Sentiment Analysis on Movie Reviews

DTSC 710 Machine Learning Team Project

Yanting Wu
Gahyeon Back
Jabili Sandadi
May 12th

Datasets

Rotten Tomatoes movies and critic reviews dataset

1-10 1130017 rows × 8 columns pd.Dataframe

	rotten_tomatoes_link	critic_name	top_critic	publisher_name	review_type	review_score	review_date	review_content
0	m/0814255	Andrew L. Urban	False	Urban Cinefile	Fresh	NaN	2010-02-06	A fantasy adventure that fuses Greek m…
1	m/0814255	Louise Keller	False	Urban Cinefile	Fresh	NaN	2010-02-06	Uma Thurman as Medusa, the gorgon with…
2	m/0814255	NaN	False	FILMINK (Australia)	Fresh	NaN	2010-02-09	With a top-notch cast and dazzling spe…
3	m/0814255	Ben McEachen	False	Sunday Mail (Australia)	Fresh	3.5/5	2010-02-09	Whether audiences will get behind The …
4	m/0814255	Ethan Alter	True	Hollywood Reporter	Rotten	NaN	2010-02-10	What's really lacking in The Lightning…
5	m/0814255	David Germain	True	Associated Press	Rotten	NaN	2010-02-10	It's more a list of ingredients than a…
6	m/0814255	Nick Schager	False	Slant Magazine	Rotten	1/4	2010-02-10	Harry Potter knockoffs don't come more…
7	m/0814255	Bill Goodykoontz	True	Arizona Republic	Fresh	3.5/5	2010-02-10	Percy Jackson isn't a great movie, but…
8	m/0814255	Jordan Hoffman	False	UGO	Fresh	B	2010-02-10	Fun, brisk and imaginative
9	m/0814255	Jim Schembri	True	The Age (Australia)	Fresh	3/5	2010-02-10	Crammed with dragons, set-destroying f…

Datasets

IMDb Dataset

1-10 50000 rows × 2 columns pd.Dataframe

	review	sentiment
0	One of the other reviewers has mentione…	positive
1	A wonderful little production. <b…	positive
2	I thought this was a wonderful way to s…	positive
3	Basically there's a family where a litt…	negative
4	Petter Mattei's "Love in the Time of Mo…	positive
5	Probably my all-time favorite movie, a …	positive
6	I sure would like to see a resurrection…	positive
7	This show was an amazing, fresh & innov…	negative
8	Encouraged by the positive comments abo…	negative
9	If you like original gut wrenching laug…	positive

Data Preprocessing

```
df = pd.read_csv('rotten_tomatoes_critic_reviews.csv')
print(df.shape)
df['review_type'] = df['review_type'].replace({'Rotten': '0', 'Fresh': '1'})
df['Id'] = df.reset_index().index + 1
df = df.head(1000)
print(df.shape)
```

```
(1130017, 8)
(1000, 9)
```

```
df.head()
```

	rotten_tomatoes_link	critic_name	top_critic	publisher_name	review_type	review_score	review_date	review_content	Id
0	m/0814255	Andrew L. Urban	False	Urban Cinefile	1	NaN	2010-02-06	A fantasy adventure that fuses Greek mythology...	1
1	m/0814255	Louise Keller	False	Urban Cinefile	1	NaN	2010-02-06	Uma Thurman as Medusa, the gorgon with a coiff...	2
2	m/0814255	NaN	False	FILMINK (Australia)	1	NaN	2010-02-09	With a top-notch cast and dazzling special eff...	3
3	m/0814255	Ben McEachen	False	Sunday Mail (Australia)	1	3.5/5	2010-02-09	Whether audiences will get behind The Lightnin...	4
4	m/0814255	Ethan Alter	True	Hollywood Reporter	0	NaN	2010-02-10	What's really lacking in The Lightning Thief i...	5

