Horror or Comedy? Classifying Two Sentence Stories by Subreddit.

Contents

Background

1. TwoSentenceHorror is a reasonably popular subreddit with about 515k subscribers
2. The challenge is to write a compelling horror story in exactly two sentences

Background

Here’s an example of a two sentence horror story:

“There is a reason why this subreddit is called TwoSentenceHorror. They never live to say the third.”

Background

The TwoSentenceComedy subreddit follows the same idea, but for comedy stories. Here's an example:

"I always clean off my plate no matter how full I am. I have no problem with letting food go to waist."

Background

Which subreddit do you think this story came from?

“My mother said our family had to suffer for the time being. The Time Being likes to watch us suffer.”

Background

It was posted in r/TwoSentenceHorror by u/NemoWatches on October 10, 2020 receiving about 120 upvotes.

Problem Statement

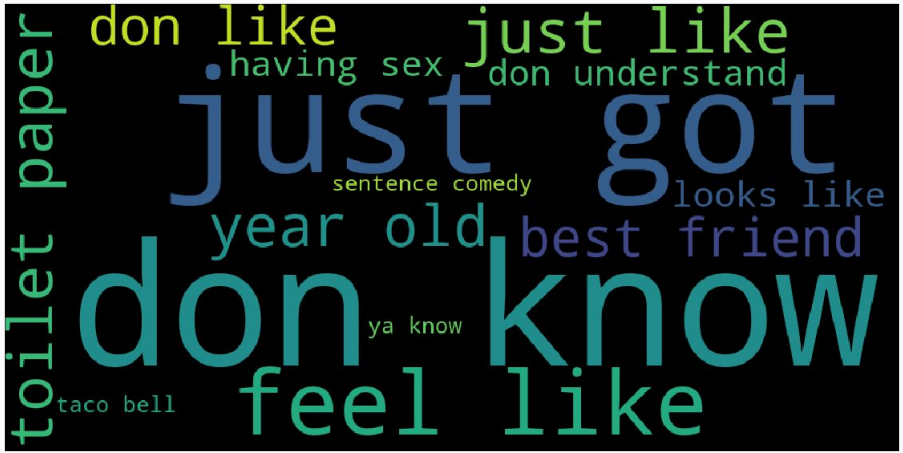
Can machine learning models do better than random guess which subreddit a two sentence story comes from?

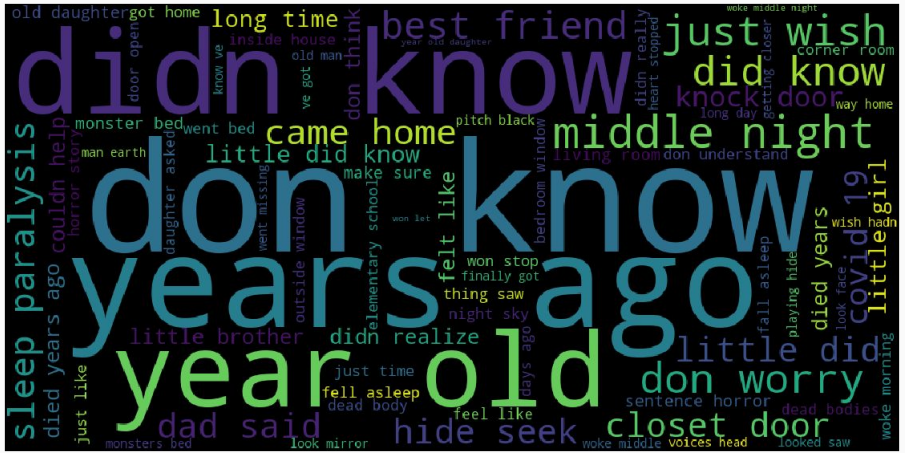
Data Collection

1. Text was obtained from the r/TwoSentenceHorror and r/TwoSentenceComedy subreddits
2. Scraped using the `pushshift` API
3. A total of 3582 posts were obtained
    a. 2366 horror
    b. 1216 comedy
4. Stories with at least 100 upvotes were chosen
    a. These tend to have less spelling and grammatical mistakes
    b. More people agree that these stories belong in the subreddit they've been posted

EDA and Data Cleaning

1. The following word clouds were generated from the 2-grams found in the text obtained from each subreddit.
2. Which word cloud do you think was generated from the horror subreddit?





