COMP42415 Coursework Report

1. Introduction

In this project, we focus on sentiment analysis on two different datasets, namely, Rotten Tomatoes movie reviews (RTMR) and Twitter posts (S140). We leverage several popular machine learning models for learning the relationship between text and underlying sentiment. Specifically, we build Naive Bayes models, Convolutional Neural Network model [2] and Long Short Term Memory (LSTM) model [3]. And finally, we evaluate the performance of our models using metrics like accuracy, precision, recall and F1-score [4].

2. Dataset

We work with two datasets in this project. The first one is Rotten Tomatoes movie review dataset in which each phrase is annotated with a sentiment value. The sentiment value of a phrase can be 0 - negative, 1 - somewhat negative, 2 – neutral, 3 - somewhat positive or 4 – positive. The second one is Sentiment140 which consists of tweets and corresponding sentiment values. The sentiment value of a tweet can be 0 - negative, 2 - neutral or 4 - positive. The criteria for sentiment labeling might be inconsistent between two datasets, which will affect the performance of models. However, in the context of this project, we will assume that labeling criteria are the same.

3. Data preprocessing

As we know, there are lots of noise and redundant information in the raw text. Therefore, data preprocessing [5] is a necessary step before building the models. In this project, we apply the same transformation to texts in both RTMR and S140 datasets. The details are as follows:

1. Tokenize the text
2. Remove all hash tags
3. Remove all stop words
4. Remove all URL
5. Transform letters to lower case
6. Remove extra spaces
7. Remove tokens that are neither alphabets nor numbers
8. Lemmatize the tokens

4. Text representation

Text are represented in different ways for Naive Bayes model and CNN/LSTM model. In Naive Bayes model, we use CountVectorizer in sklearn to convert collection of phrases/tweets to a matrix of token counts where each row's length is the same as the vocabulary size and each column's length is the same as the number of phrases/tweets. However, in CNN/LSTM model, the text is vectorized in another way. We use the texts_to_sequences() function in Keras to convert text to a sequence of integers where each integer stands for the index of the token in the corpus. In this approach, the length of a vector representing a piece of text depends on the number of tokens in the text so we need to use pad_sequences() so that vectors' lengths are the same.

5. Machine learning model

Three types of machine learning models are implemented for sentiment analysis and their architectures are described in details below.

5.1 Naive Bayes

Firstly, we leverage the multinomial Naive Bayes classifier in sklearn as sentiment analysis is essentially a classification task with discrete features. Naive Bayes classifier is a simple probabilistic classifier based on Bayes hypothesis. Given the arrangement of N factors, $X = \{X_1, X_2, X_3 ... X_n\}$, we need to develop the posterior for the occasion C_k among a set of possible results $C = \{C_1, C_2, C_3 ... C_k\}$. More concretely, we have the following rules:

$$P(C_k|X) = P(X|C_k)\, P(C_k)/P(X)$$

Where $P(C_k|X)$ is posterior, $P(X|C_k)$ is likelihood, $P(C)$ is prior and $P(X)$ is evidence. In the context of sentiment analysis, C_k is sentiment value like negative or positive, and X_n is the count for a particular token in a tweet.

5.2 CNN

Secondly, we build a convolution neural network model using Keras for predicting the sentiment of a given piece of text. The model consists of embedding layer, convolution layer, pooling layer, dropout layer, batch normalization layer and dense layer. And the scalar label is encoded using one-hot encoding so that it matches the size of the model output.

Next are the details of the CNN model architecture, some of which also apply to LSTM in the next subsection. The first layer, embedding layer, transforms input into dense vectors

with specified size. The second layer is convolution layer, which has a number of tunable hyper-parameters such as padding, strides and activation function. Both convolution layer and dense layer require an activation function, and we select “ReLU” and “softmax” for them, respectively. Dropout layer is considered as a form of regularization, which can effectively prevent the model from overfitting. Batch normalization layer is another form of regularization, which works by normalizing the inputs to each layer. There are lots of options for the optimizer and we finally pick Adam optimizer with learning rate being 0.001 after rounds of experiments. As this is a multi-class classification model, the loss function we use is “categorical_crossentropy”.

