Lab 2 Homework Report: Sentiment Analysis

Introduction and Data Overview

The goal for this project is to correctly predict and categorize tweets based on the dataset crawled from Twitter. The target variables are: “anger”, “anticipation”, “disgust”, “fear”, “sadness”, “surprise”, “trust”, and “joy”. We are given 3 file datasets: data_identification which consists of tweet_id and train or test identification, emotion which consists of tweet_id and category of emotion, and tweets_DM which consists of raw data obtained from twitter.

Data Preprocess and Cleaning

Before directly trying to predict the submission, I decided to preprocess the data first by merging all three of these datasets (data_identification, tweets_df, emotion) into one dataframe, this way we can observe and visualize the data clearly. Doing this, we obtain the merged data as shown below:

	tweet_id	identification	hashtags	text	emotion
0	0x28cc61	test	[]	@Habbo I've seen two separate colours of the e...	NaN
1	0x29e452	train	[]	Huge Respect👍 @JohnnyVegasReal talking about l...	joy
2	0x2b3819	train	[spateradio, app]	Yoooo we hit all our monthly goals with the ne...	joy
3	0x2db41f	test	[]	@FoxNews @KellyannePolls No serious self respe...	NaN
4	0x2a2acc	train	[]	@KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...	trust
...	...	...	...	...	...
1867530	0x227e25	train	[rip]	@BBCBreaking Such an inspirational talented pe...	disgust
1867531	0x293813	train	[libtards, Hillary, lost, sad, growup, Trump]	And still #libtards won't get off the guy's ba...	sadness
1867532	0x1e1a7e	train	[seeds, Joy, GLTChurch]	When you sow #seeds of service or hospitality ...	joy
1867533	0x2156a5	train	[]	@lorettalrose Will you be displaying some <LH>...	trust
1867534	0x2bb9d2	train	[]	Lord, I <LH> in you.	trust

1867535 rows × 5 columns

Table 1

Based on the dataset, we also found out that there seems to be a class imbalance where overall the target emotion “joy” is a lot more than other categories. In total, there was also 1.5 million rows of tweets with 400k of them being the submission testing data. Therefore, we have around 1.1 million data to be trained for our model.

The next step we should do is to split the data into training data (for the model training later on) and testing data (for submission prediction later on). Next, we will check for any potential missing values inside the training data, doing this we found out that there were no missing values.

Additionally, we will split the training data into testing data and validation data with a split ratio of 80:20, this way we can evaluate our model’s prediction capability later on.

Feature Engineering

For the feature engineering part, I tried two different approaches which was using Bag of Words (BOW) and Term Frequency Inverse Document Frequency (TFIDF). I decided to use this because it is a good baseline for our first model. The BOW and TFIDF that I chose are 500 and 1000 features respectively (will be used for Multinomial Naïve Bayes Classifier) with using nltk.word_tokenize as the tokenizer.

Data Mining Technique 1 (Multinomial Naïve Bayes using BOW and TFIDF)

We are trying to predict a target variable from a given features, this is a data mining classification problem. My first intuition for the technique to use for this classification is using the Multinomial Naïve Bayes Classifier to predict the sentiment of the tweets, because it gives us a good baseline for our model to be improved upon.

Evaluation for Technique 1

Using the 500 features Bag of Words feature engineering, our model for MNB acquired a result of 0.42 for both the testing and accuracy dataset, the classification report is shown below:

Naive Bayes Accuracy Testing: 0.42
Naive Bayes Accuracy Training: 0.42

	precision	recall	f1-score	support
anger	0.17	0.11	0.14	7973
anticipation	0.45	0.43	0.44	49787
disgust	0.29	0.32	0.30	27820
fear	0.18	0.13	0.15	12800
joy	0.49	0.63	0.55	103204
sadness	0.37	0.37	0.37	38687
surprise	0.41	0.11	0.18	9746
trust	0.35	0.21	0.26	41096
accuracy			0.42	291113
macro avg	0.34	0.29	0.30	291113
weighted avg	0.40	0.42	0.40	291113

	precision	recall	f1-score	support
anger	0.17	0.12	0.14	31894
anticipation	0.46	0.43	0.44	199148
disgust	0.29	0.32	0.30	111281
fear	0.18	0.13	0.15	51199
joy	0.49	0.63	0.55	412813
sadness	0.37	0.37	0.37	154750
surprise	0.45	0.12	0.19	38983
trust	0.35	0.21	0.26	164382
accuracy			0.42	1164450
macro avg	0.34	0.29	0.30	1164450
weighted avg	0.41	0.42	0.40	1164450