EDA and Data Cleaning - Issues

1. The r/TwoSentenceComedy subreddit has much fewer subscribers than the horror one
    a. 15.6k vs. 515k
2. This translates to fewer documents to work with
    a. This explains there are almost twice as many horror stories than comedy

Pipeline - Summary

1. Pipeline consists of two steps
    a. Text Vectorization (2 text vectorizors)
    b. Model Fitting (6 models)
2. Each model fitted was fitted twice
    a. Once for each text vectorizer
    b. Each model is also scored twice for each vectorizer

Pipeline - Summary

1. The class `vec_n_model_pipe_fit` was created to simplify this ‘vectorize-fit-score’ flow.
2. It is initialized with train data, test data, and an estimator
3. This class has methods that report the following:
    a. accuracy
    b. precision
    c. recall
    d. Mcc - Matthews’ Correlation Coefficient
    e. confusion matrix
4. We’ll only talk about accuracy and mcc in this presentation

Modeling

1. Logistic Regression
2. k-Nearest Neighbors
3. Random Forest Classifier
4. Adaboost
    a. Decision Tree Classifier as the base estimator
5. SVM classifier
6. Multinomial naive bayes

Modeling - Evaluation

1. Accuracy - proportion of correct predictions
2. Matthews' Correlation Coefficient - mcc
    a. Basically measures the correlation between the predicted and actual values
    b. Returns a number between -1 and 1
        i. 1 means predicted and actual values agree perfectly
        ii. 0 means the predicted values are no better than if they had been randomly assigned
        iii. -1 means the predicted and actual values are exactly opposite
    c. Read more about mcc here:
        i. https://towardsdatascience.com/the-best-classification-metric-youve-never-heard-of-the-matthews-correlation-coefficient-3bf50a2f3e9a
        ii. http://www.bioinfopublication.org/files/articles/2_1_1_JMLT.pdf

Modeling - Best Untuned Model

1. The baseline accuracy (proportion of horror stories) is 66%
2. The best model for testing data was the SVM classifier with Tfidf vectorization
    a. Accuracy - 77%
    b. MCC - 46%
3. A heuristic interpretation of MCC here is that SVM with tfidf is 46% better than randomly guessing

Other Interesting Observations

1. Surprisingly, the next best model logistic regression with tfidf
    a. Accuracy - 76%
    b. MCC - 43%
2. Models using the tfidf vectorizer outperforms those with count vectorization
    a. This is possibly because count vectorization may artificially inflate the importance of certain words
        i. https://www.quora.com/Why-TF-IDF-transformation-works-better-than-Count-Vectorizer-in-Machine-Learning
    b. The exception was the multinomial naive bayes' model
    c. MCC for count vectorizer was 21% vs 20%; 68% accuracy for both
3. The worst performing model was SVM with count vectorization at 34% accuracy and 5% MCC

Back to our original question...

Can machine learning models do better than random guess at classifying two-sentence horror stories?

Back to our original question...

1. The answer is yes!
2. Can we make our models do better?

Pipeline Tuning

1. The models chosen for tuning were
    a. SVM classifier with tfidf
    b. Logistic regression with tfidf
2. GridSearchCV

Pipeline Tuning - Logistic Regression

1. The tuned hyperparameters yielded a test accuracy of 77% and test mcc of 46%
   a. Comparable to the untuned SVM classifier
2. The hyperparameters that yielded this result were:
   a. C - 10,
   b. Penalty - L2
   c. Max_df - 0.9
   d. max_features - 5000
   e. min_df - 3
   f. ngram_range - (1, 2)
   g. ignore stopwords

Pipeline Tuning - SVM

1. The hyperparameters that gridsearch returned did appear to reduce overfit.
2. The train-test accuracy is as follows:
    a. Default parameters: 100% - 77%
    b. Tuned parameters: 76.5% - 76.2%

In conclusion...

1. Machine learning models do appear to do better than random guess to classify a two sentence story
2. Hyperparameter tuning can help improve a model
    a. Refer to the example with logistic regression
3. While hyperparameter tuning did appear to reduce overfit in the case of SVM, it is debatable whether this was helpful

Thank your time!
Merci pour votre attention!