5.3 LSTM

Lastly, we also build a Long Short Term Memory (LSTM) model using Keras for predicting the sentiment of a given piece of text. The model consists of embedding layer, dropout layer, LSTM layer, and dense layer.

While some of the similar model details have been discussed above in the CNN section, LSTM has its own interesting design. LSTM is a special form of recurrent neural network (RNN), which is well known for its capability of learning long-term dependencies in the data. In the entire architecture, there are also embedding layer, dropout layer and dense layer, whose motivation and functionality has been described in the CNN section.

6. Experimental results

To comprehensively measure and compare the performance of different models on the data set, we evaluated them using several metrics including accuracy, precision, recall and F-score.

We have five sentiment labels in total, namely, 0 standing for negative, 1 standing for somewhat negative, 2 standing for neutral, 3 standing for somewhat positive and 4 standing for positive. And we have four separate datasets including RTMR training set, RTMR testing set, S140 training set and S140 testing set. It is worth mentioning that RTMR training and testing sets have all sentiment labels. However, S140 training set only has positive (4) and negative (0), and S140 testing set only has positive (4), neutral (2) and negative (0). The distribution of sentiment labels among these data sets could partially explain some of the results below.

6.1 Experiments with Naive Bayes

Experiment 1: The RTMR training set is used for training and the model is tested separately on RTMR testing set and S140 testing set. The accuracy on RTMR testing set is 0.569 and the accuracy on S140 testing set is 0.275. Figure 1 and figure 2 show the confusion matrices for experiment 1.

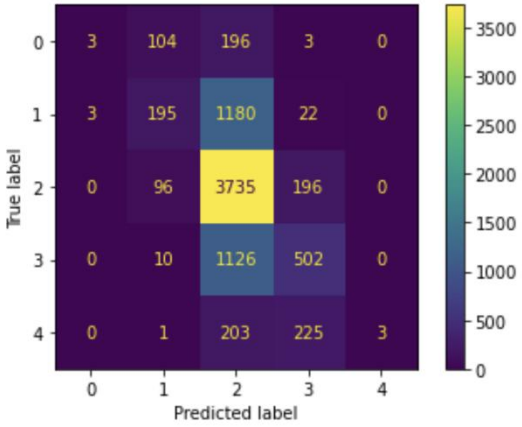

Figure 1: tested on RTMR

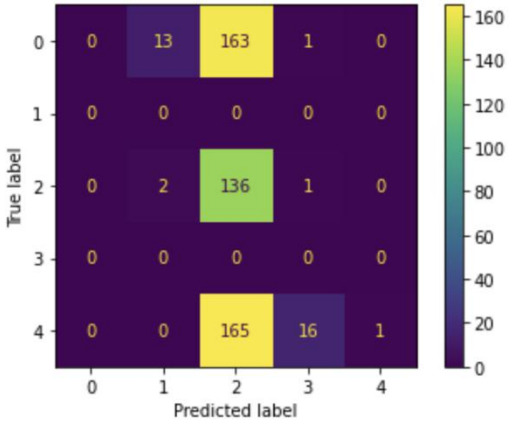

Figure 2: tested on S140

Experiment 2: The S140 training set is used for training and the model is tested separately on RTMR testing set and S140 testing set. The accuracy on RTMR testing set is 0.073 and the accuracy on S140 testing set is 0.582. Figure 3 and figure 4 show the confusion matrices for experiment 2.

