Data Preprocessing

Data Collection

There are many different Twitter Sentiment Analysis datasets that exist, but our group found that they were either in a hard to work format or the data that was collected was from many years ago and less relevant to the present day, which inevitably led us to choosing this dataset for our project. This dataset came from the Kaggle website (https://www.kaggle.com/jp797498e/twitter-entity-sentiment-analysis) as two CSV files labeled as Twitter_Training.csv and Twitter_Validation.csv. Within each dataset are four columns (without headers) that we labeled "Tweet ID", "Theme", "Sentiment" and "Tweet".

Tweet ID - Predetermined ID numbers per tweet, given by the dataset

Theme - This describes who or what the the topic of the tweet is about or directed toward

Sentiment - This column has the choices of "Positive", "Neutral", "Negative", or "Irrelevant" and describes the overall emotion of the tweet

Tweet - This column contains the actual tweet itself

According to the Kaggle entry of the dataset, the data was collected directly from the Twitter website through, “Automated collection and manual filtering & labeling.” While there is no code to be found on how the data was collected, the Tweets are public domain and Twitter provides many different avenues to download their API for research purposes. One could easily verify that these tweets did actually come from Twitter.com and that they were not collected manually, as tools are openly available to use to download mass amounts of Twitter metadata automatically.

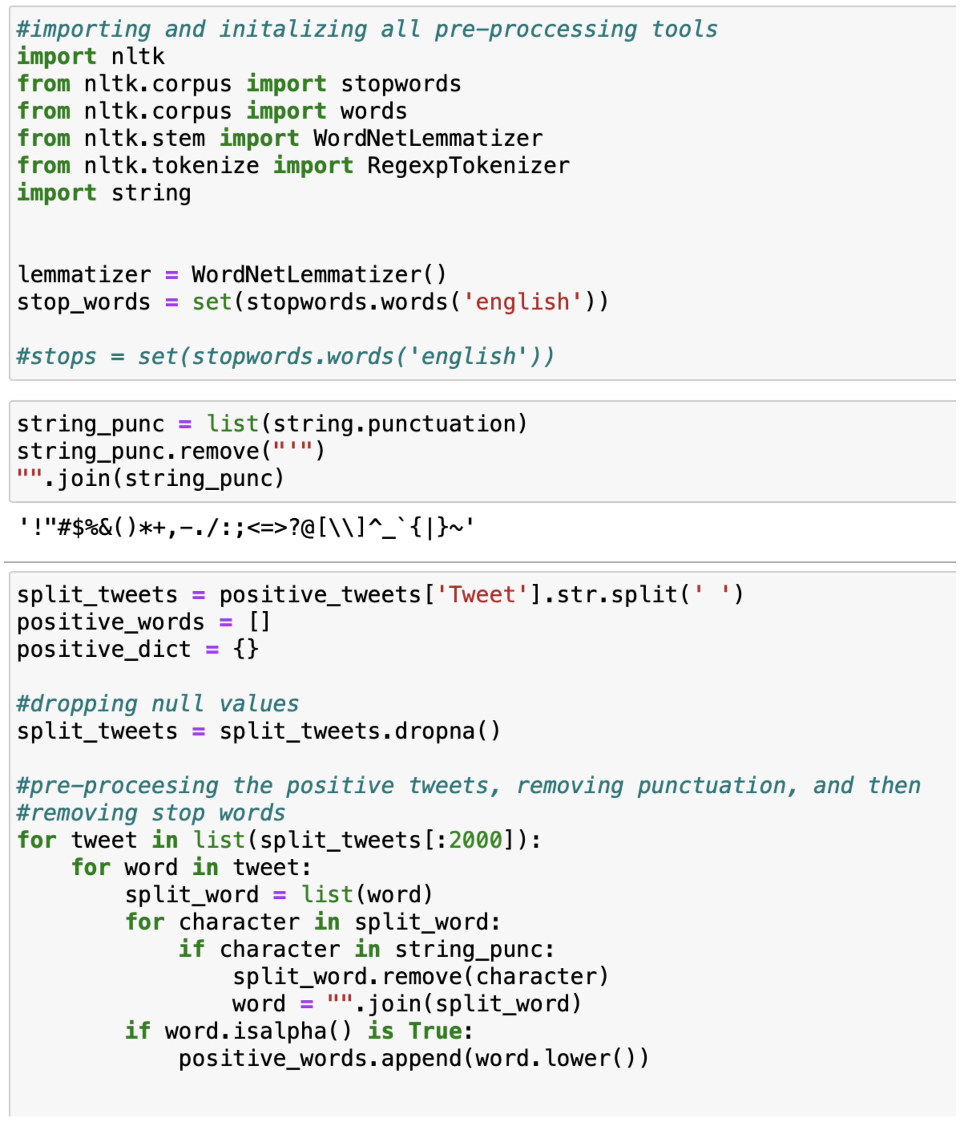
I do not believe that this data could be recollected as it is time sensitive. As each day passes, these tweets will be buried under millions of new Tweets that are being posted, or could have been deleted by the original poster.

