Sarcasm and Irony Sentiment Detection

Yao Tang

May 2023

1 Introduction

According to my paper, Sarcasm Sentiment Analysis, I introduced two major approaches, machine learning and neural network, to conduct experiments. In order to get a hands-on experience with sentiment analysis, I used two neural network models, LSTM and Bert in the Python environment, to predict sarcasm sentiment. The datasets include train and test files extracted from Twitter and contained multi-labeled classifications: figurative, irony, sarcasm, and regular. The challenge I encountered during data training was the low accuracy rate on multi-labels and mixed categorization. In order to gain a higher f1 score, the experiment tried a merge method to integrate similar labels so in the Bert-transformer approach, the average accuracy reaches over 85 percent. The report gives details about each step of the experiment, data mining techniques, text analysis, and models' application.

2 Exploratory Data Analysis and Preprocessing

The notebook gives some exploratory analysis of both datasets. First, the experiment counts the values of data, using nltk and clean function to remove stopwords as well as punctuation marks for data preprocessing. Here is a question: are all sarcastic and ironic comments negative? In order to answer it, the experiment uses Vader sentiment to analyze test data sentiment, finding that both comments can be positive or negative. From the seaborn data visualization, the experiment shows that the proportion of each category and top frequently-expressed words: overall, sarcasm and irony are on the top of the list. For each category, there are 20 top words sorted from the train data. In contrast, the experiment ranks words based on Vader's sentimental classification. For the positive sentiment, 'peace', 'lol', 'funny',' like', 'good', is on the list, while words, such as 'news', 'drugs', 'late',' can't', are on the list. However, Vader's sentiment gives a disadvantageous categorizing result: the ironic and sarcastic expressions are usually phrased in a positive tone but conveyed a contradicted meaning, which is hard to directly be classified as 'positive' or 'negative'. Besides, the word clouds demonstrate big words in each category of the training data. An obvious problem is that in the category of 'figurative', data contains both sarcastic and ironic tags. Consider the neural network can accurately predict category so the data has not been subcategorized from figurative class. Further, the experiment explores emoji expression in datasets. In both training and test data, 'laugh cry', a contradictory expression, is the most used one. The experiment counts the citations of emojis on both datasets; to clean emojis for data training and testing, the remove-emoticon function has been written and processes deep cleaning. All the above are applied in LSTM and Bert models.

3 LSTM (Long Short Term Memory Model)

The experiment first explores LSTM model. Importing tensorflow.keras to tokenize cleaned text. In order to get an efficient neural network processing on categorization, the experiment uses a label-encoder to assign numeric labels for 4 classes: 0. figurative; 1. irony; 2. regular; 3. sarcasm. In the process of tokenization, the function gets sequential tokenization. Using train-test-split, the experiment tried 20 percent and 30 percent test sizes between train and test data. Next, the LSTM model requires embedding layers, a bidirectional layer, and 4 dense layers to pass through all data. The Adam optimizers calculate the loss function to get the accuracy of prediction. In reference to a similar data training model, the experiment applies keras. callback to hook into the various stages of the model training and inference. It is a trial test so the experiment only uses 2 epochs to train data, gaining 46 percent evaluation accuracy. In the prediction process, the LSTM model gets a 47 percent accuracy prediction.

4 Bert-Transformer Model

BERT creates an ideal prediction result. It is a better-performed model in data training. The experiment runs on the collab notebook so it cannot successfully run programming on a large-scale size with a large sample of data unless it could be run on GPU then the sample size and results can be more impressive. Nevertheless, it achieves 80 percent accuracy. As the EDA shows, the figurative class contains mixed categories, including sarcasm and irony. The experiment merges sarcasm and figurative class as they overlap similar words most, mapping class labels into 3. In the process of Tokenization, the experiment applies Bert's pretrained tokenizer. BERT is a model that requires a fixed-length text string, so the experiment sets a MAX LENGTH and calculates the text length of both datasets. The text processing includes token embedding, attention mask, and position embeddings, turning a large calculation capacity. That's why the experiment uses 5 percent of the training data in the CPU environment. text length and token length are all sorted on both training and testing data. Considering problems caused by imbalanced data, the experiment applies random over sampler to get the same size of the text in each class. The experiment builds a Bert model on Adam optimizer and categorical cross-entropy loss and

3 dense layers. The model only has 1 epoch, 32 batch sizes, and 10 percent validation data (less than LSTM model builds). However, from the notebook, it can be seen that the loss is 0.42 which means that Bert's model can get a relatively high accurate score. The training process lasts for an hour but the accuracy achieves 82 percent. In the prediction level, it gets a mirco average of 87 percent fl score.

5 Conclusion

From the experiment, I could have an in-depth understanding of neural network sentiment analysis. Just as the paper mentions, emojis and contradictory expressions, which are demonstrated in the Python notebook, are the main features of sarcasm and irony. The transformer model is a very effective sentiment detector in comparison to the LSTM model. But the latter could deal with a relatively large-scale dataset with multi labels in a shorter time, also being an ideal sentiment detector for binary classification or numeric historic data, for example, stock data.