CS 410: Project Progress Report

TEAM PYTHON

Team members:

- Chuanyue Shen (NetID: cs11, team leader)
- Jianjia Zhang (NetID: jianjia2)
- Runpeng Nie (NetID: runpeng3)

Tasks have been completed:

1. Data preprocessing:

Training data and test data store in JSON line file. Each data item has three fields. The first field of the training data is a label, indicating whether the response of this data is sarcasm or not. The first field of the test data is its ID. Both training data and test data have a response field and a context field. The response field stores the tweet to be classified, and the context field stores the conversation context of the response relatively.

Both response and context are string data. Because the data is tweets, so there are lots of emoji objects and @USER marks. The first step is removing the emojis and @USER. First of all, the regular expression package is used since all emojis are encoded in Unicode. By using *re.compile()* function the emoji Unicode pattern is defined, and emojis are removed through *re.sun()* function.

@USER is relatively easier to remove. Just like what we learned from lectures, there are lots of meaningless stop words like "the", "a" etc, they are almost useless for text classification. The *NLTK* package provides an English stopwords list, by adding "@USER" to the list and removing all words that appear in the list from response and context. Besides, for text classification, the punctuation character is useless too since the punctuation does not hold sentiment like normal words. Words are combined except punctuation through using *string.punctuation*. After these operations, the data is basically cleaned.

Only having a clean dataset is not enough. In English, with the change of grammar and context, there are many words with similar roots that have a similar meaning. For instance computer, computing, and computational. Their appearance increases the complexity of matrix operations. In order to improve performance and raise classification accuracy, words need to be grouped/replaced by their stem word. NLTK's *PortStemmer* tool is very useful. By replacing some words with their stem word, the computation complexity dropped significantly.

2. Model implementation

- Naive Bayes
  - For this model, we choose the *Gaussian distribution* to fit the distribution of the count of each word. Then use Naive Bayes classifier to fit the data.
  - Result:
    - Precision = 0.5371
    - Recall = 0.7967
    - F1 = 0.6416
- SVM
  - For this model, we preprocess the data using *TF-IDF* methods. Due to the reason that we have too many unique words and which will result in around 20,000 features, but the training data has only 5000 rows which will possibly result in underfitting. We remove the words with term count less than 3. After the data cleaning, the unique words are around 6,000, which is reasonable compared to original unique words.
  - Parameters:
    - min_df = 3
  - Result:
    - Precision = 0.5829
    - Recall = 0.8633
    - F1 = 0.6959
- **LSTM (Passed the baseline)**
  - For this model, we use three densely-connected layers and use *Sigmoid* as the activation function in the output layer. We choose *binary cross entropy* as the loss function, and *RMSprop* as the optimizer. We take the 5000 strings as the input and for each string we choose at most max_len = 100 words from the right side to left side.
  - Parameters: max_words = 5000, max_len = 100, batch_size = 128, epochs = 50, dropout_value = 0.5
  - Result
    - Precision = 0.6068
    - Recall = 0.8967
    - **F1 = 0.7238**
- CNN model (in progress):
  - We start with a basic CNN model based on PyTorch framework. The model uses

three convolutional layers with *LeakyReLU* and *BatchNorm2d*, and *Sigmoid* as the activation function in the output layer. We use SGD as the optimizer, and NLLLoss as the loss function. With proper fine tuning, the model is expected to pass the baseline and beat the LSTM model.

Tasks are pending:

- Finish the CNN model and fine-tune the learning parameters
- Implement some other state of art models/methods, such as transformer-based machine learning technique BERT

Challenges:

- During tuning LSTM models, we found it is not easy to improve the precision value.
- Current completed models cannot achieve much higher F1 scores. We will try to achieve the F1 scores by using CNN model and/or BERT model and finetuning the parameters.