COMP-4448

Christopher Kramer

2021-02-18

Sentiment Analysis with NB

In [1]:
```
import pandas as pd
import numpy as np
from itertools import repeat
from typing import Union
import seaborn as sns
from collections import Counter, defaultdict
```

4) Find some text data of your own choice, it could be labelled tweets, etc. Your dataset should have at least 200 instances, and if there are several columns of text, you can choose to merge the text columns into a single text column. Each text instance should have at least 60 words. Clean the data, split the data, transform the data to a representation suitable for your algorithm, build your model and evaluate the model. Tune some parameters of interest and write a short report about what problem your mini project is trying to address, the description of your data, the choice of algorithm used, the performance of your algorithm, overfitting, the choice of hyperparameters tunned, then your recommendation or conclusion (imagine you were trying to recommend this algorithm to a stakeholder, and you need this report to include important and persuasive elements). Your report could be in one or two paragraphs and should include relevant code and output at the end.

The sentiment of tweets concerning the global pandemic could be a useful tool in understanding the public reaction to government rollout of vaccines, etc.

Is it possible to reliably predict the sentiment of a Corona Virus tweet on a 5-point scale using a Multinomial Naive Bayes classifier?

Load

In [97]:
```
data = pd.concat([pd.read_csv('Corona_NLP_train.csv'), pd.read_csv('Corona_NLP_test.csv')], ignore_index=True)
```

In [98]:
```
data.sample(5)
```

Out[98]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
15812	19611	64563	NaN	22-03-2020	I am loving these COVID-19 gas prices! It's b...	Positive
43951	2795	47747	Correct Time Zone	14-03-2020	BEING NEIGHBORLY IN A TIME OF COVID-19 ? \r\n...	Positive
37952	41751	86703	Portland, OR	11-04-2020	The Consumer in the Age of Corona Virus - how ...	Neutral
35910	39709	84661	Cambridge, MA	09-04-2020	Dairy farmers:: reserves are tapped out after y...	Negative
39494	43293	88245	NaN	13-04-2020	@WAGSocialCare @WAGSocialCare your Walgreens w...	Extremely Positive

In [99]:
```
data['Sentiment'].unique()
```

Out[99]:
```
array(['Neutral', 'Positive', 'Extremely Negative', 'Negative',
       'Extremely Positive'], dtype=object)
```

Preprocess

In [100]:
```
data.dtypes
```

Out[100]:
```
UserName          int64
ScreenName        int64
Location         object
TweetAt          object
OriginalTweet    object
Sentiment        object
dtype: object
```

The tweet text will be the primary feature used in analysis.

Time data will also be captured as a feature to include in the model. Individuals may be more or less positive depending on the time of day, and days into the pandemic.

In [101]:
```
data['TweetAt'] = pd.to_datetime(data['TweetAt'])
```

In [102]:
```
data['daysago'] = pd.Timestamp.now() - data['TweetAt']
```

In [103]:
```
data
```

Out[103]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment	daysago
0	3799	48751	London	2020-03-16	@MeNytbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral	343 days 17:06:40.468433
1	3800	48752	UK	2020-03-16	advice Talk to your neighbours family to excha...	Positive	343 days 17:06:40.468433
2	3801	48753	Vagabonds	2020-03-16	Coronavirus Australia: Woolworths to give elde...	Positive	343 days 17:06:40.468433
3	3802	48754	NaN	2020-03-16	My food stock is not the only one which is emp...	Positive	343 days 17:06:40.468433
4	3803	48755	NaN	2020-03-16	Me, ready to go at supermarket during the #COV...	Extremely Negative	343 days 17:06:40.468433
...	...	...	...	...	...	...	...
44950	3794	48746	Israel ??	2020-03-16	Meanwhile In A Supermarket in Israel -- People:...	Positive	343 days 17:06:40.468433
44951	3795	48747	Farmington, NM	2020-03-16	Did you panic buy a lot of non-perishable item...	Negative	343 days 17:06:40.468433
44952	3796	48748	Haverford, PA	2020-03-16	Asst Prof of Economics @cconces was on @NBCPhi...	Neutral	343 days 17:06:40.468433
44953	3797	48749	NaN	2020-03-16	Gov need to do somethings instead of biar je r...	Extremely Negative	343 days 17:06:40.468433
44954	3798	48750	Arlington, Virginia	2020-03-16	I and @ForestandPaper members are committed to...	Extremely Positive	343 days 17:06:40.468433

44955 rows × 7 columns

Unfortunately, the use of time of day/days from today as features is not feasible. The tweets do not contain HH:MM data and are sourced entirely from a single day, so time features will be dropped