Basic NLTK

```
tokens = nltk.word_tokenize(example)
tokens[:10]
```

```
['My', 'problems', 'with', 'it', 'are', 'the', 'same', 'as', 'with', 'most']
```

```
▶ 0.1s
tagged = nltk.pos_tag(tokens)
tagged[:10]
```

```
[('My', 'PRP$'),
 ('problems', 'NNS'),
 ('with', 'IN'),
 ('it', 'PRP'),
 ('are', 'VBP'),
 ('the', 'DT'),
 ('same', 'JJ'),
 ('as', 'IN'),
 ('with', 'IN'),
 ('most', 'JJS')]
```

Basic NLTK

```
entities = nltk.chunk.ne_chunk(tagged)
entities.pprint()
```

```
  they/PRP
  forget/VBP
  which/WDT
  parts/NNS
  of/IN
  the/DT
  story/NN
  people/NNS
  want/VBP
  to/TO
  see/VB
  .../:
  they/PRP
  figure/VBP
  that/IN
  people/NNS
  (/(
  kids/NNS
  )/))
```

VADER Sentiment Scoring

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

sia = SentimentIntensityAnalyzer()
```

```
sia.polarity_scores(example)
```

```
{'neg': 0.156, 'neu': 0.773, 'pos': 0.071, 'compound': -0.6326}
```

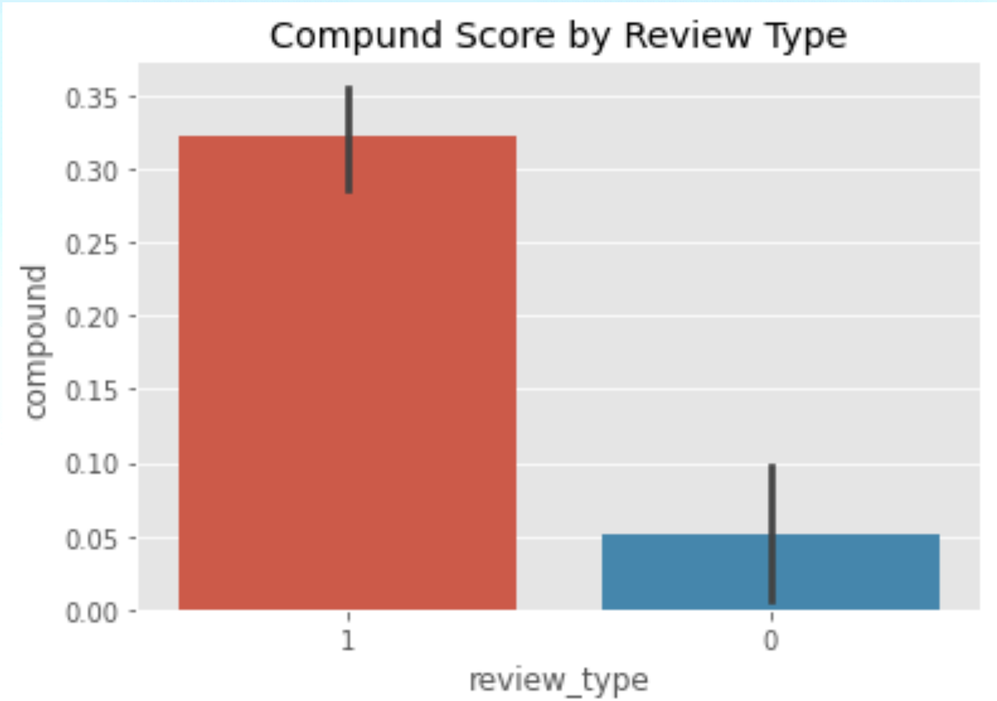
```
df['review_content'] = df['review_content'].astype(str)

