Sentiment Analysis on Tweets

Cameron Kline
Edison Pan

Introduction to Sentiment Analysis

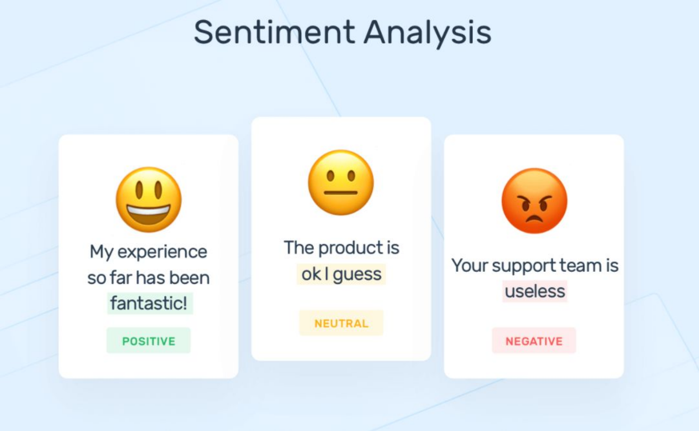
What is NLP sentiment analysis?:

The process of algorithmically detecting emotion

- Easy for humans
- Hard for machines

Applications in

- Business
- Marketing
- Customer service


Sentiment Analysis
My experience so far has been fantastic!
POSITIVE
The product is ok I guess
NEUTRAL
Your support team is useless
NEGATIVE

https://vitalflux.com/sentiment-analysis-machine-learning-techniques/

Dataset

~74k tweets from around 2021 about video games and tech

Positive:

“I don’t see how this looks like as Xbox controller but y’all will say anything. Anyway this is fire.”

“I love @Rainbow6Game so much 💙”

Negative:

“@verizon Can you waive some data overage charges? Been tough for folks out here.”

“@ Borderlands, how can I file a complaint? Your CEO doesn't pay his employees their bonuses.”

Datapoints comes with tweet’s label as well as its “category”. Category comes from content of a tweet and it’s context

4 Labels:

- Positive
- Negative
- Neutral
- Irrelevant

Irrelevant class given to tweets that contain text not related to its category.

Dataset limitations:

Poorly labeled classes:

For example:

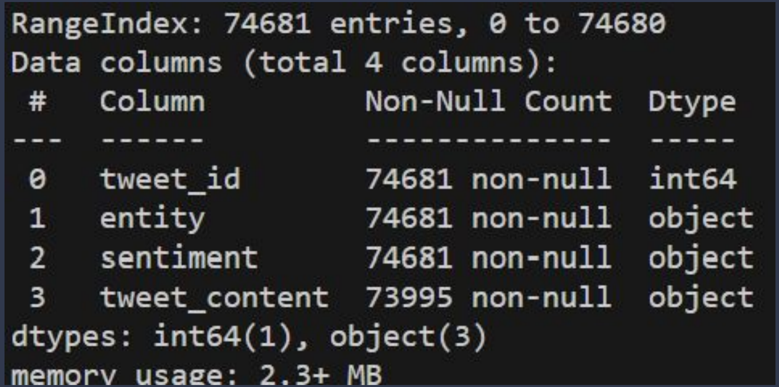
“im getting on borderlands and i will kill you all,” and “was”

being labeled as “Positive” and

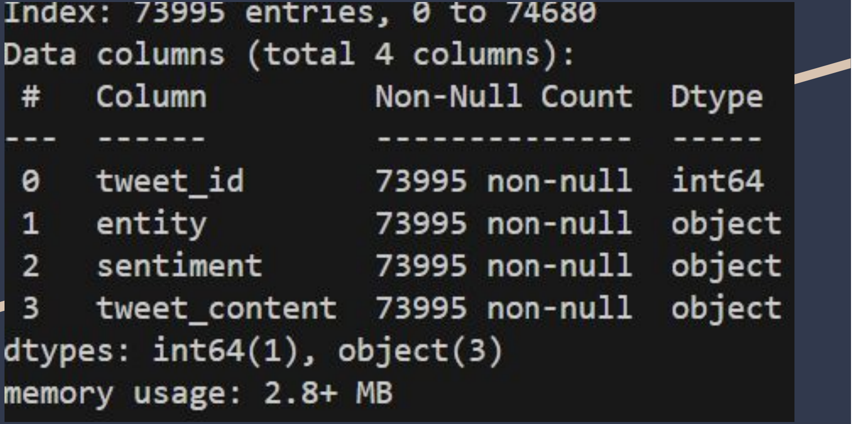
“Already loving Bleeding Edge... what a clever game!”

