Predicting Response Sentiment Using Tweets

1 Problem

Twitter has grown rapidly in the last decade as a platform for sharing short-form written content. It's used by individuals to share thoughts and jokes, academics to discuss research, and even governments as a primary form of communication. This project seeks to predict response sentiment on a tweet's replies using the tweet's content.

This problem is relevant today since many creators want to gauge how their audience will respond to their content before publishing it. This can also be beneficial to organizations to help rephrase tweets that may be interpreted negatively to protect reputation. Finally, while this project will be focused on Twitter, the techniques can easily be applied to other major platforms like YouTube and Facebook, to help content creators gather insights.

2 Data

We plan on using the Twitter API to collect data. This API is documented extensively and has a fairly large rate limit of 900 tweets per 15-minute window, so we believe we will have no trouble collecting tweets to train a model on. The specific API we plan on using is Tweepy, a popular Twitter API for Python. Tweepy makes it extremely easy to get started with Twitter data. For example, here's how to scrape tweets from a user's timeline:

```
import tweepy

auth = tweepy.OAuth1UserHandler(
   consumer_key, consumer_secret, access_token, access_token_secret
)

api = tweepy.API(auth)

public_tweets = api.home_timeline()
for tweet in public_tweets:
    print(tweet.text)
```

Source: https://docs.tweepy.org/en/stable/getting started.html#hello-tweepy

We will collect the following pieces of information for our model:

- **Tweets:** We'll collect a random sample of tweets that represent Twitter as well as possible. We will likely need at least 10,000 tweets to train an accurate model.

- **Replies:** For each tweet, we'll collect about 1,000 replies. This will help us generate an accurate response sentiment score while maximizing reply diversity.

If using the Twitter API is unsuccessful for any reason (rate limiting, costs, efficiency, etc.) we can use web scraping instead. Scraping tweets is relatively straightforward using browser automation through Selenium WebDriver. We can automatically fetch tweets and their replies through Python and parse the HTML using BeautifulSoup4, a powerful library. This is all possible because Twitter's HTML is well-formed and uses consistent class names for objects like parent tweets and replies.

3 Approach

We plan on training a model with supervised learning using the data discussed above. Each sample in our training set will consist of a tweet's content labeled with a response sentiment score that we'll generate using the replies to that tweet. A more detailed outline is described below.

1. **Data Collection:** We'll collect data using the methodologies described in the above section. The tweets will be split into a training set X_train and a testing set X_test.

2. **Pre-Processing:** We will perform basic pre-processing on all of the tweets and their replies by removing whitespace, punctuation, numbers, special characters, and stop words (short words). We will also tokenize the input and use lemmatization to strip words into their root form. This will allow words like player, played, and plays to be interpreted as their root word play. Lastly, we'll also need to remove the @user token that appears at the beginning of all tweet replies. All of these steps can be done easily using NLTK, a Python text pre-processing package.

3. **Tweet Vectorization:** Creating a feature matrix will involve using TfidfVectorizer, a feature extraction tool available in Scikit-learn. This tool uses term frequency-inverse document frequency (TF-IDF) to build a feature matrix from rows of tokenized tweets. It computes the following ratio: frequency of a word in a document / frequency of a word in all documents. If this number is close to 1, it tells us that a word is unique to a specific document. This turns our language data into a feature set that makes it easy to train models and compare document vectors in linear space.

4. **Label Generation with Sentiment Analysis:** Generating the labels for the tweets can be done by applying sentiment analysis to the replies of a tweet. Sentiment analysis is a natural language processing (NLP) technique used to determine whether text is positive, negative, or neutral. We will compute sentiment scores for each reply and take the mean of them to be our response sentiment score for a tweet. Then, we will divide these ground truth labels into a training set y_train and a testing set y_test that correspond to our training and testing data.

5. **Training:** We will experiment with different models to predict tweet response sentiment. We will try linear models (LinearRegression, LogisticRegression), decision trees (XGBRegressor, RandomForestRegressor), and support-vector machines (SVR).

6. **Testing:** We will use hyperparameter tuning with GridSearchCV to fine-tune these models and improve accuracy on our testing set. The testing set will be critical to steer our models in the right direction and to make sure we are not overfitting on our training set.

7. **Product Development:** If we have time, we would like to create a web-app that implements our final model. This app should allow users to enter their own tweets and see what the predicted response sentiment would be, in order to help them determine how an audience would respond. We can also build a feature that lets users see response sentiments of public tweets.

4 References

4.1 Twitter API

- **Twitter API v2 Rate Limits:** https://developer.twitter.com/en/docs/twitter-api/rate-limits#v2-limits
- **Twitter API for Python (Tweepy):** https://github.com/tweepy/tweepy
- **Mining Tweet Replies with Tweepy:** https://github.com/nirholas/Get-Tweet-Replies-With-Python-Tweepy
- **Minimalist Twitter API for Python (TwitterAPI):** https://github.com/geduldig/TwitterAPI
- **Mining Tweet Replies with TwitterAPI:** https://towardsdatascience.com/mining-replies-to-tweets-a-walkthrough-9a936602c4d6

4.2 Web Scraping

- **Selenium WebDriver:** https://www.selenium.dev/
- **HTML Parsing with Beautiful Soup:** https://www.crummy.com/software/BeautifulSoup/bs4/doc/

4.3 Sentiment Analysis

- **Predicting Sentiment of a Tweet:** https://towardsdatascience.com/social-media-sentiment-analysis-49b395771197
- **Predicting Likes on a Tweet:** https://www.nature.com/articles/s41598-021-86510-w
- **Sentiment Analysis with Python:** https://towardsdatascience.com/a-beginners-guide-to-sentiment-analysis-in-python-95e354ea84f6
- **Text Pre-Processing with NLTK:** https://machinelearningknowledge.ai/11-techniques-of-text-preprocessing-using-nltk-in-python/

4.4 Natural Language Processing

- **Text Pre-Processing with NLTK:** https://machinelearningknowledge.ai/11-techniques-of-text-preprocessing-using-nltk-in-python/
- **Scikit-learn Feature Extraction** https://scikit-learn.org/stable/modules/feature_extraction.html#feature-extraction
- **Term Frequency-Inverse Document Frequency:** https://en.wikipedia.org/wiki/Tf-idf