Amazon Reviews

Sentiment Analysis

Chaya Fradel Abramovitz

Business Problem

- How positive or negative are reviews?
- We generally assume that higher ratings are more positive and lower ratings are more negative. Is that the case?
- Can we *somewhat* predict the rating of a review based on the positive/negative score the models will give for each rating?

Approach

- I will run 2 different models that will give a positive/negative score for each review
- I will compare both models and see which one gives a more accurate score
- I will see if any of these models are reliable

Data

- Dataset = Amazon Reviews (SD Cards)
- Around 5,000 rows

	A	B	
1	Id	Score	Text
2	0	4	No issues.
3	1	5	Purchased this for my device, it worked as advertised. You
4	2	4	it works as expected. I should have sprung for the higher ca
5	3	5	This think has worked out great.Had a diff. bran 64gb card
6	4	5	Bought it with Retail Packaging, arrived legit, in a orange en
7	5	5	It's mini storage. It doesn't do anything else and it's not su
8	6	5	I have it in my phone and it never skips a beat. File transfer
9	7	5	It's hard to believe how affordable digital has become. 32
10	8	5	Works in a HTC Rezound. Was running short of space on a

How Many Reviews per Rating?

- Most reviews are 5 star

VADER Model

- Older and simpler
- Faster

```
from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm

sia = SentimentIntensityAnalyzer()

res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    text = row['Text']
    myid = row['Id']
    res[myid] = sia.polarity_scores(str(text))

vaders = pd.DataFrame(res).T
vaders = vaders.reset_index().rename(columns={'index': 'Id'})
vaders = vaders.merge(df, how='left')
```

	Id	neg	neu	pos	compound	Score	Text
0	0	0.688	0.312	0.000	-0.2960	4	No issues.
1	1	0.075	0.925	0.000	-0.2960	5	Purchased this for my device, it worked as adv...
2	2	0.077	0.923	0.000	-0.3089	4	it works as expected. I should have sprung for...
3	3	0.000	0.909	0.091	0.7081	5	This think has worked out great.Had a diff. br...
4	4	0.038	0.835	0.127	0.7087	5	Bought it with Retail Packaging, arrived legit...