Being labeled as “Neutral”

Preprocessing Data



```
RangeIndex: 74681 entries, 0 to 74680
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   tweet_id       74681 non-null  int64 
 1   entity         74681 non-null  object
 2   sentiment      74681 non-null  object
 3   tweet_content  73995 non-null  object
dtypes: int64(1), object(3)
memory usage: 2.3+ MB
```



```
Index: 73995 entries, 0 to 74680
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   tweet_id       73995 non-null  int64 
 1   entity         73995 non-null  object
 2   sentiment      73995 non-null  object
 3   tweet_content  73995 non-null  object
dtypes: int64(1), object(3)
memory usage: 2.8+ MB
```

Dataset Problems:

- Missing entries
- Models cannot directly process tweet content as raw text

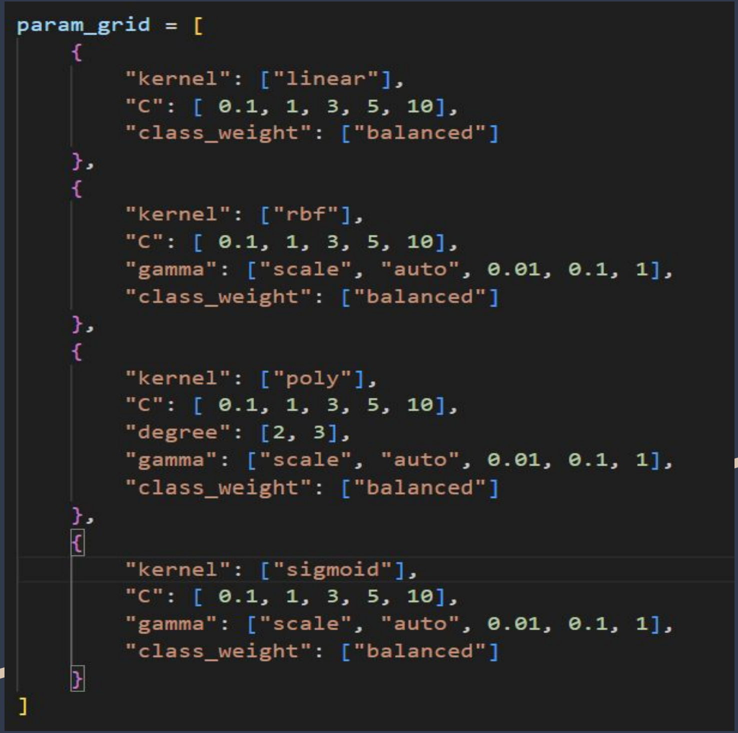
Solution:

- Drop samples with missing entries
- Vectorize the tweet content

TF-IDF

- Used to evaluate how important a word is to a sample in relation to a larger collection of samples
- Returns a matrix
- Each row = a sample(a tweet)
- Each column = a word
- Each cell(i, j) in matrix contains the score for the j-th term in the i-th sample

Methods – SVM



```
param_grid = [
    {
        "kernel": ["linear"],
        "C": [ 0.1, 1, 3, 5, 10],
        "class_weight": ["balanced"]
    },
    {
        "kernel": ["rbf"],
        "C": [ 0.1, 1, 3, 5, 10],
        "gamma": ["scale", "auto", 0.01, 0.1, 1],
        "class_weight": ["balanced"]
    },
    {
        "kernel": ["poly"],
        "C": [ 0.1, 1, 3, 5, 10],
        "degree": [2, 3],
        "gamma": ["scale", "auto", 0.01, 0.1, 1],
        "class_weight": ["balanced"]
    },
    {
        "kernel": ["sigmoid"],
        "C": [ 0.1, 1, 3, 5, 10],
        "gamma": ["scale", "auto", 0.01, 0.1, 1],
        "class_weight": ["balanced"]
    }
]
```

Different SVM kernels may perform differently depending on the dataset.

We need to find the best combination of kernel type and hyperparameters.