Data Management

In terms of data cleaning, our data came very clean and did not require much processing to get it into the state it is currently in. Since the files are pre-split into testing and validation files, the data is at a very clean state and does not contain any null or missing fields.

However, one major processing task our group had to complete was including headers for each of the columns, as the CSV files did not include those directly. This was easily overcome as our group was able to open both CSV files in Excel and add the headers ourselves. This in turn led to an improvement of readability of the data.

One cleaning task we did have to perform on the dataset involved removing “filler” or stop words from the dataset, as they just provided extra fluff and not the words that actually make a sentence positive or negative. This cleaning was done using the built-in library NLTK and then downloading the list of “stop words” (amongst other lists of filler words) and removing these words from our collected positive, negative, neutral, and irrelevant word lists.



```
#importing and initalizing all pre-proccessing tools
import nltk
from nltk.corpus import stopwords
from nltk.corpus import words
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import RegexpTokenizer
import string


lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

#stops = set(stopwords.words('english'))
```

```
string_punc = list(string.punctuation)
string_punc.remove("'")
"".join(string_punc)
```

```
'!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
split_tweets = positive_tweets['Tweet'].str.split(' ')
positive_words = []
positive_dict = {}

#dropping null values
split_tweets = split_tweets.dropna()

#pre-proceesing the positive tweets, removing punctuation, and then
#removing stop words
for tweet in list(split_tweets[:2000]):
    for word in tweet:
        split_word = list(word)
        for character in split_word:
            if character in string_punc:
                split_word.remove(character)
                word = "".join(split_word)
        if word.isalpha() is True:
            positive_words.append(word.lower())
```

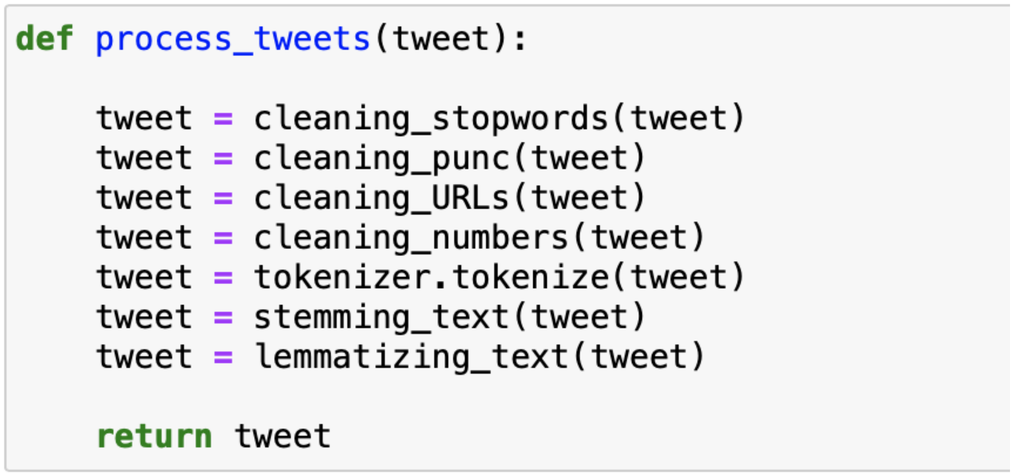
After all of this data was collected and cleaned to its final state, the words were then appended to appropriately titled word dictionaries, such as “irrelevant_dict.” These dictionaries contain only the most relevant words that would entice that emotion, rather than filler words or words with non-alphabetical characters. As all of our data only came from this one dataset, all words were originally used in the initial set of tweets collected, and no other data was merged in. As well as, the only manual data that was manipulated, was the placing of column headers to increase readability on the initial csv files.

~~~~~~

After submitting the midterm report, we actually completely refined our preprocessing functions to help speed up the cleaning of our data. Before the final iteration of the project we were using nested for loops and since the dataset is rather large, the preprocessing and cleaning of the data
~~~~~~