ROBERTA Pretrained Model

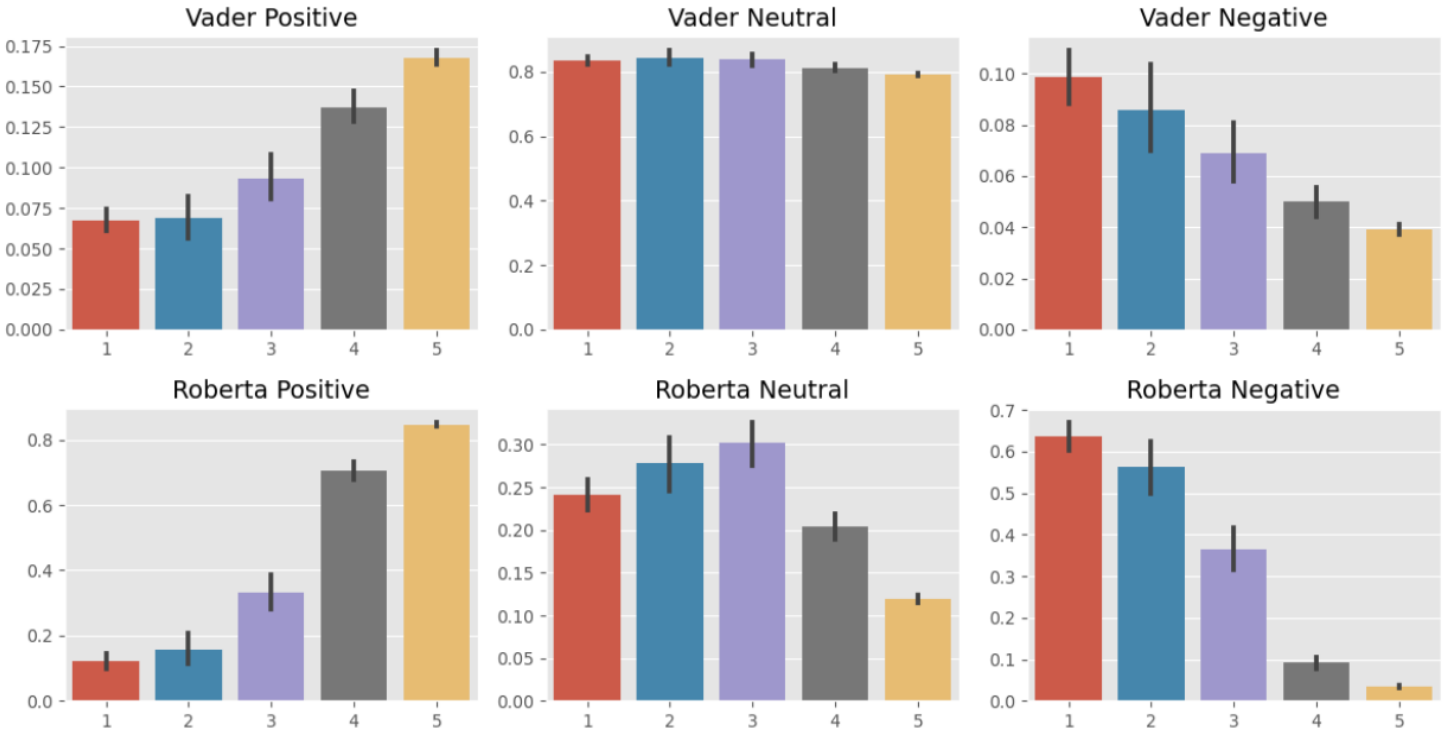
- Newer and more complex
- Slower

	Id	vader_neg	vader_neu	vader_pos	vader_compound	roberta_neg	roberta_neu	roberta_pos	Score	Text
0	0	0.688	0.312	0.000	-0.2960	0.074959	0.589215	0.335826	4	No issues.
1	1	0.075	0.925	0.000	-0.2960	0.007265	0.095529	0.897206	5	Purchased this for my device, it worked as adv...
2	2	0.077	0.923	0.000	-0.3089	0.423757	0.448311	0.127933	4	it works as expected. I should have sprung for...
3	3	0.000	0.909	0.091	0.7081	0.002447	0.019229	0.978324	5	This think has worked out great.Had a diff. br...
4	4	0.038	0.835	0.127	0.7087	0.003552	0.067459	0.928989	5	Bought it with Retail Packaging, arrived legit...

```
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax

MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)

# Run for Roberta Model
encoded_text = tokenizer(example, return_tensors='pt')
output = model(**encoded_text)
scores = output[0][0].detach().numpy()
scores = softmax(scores)
scores_dict = {
    'roberta_neg' : scores[0],
    'roberta_neu' : scores[1],
    'roberta_pos' : scores[2]
}
print(scores_dict)

def polarity_scores_roberta(example):
    encoded_text = tokenizer(example, return_tensors='pt')
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    scores_dict = {
        'roberta_neg' : scores[0],
        'roberta_neu' : scores[1],
        'roberta_pos' : scores[2]
    }
    return scores_dict

res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    try:
        text = str(row['Text'])
        myid = row['Id']
        vader_result = sia.polarity_scores(text)
        vader_result_rename = {}
        for key, value in vader_result.items():
            vader_result_rename[f"vader_{key}"] = value
        roberta_result = polarity_scores_roberta(text)
        both = {**vader_result_rename, **roberta_result}
        res[myid] = both
    except RuntimeError:
        print(f'Broke for id {myid}')

results_df = pd.DataFrame(res).T
results_df = results_df.reset_index().rename(columns={'index': 'Id'})
results_df = results_df.merge(df, how='left')
```