Solution: Grid Search

Kernels

- Linear
- RBF
- Poly
- Sigmoid

Hyperparameters:

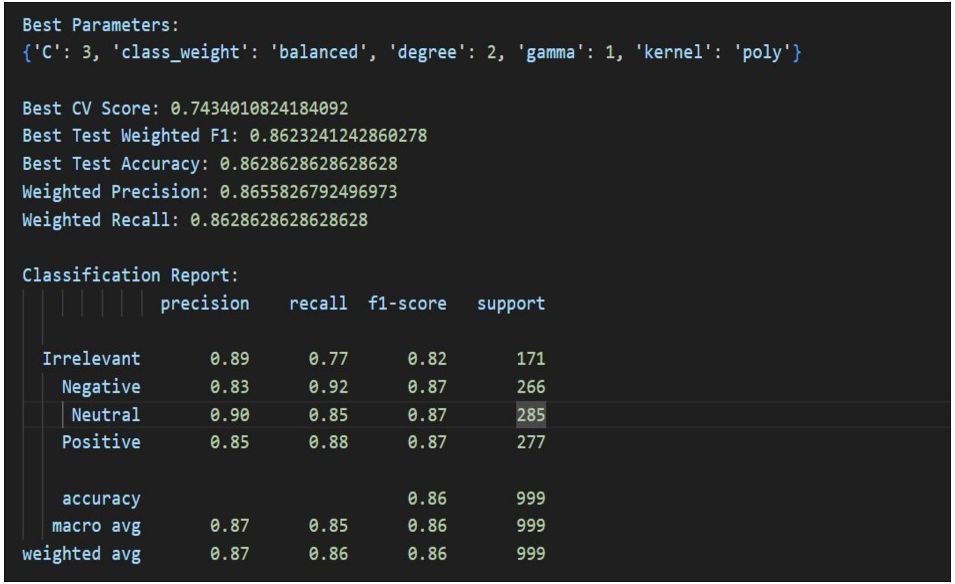
- C
- Gamma (For RBF, Poly, Sigmoid)
- Degree (for Poly)
- Class Weight

Results – SVM

HOW BEST MODEL WAS CHOSEN

Weighted F1 score: 0.787

Accounts for class imbalance by giving proportional importance to each class..



```
Best Parameters:
{'C': 3, 'class_weight': 'balanced', 'degree': 2, 'gamma': 1, 'kernel': 'poly'}

Best CV Score: 0.7434010824184092
Best Test Weighted F1: 0.8623241242860278
Best Test Accuracy: 0.8628628628628628
Weighted Precision: 0.8655826792496973
Weighted Recall: 0.8628628628628628

Classification Report:
              precision    recall  f1-score   support

  Irrelevant       0.89      0.77      0.82       171
    Negative       0.83      0.92      0.87       266
     Neutral       0.90      0.85      0.87       285
    Positive       0.85      0.88      0.87       277

    accuracy                           0.86       999
   macro avg       0.87      0.85      0.86       999
weighted avg       0.87      0.86      0.86       999
```

Not the absolute global best model due to limitations

Methods – BERT

BERT (Bidirectional Encoder Representations from Transformers) is an encoder only model trained on unlabeled text (Wikipedia and BookCorpus).

Embedders:

- “all-Mini-LM-L12-v2”
- “bert-base-uncased”
- “all-mpnet-base-v2”

Logistic Regression vs Random Forest

Grid Search to find optimal hyperparameters:

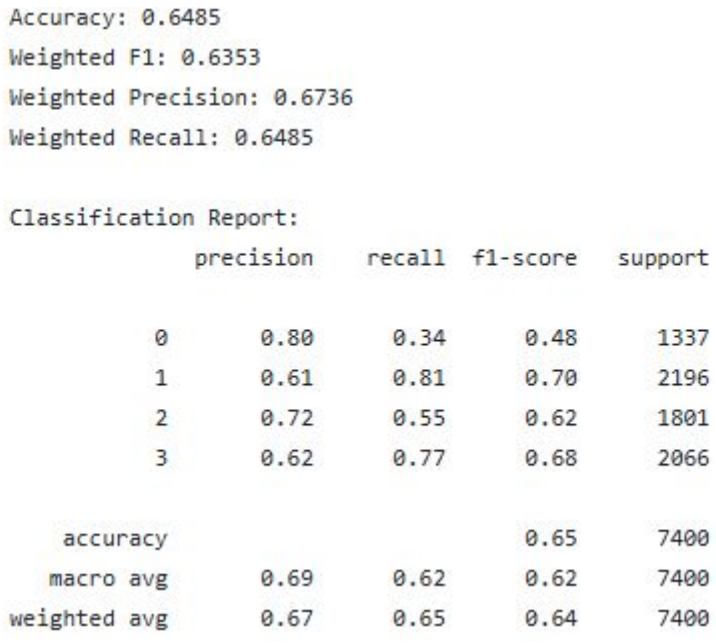
- N estimators
- Max depth
- Min samples split
- Min samples leaf
- Max features