took a very long time. After more research and reviewing other online tutorials we were able to completely change how we cleaned each tweet in the dataset while also dramatically improving the run time. For this part of the project we took inspiration from the last homework assignments in class as well as from the article, “Twitter Sentiment Analysis- A NLP Use-Case for Beginners" by Gunjan Goyal.

In the screenshot below is our “master function” that calls all of the individual cleaning functions on every tweet that is run through it. Each of these functions clean a specific part of the tweet before then passing it as an input into the next function.



```
def process_tweets(tweet):

    tweet = cleaning_stopwords(tweet)
    tweet = cleaning_punc(tweet)
    tweet = cleaning_URLs(tweet)
    tweet = cleaning_numbers(tweet)
    tweet = tokenizer.tokenize(tweet)
    tweet = stemming_text(tweet)
    tweet = lemmatizing_text(tweet)

    return tweet
```

Analysis

Some of the analysis done on the dataset occurred while cleaning the data. To get the data to be ready for the model, it had to be “cleaned” resulting in the removal of punctuation (in our case hashtags from the tweets), removal of foreign characters, and removal of words that contained numbers in them. After the removal of all of these variables in the data, the words were then able to be tokenized and used to train and test the model.

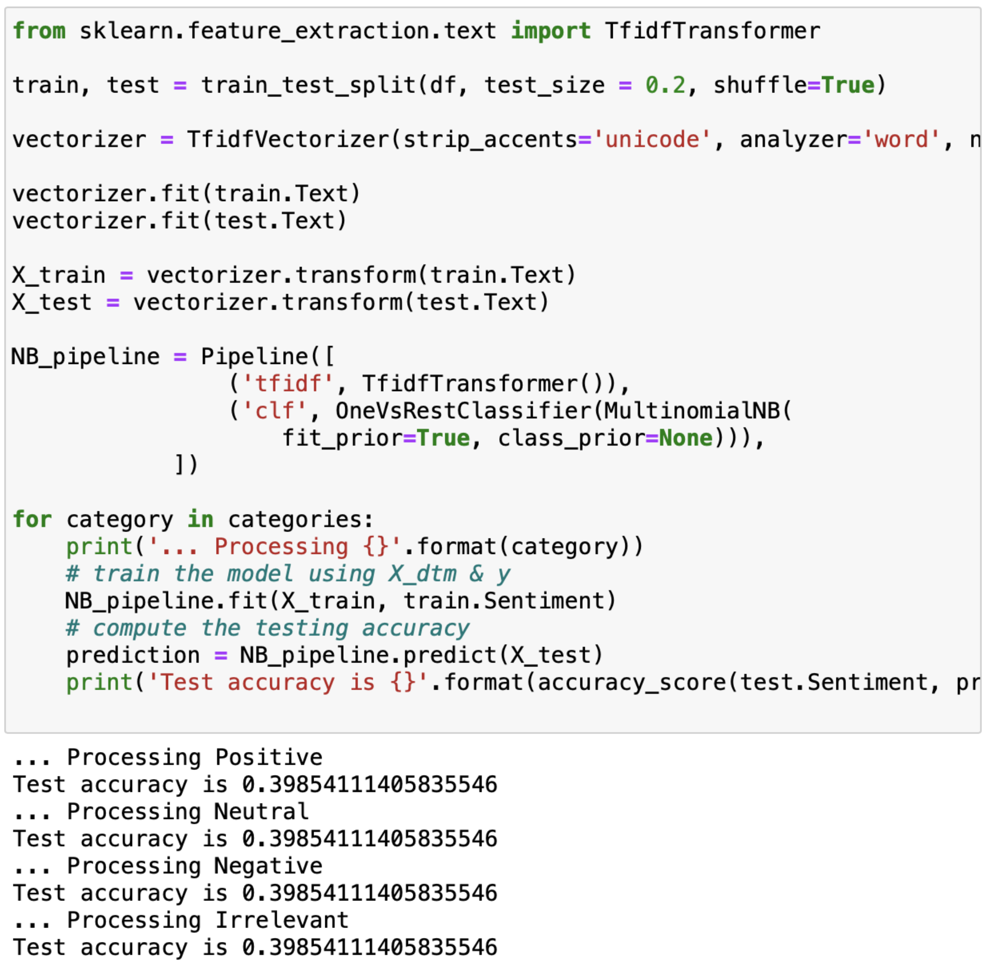
To begin picking out a model, we first had to choose a model that would be able to handle string/token inputs, handle multiple classes of data (positive, negative, neutral, and irrelevant), and be able to provide analysis based on a provided sentence based on the data. As many datasets exist for the purpose of analyzing the sentiment of tweets, there are also many notebooks of code written for many different versions of these datasets. Through our research, as of writing we have decided to use Naïve Bayes and Logistic Regression models to analyze our data. Using the ‘Towards Data Science’ articles by Susan Li titled, “Multi-Class Text Classification with Scikit-Learn“ and “Multi Label Text Classification with Scikit-Learn,” both as guidelines and tutorials to building these models. Initially we tried Logistic Regression and different Tokenizers of other models, but they both were either incompatible with our data type or provided decently low accuracy skills when predicting.