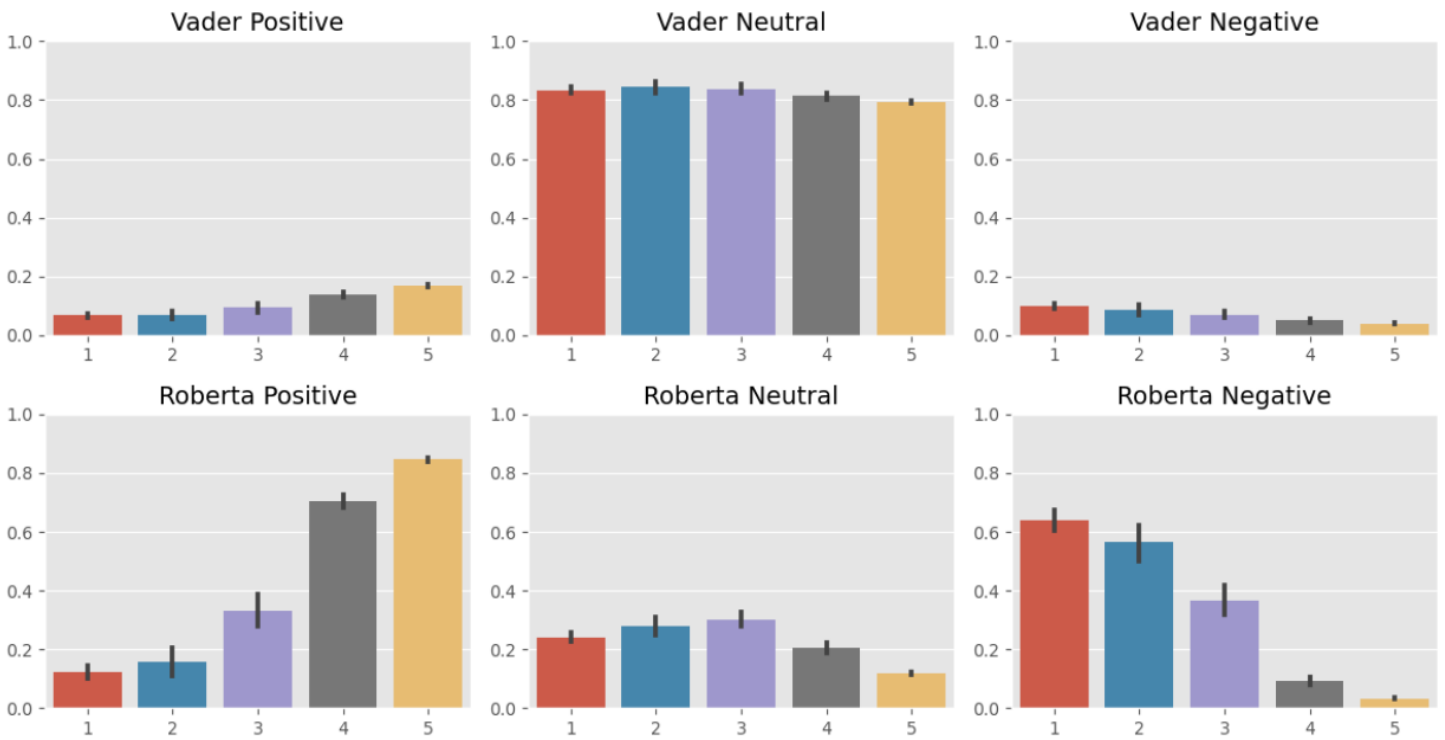
At First Glance

- Both models perform pretty well.
- ROBERTA model seems slightly better than VADER



After Some Adjustment (Y Axis)

- VADER gives a pretty similar score across all ratings
- ROBERTA seems much more accurate than VADER



Run Time

- VADER = 2 seconds
- ROBERTA = 6 hours, 37 minutes, 25 seconds

```
Broke for id 76
Broke for id 123
Broke for id 722
Broke for id 1380
Broke for id 2031
Broke for id 2799
Broke for id 2881
Broke for id 2934
Broke for id 2993
Broke for id 3345
Broke for id 3449
Broke for id 3757
Broke for id 3967
Broke for id 4176
Broke for id 4212
Broke for id 4423
Broke for id 4587
Broke for id 4596
```

Problem with ROBERTA

Breaks for some reviews

Score Prediction

	Score	vader_pred	roberta_pred
0	5	5.0	5.0
1	1	4.0	2.0
2	5	4.0	5.0
3	5	5.0	5.0
4	5	4.0	5.0
5	1	4.0	3.0

- Used Sci-Kit Learn Linear Regression to predict score
- ROBERTA has a lower RMSE and higher accuracy than VADER
  - VADER

```
Root mean squared error:  0.9968172842423195
Accuracy score:  0.1269177433926829
```

  - ROBERTA

```
Root mean squared error:  0.7137812053183863
Accuracy score:  0.5523338473708039
```

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Load data
data = results_df

# Prepare data
X = data[['vader_neg', 'vader_neu', 'vader_pos', 'vader_compound']].values
y = data['Score'].values

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=23)

# Create and fit the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on test data
y_pred_v = model.predict(X_test)

# Round predicted values to nearest integer
y_pred_v_rounded = np.round(y_pred_v)

# Evaluate model performance using root mean squared error (RMSE)
rmse = np.sqrt(mean_squared_error(y_test, y_pred_v))
print('Root mean squared error: ', rmse)

# Calculate accuracy score for the predictions
accuracy = model.score(X_test, y_test)
```

Thoughts

- 55% Accuracy Score may be quite accurate since not all reviews are rated correctly
- Next step might be to see if the model can be used to detect outliers – positive reviews that got a negative score/negative reviews that got a high score

Conclusion

- ROBERTA model seems to perform better than VADER
- ROBERTA comes at a higher run time cost
- ROBERTA won’t run for all reviews
- Look into possibility of using model for outlier detection