Table 2

Next, using the TFIDF feature engineering with 1000 features, our model for MNB acquired a result of 0.46 for both the validation and training dataset, the classification report is shown below:

Naive Bayes Accuracy Testing: 0.46
Naive Bayes Accuracy Training: 0.46

	precision	recall	f1-score	support
anger	0.89	0.04	0.07	7973
anticipation	0.60	0.33	0.43	49787
disgust	0.53	0.14	0.22	27820
fear	0.88	0.16	0.27	12800
joy	0.42	0.92	0.58	103204
sadness	0.50	0.30	0.37	38687
surprise	0.85	0.07	0.13	9746
trust	0.73	0.07	0.12	41096
accuracy			0.46	291113
macro avg	0.68	0.25	0.27	291113
weighted avg	0.56	0.46	0.39	291113

	precision	recall	f1-score	support
anger	0.88	0.04	0.08	31894
anticipation	0.60	0.34	0.43	199148
disgust	0.55	0.14	0.23	111281
fear	0.89	0.16	0.27	51199
joy	0.42	0.93	0.58	412813
sadness	0.51	0.30	0.38	154750
surprise	0.88	0.08	0.14	38983
trust	0.74	0.07	0.12	164382
accuracy			0.46	1164450
macro avg	0.68	0.26	0.28	1164450
weighted avg	0.57	0.46	0.39	1164450

Table 3

Based on these two classification report, we can say that the overall the TFIDF vectorizer performs better than the BOW vectorizer. This is because BOW only captures word counts presence and absence, cannot capture information about context, word order, and semantic meaning, while the TFIDF improves over BOW by weighting words based on the importance across documents. However, after submitting it I was only able to get a score of 0.38.

Additionally, Multinomial Naïve Bayes assumes that the features we have, which are words are conditionally independent given the label, this assumption often does not hold in real world text because words are related through grammar and meaning. This assumption may result to underfitting due to oversimplification. Added with the imbalance data for the category data “joy” that we have seen, it affects the quality of the model even more.

Data Mining Technique 2 (Deep Neural Network using TFIDF)

Based on previous feature engineering that we did, TFIDF performs better. Therefore, I decided to use TFIDF instead of BOW for this model also. Deep Neural Network needs a lot of training data for it to work better, this is a good fit for our dataset because it consists of over 1 million data which will allow our neural network to not overfit because of low data.

Additionally, after previous tries in number of features for TFIDF, I found out that using 10k features yields best result compared to other smaller or bigger values and therefore I decided to use this number of feature. I also used one-hot encoding to deal with the categorical label (y) because we cannot directly use emotion into the model. We encode all the possible emotions which are anger, anticipation, disgust, fear, sadness, surprise, trust, and joy.

Overall, the architecture of the DNN model consists of 10000 input shape wth 2 hidden layers consisting of 64 neurons each with the output layer of 8 which represents all the possible output of emotion. The file shown below is the DNN model architecture:

```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 10000)]           0

 dense_12 (Dense)            (None, 64)                640064

 re_lu_8 (ReLU)              (None, 64)                0

 dense_13 (Dense)            (None, 64)                4160

 re_lu_9 (ReLU)              (None, 64)                0

 dense_14 (Dense)            (None, 8)                 520

 softmax_4 (Softmax)         (None, 8)                 0

=================================================================
Total params: 644744 (2.46 MB)
Trainable params: 644744 (2.46 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Table 4

Evaluation 2

At first, I tried to use training and validation dataset to check for the model's accuracy in predicting the emotions then submitting it to the Kaggle submission, this way I obtained a 0.453 f1 score. Therefore, I decided to not split the dataset into training and validation but directly train all of them and then allow the model to predict the submission file itself. This has allowed to get better f1 score of 0.458.

Conclusion

Based on the few models, that I did the Deep Neural Network model is better because it is able to capture patterns in the data more compared to MNB. This is because DNN can learn more complex and nonlinear relationships between the features while also adapting to data flexibility even with sparse and high dimensional input from TFIDF.