Results – BERT

Best results came from Random Forest and all-MiniLM-L12-v2

Best parameters: {'clf__max_depth': None, 'clf__max_features': 'sqrt', 'clf__min_samples_leaf': 2, 'clf__min_samples_split': 2, 'clf__n_estimators': 200}

Weighted F1 Score: 0.635



```
Accuracy: 0.6485
Weighted F1: 0.6353
Weighted Precision: 0.6736
Weighted Recall: 0.6485

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.34      0.48      1337
           1       0.61      0.81      0.70      2196
           2       0.72      0.55      0.62      1801
           3       0.62      0.77      0.68      2066

    accuracy                           0.65      7400
   macro avg       0.69      0.62      0.62      7400
weighted avg       0.67      0.65      0.64      7400
```

Conclusions:

TF-IDF + SVM outperforming BERT-based models

SVM Weighted F1 Score: 0.787

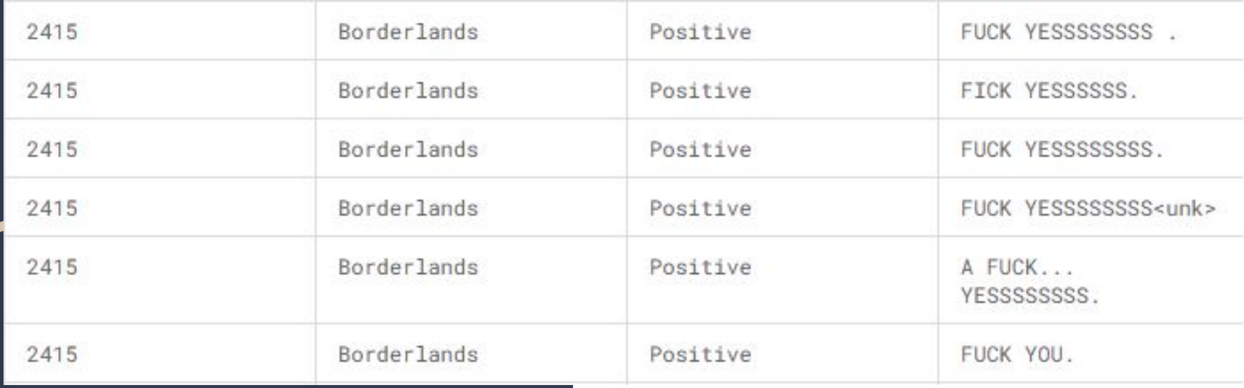
BERT Weighted F1 Score: 0.635

Why?

TF-IDF lacks context

BERT captures “double” context

Hypothesis: The data has been labeled w/o context. Therefore models that ignore context would perform better.



2415	Borderlands	Positive	FUCK YESSSSSSSS .
2415	Borderlands	Positive	FICK YESSSSSS.
2415	Borderlands	Positive	FUCK YESSSSSSSS.
2415	Borderlands	Positive	FUCK YESSSSSSSS<unk>
2415	Borderlands	Positive	A FUCK... YESSSSSSSS.
2415	Borderlands	Positive	FUCK YOU.

Limitations of TF-IDF + SVM

TF-IDF

- Ignores context

"not good" treated as "not" and "good" separately.

- Large vocabularies result in huge matrixes
- Each tweet only uses a subset of words, most entries in matrix are zeros.
- Out of 73,955 training samples, only 15,000 were used due to computation cost in SVM

GridSearch

- Many hyperparameter combinations untested.
- More in depth search requires many fits which means high computation cost.

Limitations of BERT:

Same GridSearch limitations:

- Some hyperparameters untested
- In between values untested

High memory usage

Misspelled words

Acknowledgements - References

https://www.geeksforgeeks.org/machine-learning/understanding-tf-idf-term-frequency-inverse-document-frequency/

https://huggingface.co/docs/transformers/en/model_doc/distilbert

https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/

https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Acknowledgements - Libraries

Libraries used:

kagglehub~=0.3.13

numpy~=2.2.6

scikit-learn~=1.7.2

pandas~=2.3.3

joblib~=1.5.2

pip~=25.3

wheel~=0.37.1

cryptography~=44.0.1

setuptools~=60.2.0