Logistic Regression Model

```
categories = ['Positive', 'Neutral', 'Negative', 'Irrelevant']


df = pd.DataFrame.from_dict(cleaned_training_data, orient='index')
df = df.reset_index(drop=False)
df.columns = ['Text','Sentiment']

train, test = train_test_split(df, test_size = 0.2, shuffle=True)

vectorizer = TfidfVectorizer(strip_accents='unicode', analyzer='word', n

vectorizer.fit(train.Text)
vectorizer.fit(test.Text)

X_train = vectorizer.transform(train.Text)
X_test = vectorizer.transform(test.Text)

#print(X_train)
#print(X_test)

LogReg_pipeline = Pipeline([
            ('clf', OneVsRestClassifier(LogisticRegression(solver='sag',
])

for category in categories:
    print('... Processing {}'.format(category))

    LogReg_pipeline.fit(X_train, train.Sentiment)

    prediction = LogReg_pipeline.predict(X_test)
    print('Test Accuracy is {}'.format(accuracy_score(test.Sentiment, pr
```

```
... Processing Positive
Test Accuracy is 0.41909814323607425
... Processing Neutral
Test Accuracy is 0.41909814323607425
... Processing Negative
Test Accuracy is 0.41909814323607425
... Processing Irrelevant
Test Accuracy is 0.41909814323607425
```

Naïve Bayes Model



```
from sklearn.feature_extraction.text import TfidfTransformer

train, test = train_test_split(df, test_size = 0.2, shuffle=True)

vectorizer = TfidfVectorizer(strip_accents='unicode', analyzer='word', n

vectorizer.fit(train.Text)
vectorizer.fit(test.Text)

X_train = vectorizer.transform(train.Text)
X_test = vectorizer.transform(test.Text)

NB_pipeline = Pipeline([
                ('tfidf', TfidfTransformer()),
                ('clf', OneVsRestClassifier(MultinomialNB(
                    fit_prior=True, class_prior=None))),
            ])

for category in categories:
    print('... Processing {}'.format(category))
    # train the model using X_dtm & y
    NB_pipeline.fit(X_train, train.Sentiment)
    # compute the testing accuracy
    prediction = NB_pipeline.predict(X_test)
    print('Test accuracy is {}'.format(accuracy_score(test.Sentiment, pr
```

```
... Processing Positive
Test accuracy is 0.39854111405835546
... Processing Neutral
Test accuracy is 0.39854111405835546
... Processing Negative
Test accuracy is 0.39854111405835546
... Processing Irrelevant
Test accuracy is 0.39854111405835546
```

Since our data came pre-split into testing and training data files, there was not much segmentation of the data while pre-processing, except for the classes of the data provided by the initial evaluation of the sentiment of the tweet.