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

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

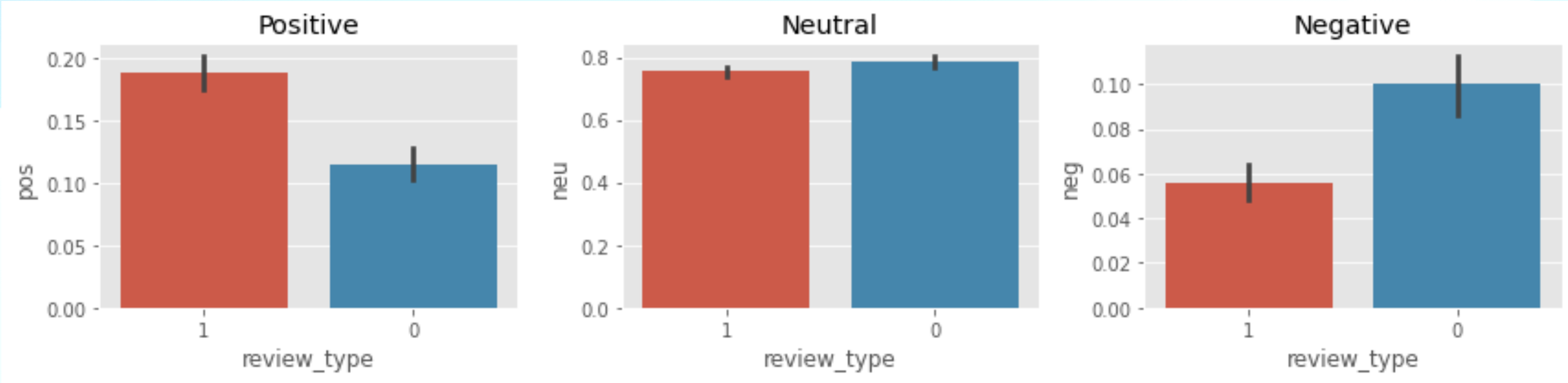
VADER Sentiment Scoring

	Id	neg	neu	pos	compound	rotten_tomatoes_link	critic_name	top_critic	publisher_name	review_type	review_score	review_date	review_content
0	1	0.000	0.762	0.238	0.7579	m/0814255	Andrew L. Urban	False	Urban Cinefile	1	NaN	2010-02-06	A fantasy adventure that fuses Greek mythology...
1	2	0.000	1.000	0.000	0.0000	m/0814255	Louise Keller	False	Urban Cinefile	1	NaN	2010-02-06	Uma Thurman as Medusa, the gorgon with a coiff...
2	3	0.000	0.870	0.130	0.4019	m/0814255	NaN	False	FILMINK (Australia)	1	NaN	2010-02-09	With a top-notch cast and dazzling special eff...
3	4	0.080	0.727	0.193	0.7050	m/0814255	Ben McEachen	False	Sunday Mail (Australia)	1	3.5/5	2010-02-09	Whether audiences will get behind The Lightnin...
4	5	0.124	0.876	0.000	-0.5267	m/0814255	Ethan Alter	True	Hollywood Reporter	0	NaN	2010-02-10	What's really lacking in The Lightning Thief i...

Plot VADER results



Plot VADER results



Sentiment Analysis with RoBERTa

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

```
0.1s
print(example)
sia.polarity_scores(example)
```

```
My problems with it are the same as with most screen adaptations of young adult novels...they spend so much time struggling to expla:

{'neg': 0.156, 'neu': 0.773, 'pos': 0.071, 'compound': -0.6326}
```

```
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)
```

```
{'roberta_neg': 0.8077242, 'roberta_neu': 0.17579375, 'roberta_pos': 0.016482059}
```

