Figure 1: Model

The model first takes as input an SMS text. Note that any length is accepted by the model as text with fewer than 9 tokens will be padded to the minimum length. The model then turns every token into a vector of dimension 100 according to the word embedding, which is trained on every step of training. The word vectors matrix is fed to the convolutional layer and result in multiple convolved features, which are then concatenated and fed to the linear layer and used for final classification.

2.1 Embedding

A word embedding is a kind of representation that reflects some meaning of words. Classical word embeddings such as *GloVe* [1] and *word2vec* [2] are helpful in NLP tasks that they generate similar representation for words with similar meanings, from a more semantic point of view. Word embeddings are often fined-tuned to adapt to different kind of tasks in order to generate better performance.

Here, for spam detection, we propose that the model can perform well without pretrained embedding, based on the assumption that spam detection depends little on semantic or sentiment analysis [3]. Therefore, we start with a blank word embedding of dimension 100. The matrix of any given text input is of size $m * 100$, where m is the number of tokens in the text. By applying the *EmbeddingLayer* in pytorch, we allow gradients to flow backwards to the word embedding during training. In this case, we utilize the power of CNN and create an embedding specific to spam situations.

2.2 Convolutional Layer

In order to explore the correlation between tokens in the message, we introduce filters of width n = 2, 3, . . . , 9 to look for the occurrence of different n-grams in the text. Note that in 2d CNNs, especially in graphic models CNNs, filter's dimension usually varies in both length and width. However, in this NLP task, we only vary the width of the filters, keeping length fixed at 100, which allows the filter to retain all information for any given token. Thus, in our model, filters have dimensions $n * 100$, where n is width and $2 \leq n \leq 9$.

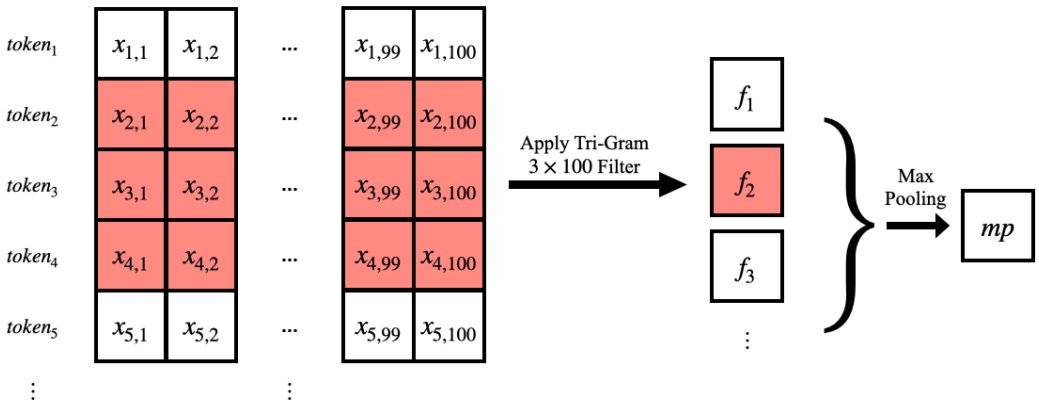

Figure 2: Convolution

Max Pooling is used to extract most important feature after every filter is applied to the word vectors. There are 512 filters for each of 8 different sizes, which means there are 4096 features extracted from the n-gram models. These features are concatenated and go through a fully connected linear layer to produce a classification result, in this case spam or ham.

3 Training Details

3.1 Data Preparation and Processing

We used the **SMS Spam Collection Dataset** [4] which contains total of 5572 valid messages, with 87% ham and 13% spam. The dataset is imported to torchtext from csv and then tokenized with *spaCy*. We split the entire dataset into training set, validation set, and test set with the ratio 0.70 : 0.15 : 0.15.

3.2 Parameters

We employed the loss function *BCEWithLogitsLoss* provided by pytorch. It combines sigmoid layer and binary cross-entropy loss, which corresponds well with this spam detection scenario. Adam optimization is selected to train the model which converges faster than Stochastic Gradient Descent. In addition, a dropout of 0.5 is used for regularization on the weights.