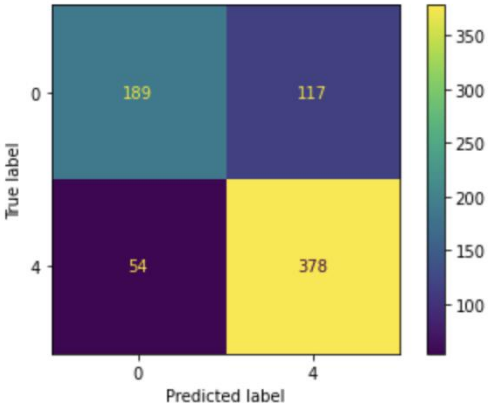

Figure 3: tested on RTMR

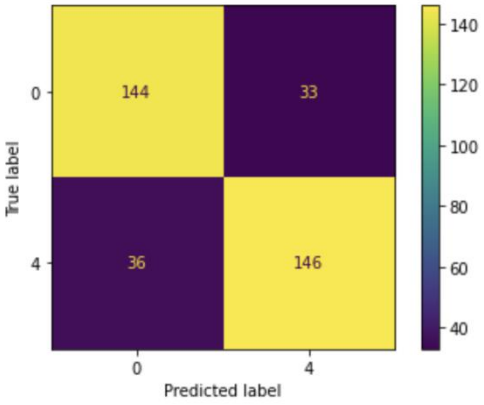

Figure 4: tested on S140

Experiment 3: The combination of two training sets is used for training and the model is tested separately on RTMR testing set and S140 testing set. The accuracy on RTMR testing set is 0.228 and the accuracy on S140 testing set is 0.578. Figure 5 and figure 6 show the confusion matrices for experiment 3.

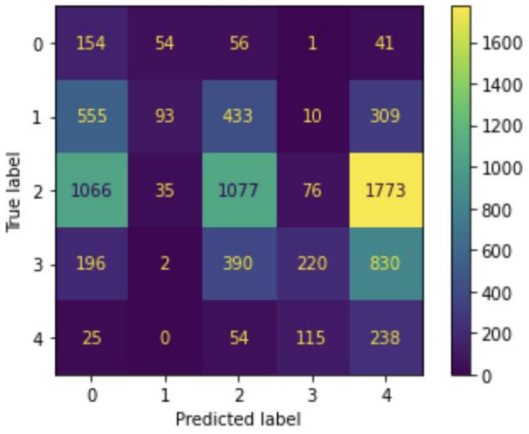

Figure 5: tested on RTMR

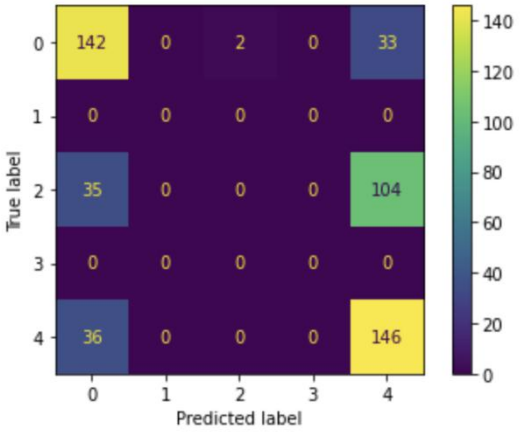

Figure 6: tested on S140

6.2 Experiments with CNN

The CNN model is trained on the training set of RTMR and tested separately on testing set of RTMR and S140. The accuracy on RTMR testing set is 0.579 and the accuracy on S140 testing set is 0.257. Figure 7 and figure 8 show the confusion matrices for experiment with CNN.

