楊添貴

110006211

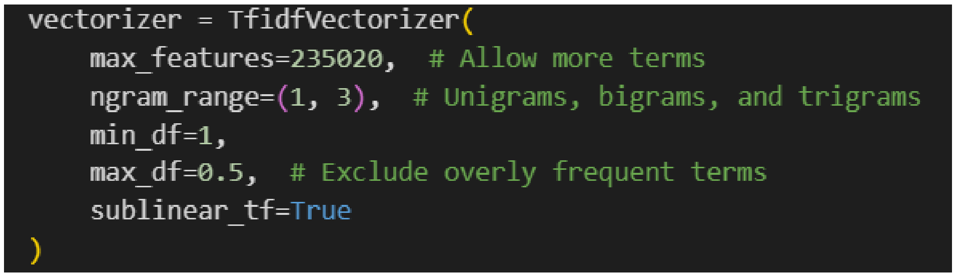
Lab 2 Homework Report

The competition involves predicting the emotion expressed in tweets based on labeled data. The emotions include anger, anticipation, disgust, fear, sadness, surprise, trust, and joy. The dataset contains tweets with their corresponding emotional labels and is divided into training and test sets. The primary tasks are data preprocessing, feature engineering, and model development.

For my preprocessing steps, I noticed that on the raw dataset, there are some unnecessary words like <LH> and other stop words that can be removed to improve my model performance, so I did the following steps:

- Lowercasing: Converted all text to lowercase to normalize the data.
- Removing Hashtags, Mentions, and URLs: These elements were often irrelevant to the tweet's emotion.
- Removing Special Characters and Digits: Simplified text by retaining only alphabets.
- Replacing <LH> Tags: Eliminated placeholders like <LH> which were artifacts in the data.
- Stopword Removal: Removed common words like "the," "is," which don't add meaning to the emotions.
- Lemmatization: Reduced words to their base form to handle inflections (e.g., "running" → "run").

For my Feature Engineering part, I used TF-IDF Vectorization to convert the text into numerical features for modeling. The tricky part would be testing each of the parameters in TF-IDF vectorizer that yield the best accuracy on the competition.



```
vectorizer = TfidfVectorizer(
    max_features=235020,  # Allow more terms
    ngram_range=(1, 3),  # Unigrams, bigrams, and trigrams
    min_df=1,
    max_df=0.5,  # Exclude overly frequent terms
    sublinear_tf=True
)
```

So there are 4 main parameters in the TF-IDF vectorizer that I mainly changed:

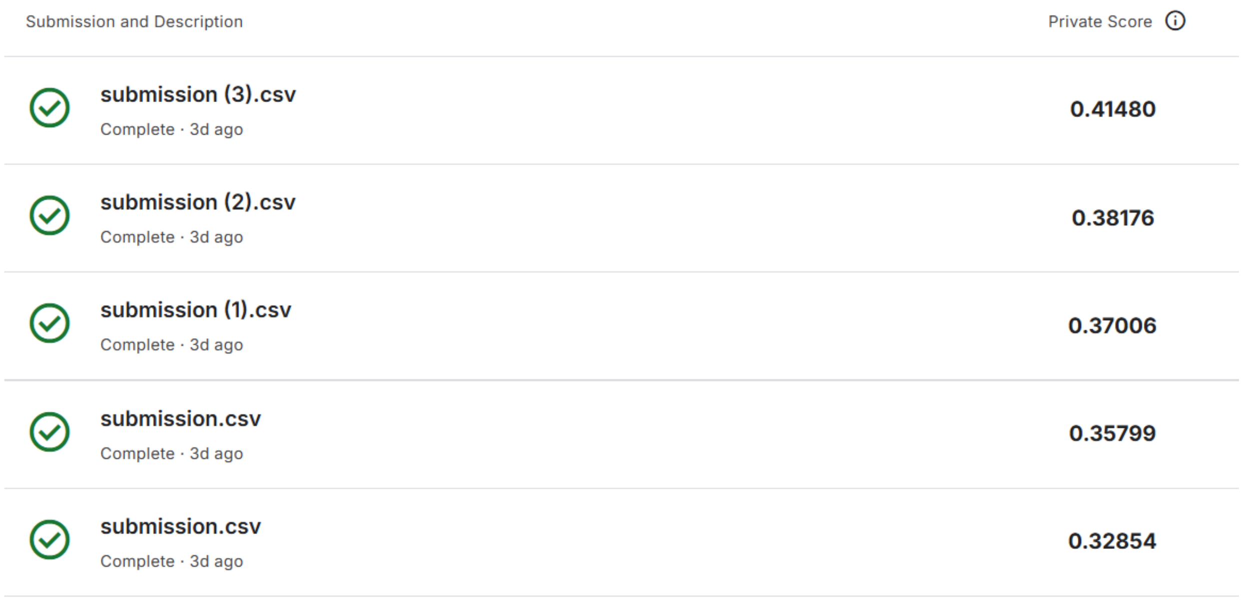
max_features=235020: Specifies the maximum number of unique terms (features) to include in the TF-IDF representation.

ngram_range=(1, 3): Specifies the range of n-grams to include in the feature set. Included unigrams, bigrams and trigrams for better context.

min_df=1 : Includes terms that appear in at least 1 document.

max_df=0.5: Excludes terms that appear in more than 50% of the documents.

I submitted 5 submissions for the competition and here are the results:



Submission and Description	Private Score
✓ submission (3).csv Complete · 3d ago	0.41480
✓ submission (2).csv Complete · 3d ago	0.38176
✓ submission (1).csv Complete · 3d ago	0.37006
✓ submission.csv Complete · 3d ago	0.35799
✓ submission.csv Complete · 3d ago	0.32854

On my first ever submission, I did not fine tune any parameter of the vectorizer at all and I ended up receiving the lowest score.

For my second till the third submission, I tested 4 different value of max features, and min-max document frequency to find the best score. On each attempt, my score just keeps going up.

For my 4th submission, I ended up with my final parameter which I have shown the snippet of the values above.

For my last submission, I still used the same parameter as my 4th submission. I noticed that I have tried every possible parameter but the testing/validation accuracy does not increases. So on my last attempt, I tried to not preprocess the text which ended up giving me the highest accuracy amongst all 5 attempts.

For my model of choice, I used Multinomial Naive Bayes (NB) for its simplicity and effectiveness in text classification tasks. I Split the training data into 80% training and 20% validation using stratified sampling to maintain class distribution. I also Adjusted alpha (Laplace smoothing) to 0.5 after experimentation (adjusting the alpha does not really effect the final score)

On the validation set, I evaluated using metrics like accuracy, precision, recall, and F1-score. So I tested a bunch of parameters of vectorizer on this sample dataset first, if the accuracy increases, then I will submit it to the competition, else I know that the parameters does not increase the final accuracy. I noticed that the NB model performed well on the high-frequency features but struggled with rare terms.

Conclusion

The chosen approach of TF-IDF with Multinomial Naive Bayes was simple and fast but the thing with it is that it does not yield the best final score, I realized a few hours later that the first place person on the competition is definitely using an LLM Pretrained Model, but when I tried it, I realized that it took a really long time to predict all 300,000 tweet's emotions. I was thinking of using either ChatGPT 4.0 or Gemini API to do predict the emotions. But either way, I think from this lab, I gained a lot of knowledge about sentimental analysis and the tips and tricks to achieve a better score, like not preprocessing text, parameter fine tuning and usage of LLMs.