4 Analysis

4.1 Training Result

A total of 5 epochs is performed on the training set, with every epoch taking approximately 80 seconds on an Intel-i5 core. To be noticed, the first epoch has a relatively low training accuracy due to the design of model, which meets with our expectation. Since word embedding is initialized to be blank, the model will perform poorly on the very beginning of the training set. A certain amount of data is needed for the model to figure out an effective embedding that is helpful to classification.

As training proceeds, the embedding and filter's weights tend to stabilize, supported by the fact that model achieves approximately 97% accuracy on the first validation set, which is significantly higher than the accuracy on first training set and only slightly lower than the accuracy on test case. While training accuracy improves over epochs and eventually lends between 99%-100%, validation accuracy maintains around 98.9%, meaning that the model does not gain much performance in latter epochs.

Our model's performance on test set is similar to that on the latter validation sets, achieving 98.88% accuracy with small variation.

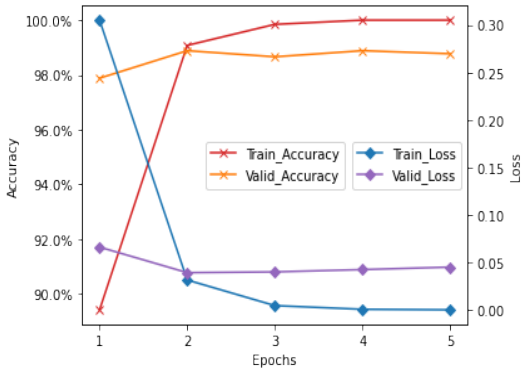

Figure 3: Training Results

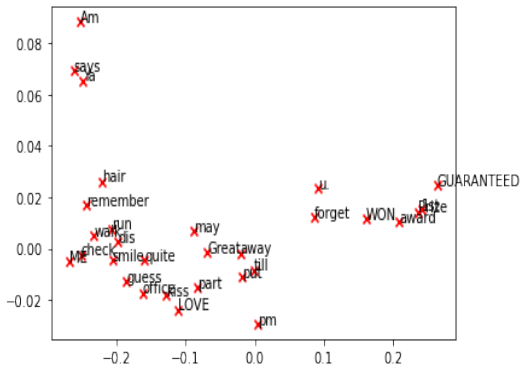

Figure 4: Embedding Plot

4.2 Embedding Result

We acquired a spam-specific word-embedding of 9448 tokens after training the entire model. The values are relatively small due to tiny gradient flowing back to word embedding in training, but this does not affect performance of the model.

By applying singular vector decomposition (SVD) to the word vectors and plotting it on a 2d space, we are able to discover some patterns of the word embedding. While exact meaning of x and y axis of this plot is not clear at this point, we state that small distance between words indicates certain similarities. For example, in Figure 4, suspicious words such as "WON," "award," "Prize," and "GUARANTEED" cluster on the right, supporting the assumption above.

4.3 Comparison

As shown in Table 1, our model outperforms Naïve Bayes and Logistic Regression models on the same dataset, achieving 98.88% accuracy. Although more computation is required, result show that neural network can be utilized to improve model performance in traditional machine learning tasks.

Model	Our Model	Naïve Bayes [5]	Logistic Regression [6]
Accuracy	98.88%	98.08%	96.04%

Table 1: Comparison Between Models

5 Conclusion

In conclusion, we introduced a fancy model to perform on spam classification problem which includes ideas of convolutional neural network and word embedding. While the model accomplished a higher accuracy compared with Naïve Bayes and Logistic Regression, it also generated a word embedding based on context of messages. In future study, we look forward to implementing this embedding in other similar tasks and exploring its generality.

References

[1] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages: 1532-1543. ACL, October 2014.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space *arXiv preprint arXiv:1301.3781*, 2013.

[3] Ben Trevett: pytorch-sentiment-analysis.
`https://github.com/bentrevett/pytorch-sentiment-analysis`

[4] UCI Machine Learning: SMS Spam Collection Dataset.
`https://www.kaggle.com/uciml/sms-spam-collection-dataset`

[5] Raj Tulluri: Spam-Filter-using-Naive-Bayes.
`https://github.com/rajtulluri/Spam-Filter-using-Naive-Bayes`

[6] Pratyush M: spam-message-classifier.
`https://github.com/pratyushmp/spam-message-classifier`