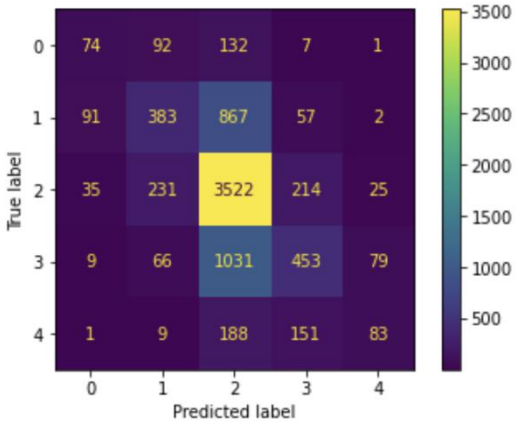

Figure 7: tested on RTMR

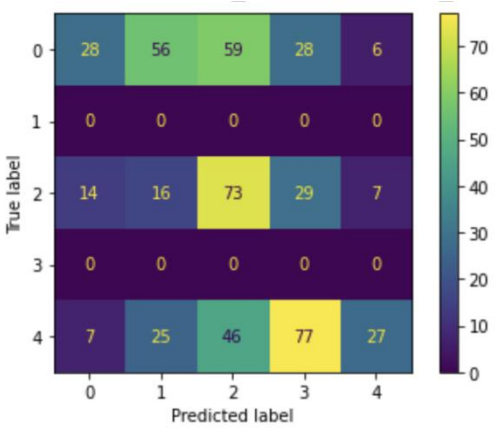

Figure 8: tested on S140

6.3 Experiments with LSTM

The LSTM model is trained on the training set of RTMR and tested separately on testing set of RTMR and S140. The accuracy on RTMR testing set is 0.526 and the accuracy on S140 testing set is 0.211. Figure 9 and figure 10 show the confusion matrices for experiment with LSTM.

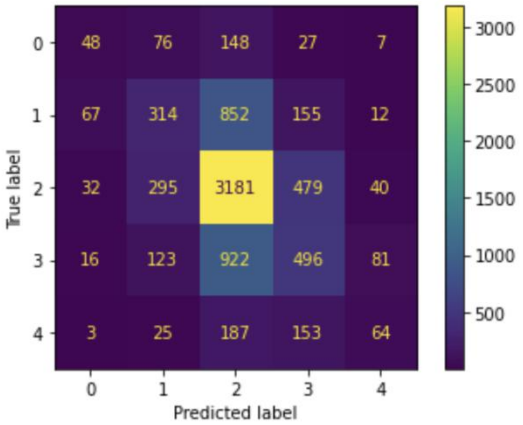

Figure 9: tested on RTMR

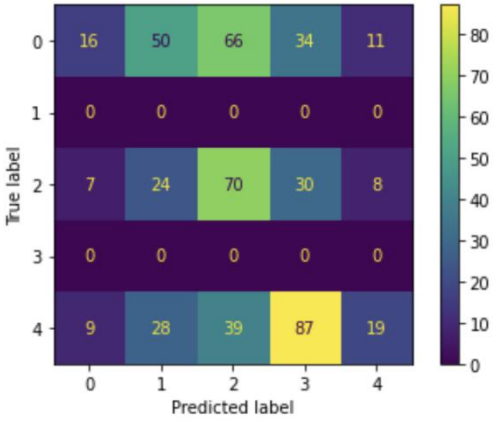

Figure 10: tested on S140

7. Discussion

One of the biggest limitation in this project is the lack of “somewhat negative” and “somewhat positive” samples in the S140 dataset. Therefore, when we train the model on S140 and test the model on RTMR, or inversely, the results would not be satisfying. One of the potential solution is transfer learning [6], but we will leave it for future study.

Also, there are some improvements we can make. For example, we can perform grid search on the hyper-parameter space of neural networks as there are a huge amount of combinations of hyper-parameters which might lead to significantly different model performance. Also, we can introduce ensemble learning, which means the prediction will be based on the outputs of several models including Naive Bayes, CNN and LSTM.

Reference

[1] Feldman, R., 2013. Techniques and applications for sentiment analysis. Communications of the ACM, 56(4), pp.82-89.

[2] O'Shea, K. and Nash, R., 2015. An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[3] Yu, Y., Si, X., Hu, C. and Zhang, J., 2019. A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 31(7), pp.1235-1270.

[4] Lever, J., 2016. Classification evaluation: It is important to understand both what a classification metric expresses and what it hides. Nature methods, 13(8), pp.603-605.

[5] Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J., Nithya, M., Kannan, S. and Gurusamy, V., 2014. Preprocessing techniques for text mining. International Journal of Computer Science & Communication Networks, 5(1), pp.7-16.

[6] Ruder, S., Peters, M.E., Swayamdipta, S. and Wolf, T., 2019, June. Transfer learning in natural language processing. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials (pp. 15-18).