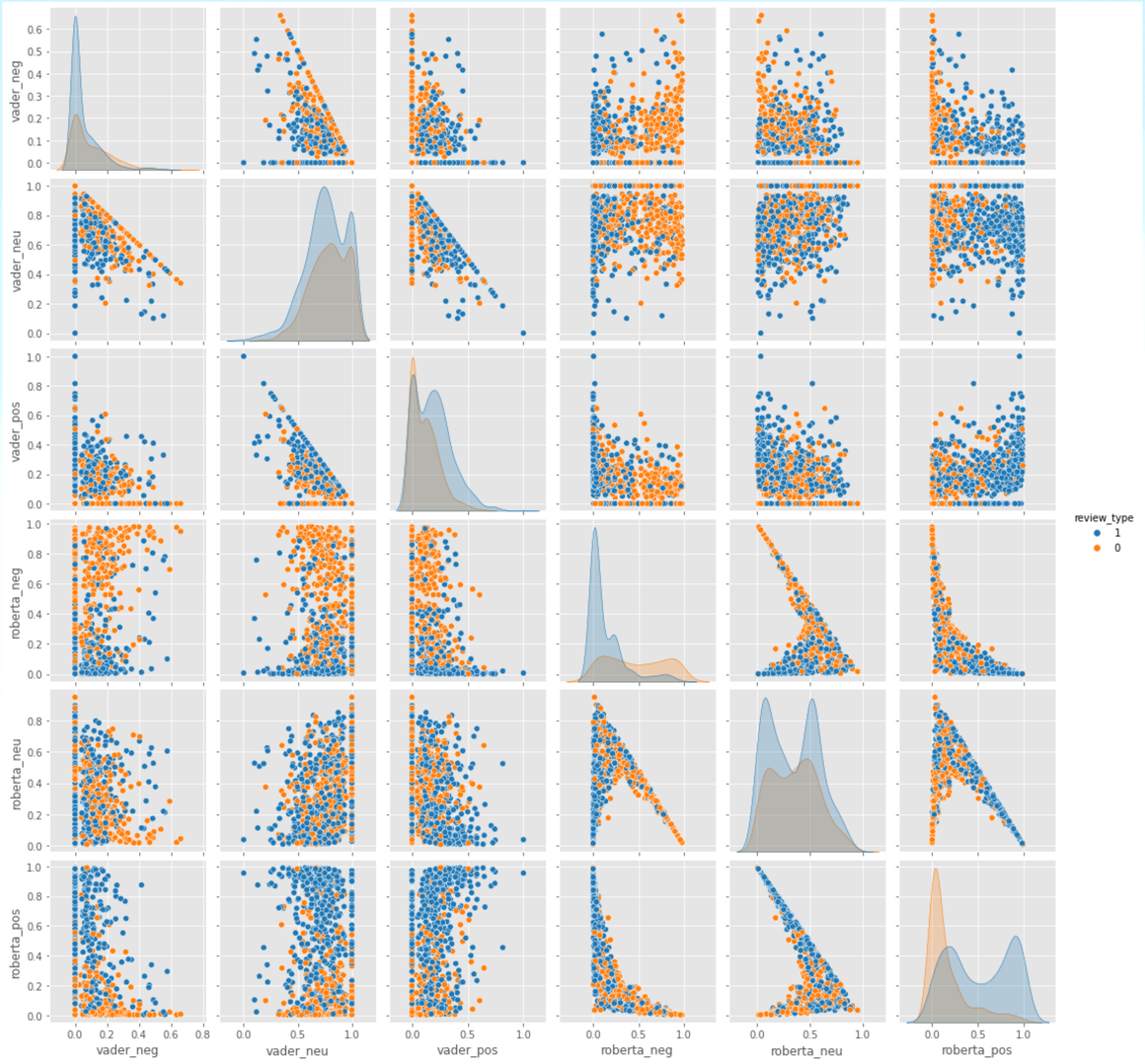
Sentiment Analysis with RoBERTa

```python
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
```

```python
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    try:
        text = row['review_content']
        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}')
```

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

Combine and compare



Prediction and Accuracy

```
new_df = pd.read_csv('IMDB Dataset.csv')
```

```
new_df['review'] = new_df['review'].astype(str)
```

```
new_df['sentiment'] = new_df['sentiment'].replace({'positive': '1', 'negative': '0'})
```

```
new_df['Id'] = new_df.reset_index().index + 1
new_df = new_df.head(100)
new_df
```

{...}

```
def get_sentiment_scores(text):
    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}
    return both
```

▶ 2m 09s

```
res = {}
for i, row in tqdm(new_df.iterrows(), total=len(new_df)):
    try:
        text = row['review']
        myid = row['Id']
        sentiment_scores = get_sentiment_scores(text)
        res[myid] = sentiment_scores
    except RuntimeError:
        print(f'Broke for id {myid}')
```

{...}

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

```
results_new_df['predicted_sentiment'] = np.where((results_new_df['roberta_pos'] > 0.5) & (results_new_df['vader_pos'] > results_new_
results_new_df['correct_prediction'] = np.where(results_new_df['predicted_sentiment'] == results_new_df['sentiment'], 1, 0)
accuracy = results_new_df['correct_prediction'].mean()
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 87.36%
```

Conclusion

Questions and Discussion