In [104]:
```
data = data[['OriginalTweet', 'Sentiment']]
```

In [105]:
```
data['OriginalTweet'] = data['OriginalTweet'].apply(lambda x: clean_string(x, 5))
```

```
<ipython-input-105-3ed8e8d8b28>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['OriginalTweet'] = data['OriginalTweet'].apply(lambda x: clean_string(x, 5))
```

In [106]:
```
data = data[~data['OriginalTweet'].isna()]
```

Split

In [107]:
```
X = data['OriginalTweet']
y = data['Sentiment']
```

Feature engineering

In [108]:
```
tfidf = TfidfVectorizer(strip_accents = 'ascii', lowercase=False)
```

In [109]:
```
X = tfidf.fit_transform(X)
```

In [110]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

Model Training

In [111]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [112]:
```
multi_clf = MultinomialNB()
```

In [113]:
```
multi_clf.fit(X_train, y_train)
```

Out[113]:
```
MultinomialNB()
```

Initial scoring

In [114]:
```
accuracy_score(y_train, multi_clf.predict(X_train))
```

Out[114]:
```
0.5321528148543632
```

In [115]:
```
accuracy_score(y_test, multi_clf.predict(X_test))
```

Out[115]:
```
0.3677700658401564
```

The model is overfit and does not generalize well.

Hyperparameter Tuning

In [116]:
```
# trying tune-sklearn which is supposed to be faster than built-in tuners.
from tune_sklearn import TuneSearchCV
```

In [117]:
```
# This dataset takes a long time to train, so I've limited my choices and number of folds
params = {
    'alpha': list(np.logspace(0,9,num=50, endpoint=True)),
    'fit_prior': [True, False]
}
```

In [118]:
```
# Below can be replaced with "GridSearchCV" for classic tuning
grid = TuneSearchCV(multi_clf, params, n_jobs=10, verbose=1, cv=5, use_gpu = True)
```

In [119]:
```
grid.fit(X_train, y_train)
```

```
== Status ==
Memory usage on this node: 17.5/31.9 GiB
Using FIFO scheduling algorithm.
Resources requested: 0/16 CPUs, 0.0/1 GPUs, 0.0/9.33 GiB heap, 0.0/3.17 GB objects
Result logdir: C:\Users\KittheKat\ray_results\_Trainable_2021-02-22_17-06-58
Number of trials: 10/10 (10 TERMINATED)
```

```
(pid=43648) Windows fatal exception: access violation
(pid=43648)
```

Out[119]:
```
TuneSearchCV(cv=5, estimator=MultinomialNB(),
             loggers=[<class 'ray.tune.logger.JsonLogger'>,
                      <class 'ray.tune.logger.CSVLogger'>],
             n_jobs=10,
             param_distributions={'alpha': [1.0, 1.5264179671752336,
                                            2.329951810515372,
                                            3.5564803062231296,
                                            5.428675439323861,
                                            8.286427728546844,
                                            12.648552168552964,
                                            19.306977288832506,
                                            29.470517025518113,
                                            44.984326683969446,
                                            68.66488450043...
                                            2023.5896477251576,
                                            3088.8435964774817,
                                            4714.866363457394,
                                            7196.856730011521,
                                            10985.411419875594,
                                            16768.3293681101,
                                            25595.479226995383,
                                            39069.39937054621,
                                            59636.23316594649,
                                            91029.81779915227,
                                            138949.5494373139,
                                            212095.08879201926, ...],
                                 'fit_prior': [True, False]},
             scoring={'score': <function _passthrough_scorer at 0x00000201282F44C0>},
             sk_n_jobs=1, use_gpu=True, verbose=1)
```

In [120]:
```
grid.best_params_
```

Out[120]:
```
{'alpha': 1.5264179671752336, 'fit_prior': False}
```

In [121]:
```
grid.best_score_
```

Out[121]:
```
0.3865457362795516 3
```

In [122]:
```
best_multi = grid.best_estimator_
```

Post-tuning scores

In [123]:
```
accuracy_score(y_train, best_multi.predict(X_train))
```

Out[123]:
```
0.5950940262205612
```

In [124]:
```
accuracy_score(y_test, best_multi.predict(X_test))
```

Out[124]:
```
0.4003381384588005
```

While hyperparameter tuning did boost performance, it did not seem to significantly improve overfitting, a common issue in NLP.

I suspect this problem may require a more complex model to solve. That, or more nuanced text preprocessing (Tweets are tricky to work with).

I would recommend utilizing a different model family for this usecase, potentially DNN with LSTM.

It may be possible to layer binomial classifiers to improve performance:

```
        Is sentiment positive?
            /     \
           /       \
          /         \
```

Yes - how positive? No - how negative?

```
        /\          /\
        1 2         3  4
```

In [ ]:
```

```