Sec B Team 8

Best accuracy: 0.82720

I. Data Cleaning

At the beginning, we ran some data cleaning processes, to remove punctuations and other outlier data like symbols or pictographs.

Then, we split each prediction into four categories and updated the get_error_type() function, which returns the four types of error.

Aslo, we modified the classify() function, added some regular expressions and the postagging function, tried random number of multiplier of len(negative_words_inqtabs) and sum(negative_words_swn.values(), and found the numbers that generate the best score with the original model, which is 0.682.

II. VADER

The first package we tried was VADER, which generated an accuracy score of 0.677.

We also ran several examples to see the different types of error:

A negative example:

```
text = '''i hired this movie out from my local movie shop, not really expecting anything to flash or fancy
classify(text, inqtabs_dict, swn_dict)
```

```
0
```

Two false positive examples:

```
text = '''
this movie isn't that great...at all but it's good when you want to just laugh, because it's pretty ridiculous :)
classify(text, inqtabs_dict, swn_dict)
```

```
1
```

```
text = '''
Frank Sinatra starred in this odd little short from RKO that is now in the public domain. The film came out at about the same time the war ended an
classify(text, inqtabs_dict, swn_dict)
```

```
1
```

III. TextBlob

The second package we ran was TextBlob. With the same regular expressions, instead of running postagging and random multipliers, we ran the model without postagging and if the

sntmnt.sentiment.polarity > 0, the score would be 1. With this package, the accuracy score is 0.700. However, the two false positive examples are still predicted as positive:

```
text = '''
this movie isn't that great...at all but it's good when you want to just laugh, because it
classify(text, inqtabs_dict, swn_dict)
```

```
1
```

```
text = '''
Frank Sinatra starred in this odd little short from RKO that is now in the public domain.
classify(text, inqtabs_dict, swn_dict)
```

```
1
```

IV. Flair

Third, we ran Flair, which is a pre-trained embedding based model. We can determine using flair which words are used in similar contexts as the words which have similar vector representations are used in similar contexts. Flair is much slower than Vader or TextBlob. However, its performance is much better than Vader or TextBlob. With Flair, we got an accuracy score of 0.827.
Also, for the false positive examples we listed above, we were able to successfully predicted them as negative using Flair:

```
text = '''
this movie isn't that great...at all but it's good when you want to just laugh, because it's pre
classify(text, inqtabs_dict, swn_dict)
```

```
0
```

```
text = '''
Frank Sinatra starred in this odd little short from RKO that is now in the public domain. The fi
classify(text, inqtabs_dict, swn_dict)
```

```
0
```

```
text = '''i hired this movie out from my local movie shop, not really expecting anything to flas
classify(text, inqtabs_dict, swn_dict)
```

```
0
```

From our research and analysis, we found rule based models struggle when determining when a negative sentence is truly negative, which is perhaps because of the high usage of emphasis, jargon and sarcasm in negative sentences. However, flair is able to escape this trap and that's why it has the highest accuracy score.
Later, if we have more time, we probably will build a transformer-based model and apply transfer learning, which is a powerful method that has been dominating NLP task leaderboards lately.