CSC2511 Assignment 1 Bonus

Tianxiang Chen 999473181

This document records some ideas I implemented for the bonus part. Though the final accuracy didn't improve (still slightly above 50%), I will show the effort I made and the result I got.

1. Feature Analysis for empty post and removed post

When I did the pre-processing part of the assignment, I noticed there are some posts whose content is just an empty string or "[removed]".

For part3.3, picking the top K features, I also noticed most of the top 50 features (out of 179) are the ones from LIWC/Receptiviti.

For the first 29 features I wrote in featureExtraction part, the empty string or "[removed]" won't contributed any information, all the elements in the matrix would be simply zero. So, the question came that what about the values for the LIWC/Receptiviti features for these posts. If they are also zero, and the empty and removed post are evenly distributed across the four categories, then there is not much meaning to include them in the training set since we know they can't provide useful information to characterize the model.

Based on these observation, I did two experiments:

- Check the number of empty/removed post from the whole data given, list the distribution of these two posts among four categories.
- For the features provided from LIWC/Receptiviti for the empty/removed post, check what are the values of those features, e.g. if they are close to zero.

Code are written in **a1_bonus_post_analysis.py** and results for this part is recorded in **a1_bonus_post_analysis.csv**.

The total data size we given is 1,999,090 posts. Inside it, there are 126 empty posts and 69,507 removed posts.

The number of posts for each category is: Left: 598,944, Center 599,872, Right 600,002, Alt 200,272.

For the category distribution, in the order of [Left, Center, Right, Alt]

Empty post distribution: [21, 36, 54, 15]

Removed post distribution: [32631, 14895, 20234, 1747]

From the distribution, though the 'Alt' does not contain much empty/removed post, I will say the distribution is well-distributed among the rest three.

Next, I checked the LIWC/Receptiviti feature values for these empty/removed posts. As indicated in the **a1_bonus_post_analysis.csv**, the values are not zero. Indeed, some values are even not small. This surprises me a bit, since in my perspective of view, it is hard to extract information from such empty or removed posts. I also discussed with professor Rudzicz about this during the office hour. What he felt is either these features are build on some random Gaussian distribution if no valid input is given, or they use the deep knowledge like the post time, etc., to analysis the features values.

This indicates me that I can try to train the model without these empty/removed posts to see the result.

2. Extract feature data for non-empty and non-removed data

This part is similar to a combination of the part1 and 2 but only extract and preprocessing 10000 * 4 non-empty or non-removed post. The code is named **a1_extractFeatures.py** and the result is saved in **feats_selected.npz**.

3. Train the model using Grid Search in Sklearn

In this part, I tried to do a simple grid search to find the optimal parameter for the MLPClassifier and AdaBoostClassifier, which are the two best classifiers from the part3 result.

The code is in **a1_bonus_gridsearch.py**. I tried the find the optimal parameter both for the data with empty/removed and without them. The accuracy is still close for both classifier, in both cases. The including empty/removed case still is a bit better in accuracy, which gives about 51% for the AdaBoostClassifier with n_estimator = 100.

4. Train the model using Keras

Due to time constraint, I just did some basic attempts for this part. I just used the origin feature set which includes all the posts, this is because:

- It shows a better performance compared to the no-empty/no-removed data
- I do not have enough time to see the training result for both case since it does take a long time for training using Keras

The code is in a1_bonus_tf.py. I simply used a layer with 25 hidden unit, and softmax with output dimension as 4 for the last layer. The accuracy in this case is still slight above 50%.