As of writing, the only testing we are running on our dataset is through the prediction models that we are building. The highest accuracy rate we have achieved is only around forty percent, so this provides uncertainty of how the model is built. By the final project, we plan to have adjusted the parameters on the model more to get a higher level of accuracy.

~~~~~~

During the final iteration of the project, we actually found another tutorial on Youtube titled, “Twitter Sentiment Analysis (Naive Bayes Classifier) by Artificial Intelligence at UCI.” This tutorial was a follow along tutorial that helped us build our frequency dictionary and the actual Naïve Bayes model.
~~~~~~

To begin using this model, we first cleaned and processed the dataset to tokenize each individual word in all of the tweets. Afterwards, we built a frequency dictionary that counted how many times each word was not only used, but also how many times it appeared in either a negative or positive tweet. Using this dictionary, we were then able to calculate the loglikelihood of each word which would then go on to represent the sentiment of the word. If a word was deemed positive, the loglikelihood of the word would be a positive number, with the higher the positive number the more positive it was. This logic was similarly applied to all negative words.

```python
def create_frequency(tweets, y_value):

    frequency_dictionary = {}

    for tweet, y in zip(tweets, y_value):
        for word in process_tweets(tweet):

            pair = (word, y)

            if pair in frequency_dictionary:
                frequency_dictionary[pair] += 1

            else:
                frequency_dictionary[pair] = frequency_dictionary.get(pair, 1)

    return frequency_dictionary
```

Finally, after all of the words had a sentiment value attached to them, the prediction function could then take a sentence as an input and return an overall sentiment score for the sentence.

```python
for tweet in ['I am happy', 'angry', 'sad', 'I am so excited', 'good']:

    probability = naive_bayes_predict(tweet, logprior, loglikelihood)

    print(f'{tweet} -> {probability:.2f}')
```

```
I am happy -> 0.43
angry -> -1.37
sad -> -0.75
I am so excited -> 0.43
good -> 1.19
```

As one can see, the prediction function is only as accurate as the dataset allows it to be. For example with the sentences, “I am happy” and “I am so excited,” the preprocessing of the tweets would reduce both of these sentences to just the words, “Happy” and “Excited.” However, both of these sentences received the same sentiment score of 0.43, when as a human we could determine that “excited” seems much more positive than “happy.” If provided more time, this is something we would like to address in the project, while also providing the model with more extensive versions of vocabulary, to help improve overall sentiment analysis.

Argument

Through the analysis of our data set, we hope to help the user be able to enter a set of words and be able to understand the emotion of the message. Through the training and testing provided data files, this provides evidence that certain tweets, based on word choice, can be detected as different emotions. Through the causal argument of "a person's word choice can lead to a sentence or phrase being read or heard as a particular sentiment by others," is how we plan to train our model to be more accurate.

With the finishing of our prediction model, we believe that we can analyze the sentiment behind a sentence rather well, but we definitely can tell that there is room for improvement. However, after the midterm report we were struggling to find which model would be a good fit as they all were returning rather low accuracy, but once we settled on Naïve Bayes, everything seemed to come together rather nicely.

As mentioned above in the original Midterm Project Report, we were able to accomplish our goal of being able to train a prediction model and read the sentiment through a tweet from a negative or positive standpoint. However, for our argument we would also need to consider "how" negative or positive the tweet was, not just if it is negative or positive to be able to firmly say our model is accurate.

Design Process

Intervention

Our final design is a web for our stakeholders to have an overview of social media usage. Depending on different groups of stakeholders, the goal of our website is to help them to achieve their purpose of either monitoring their social media usage or marketing purpose. This website is designed for every user so it doesn't require a high level of experience in technology.

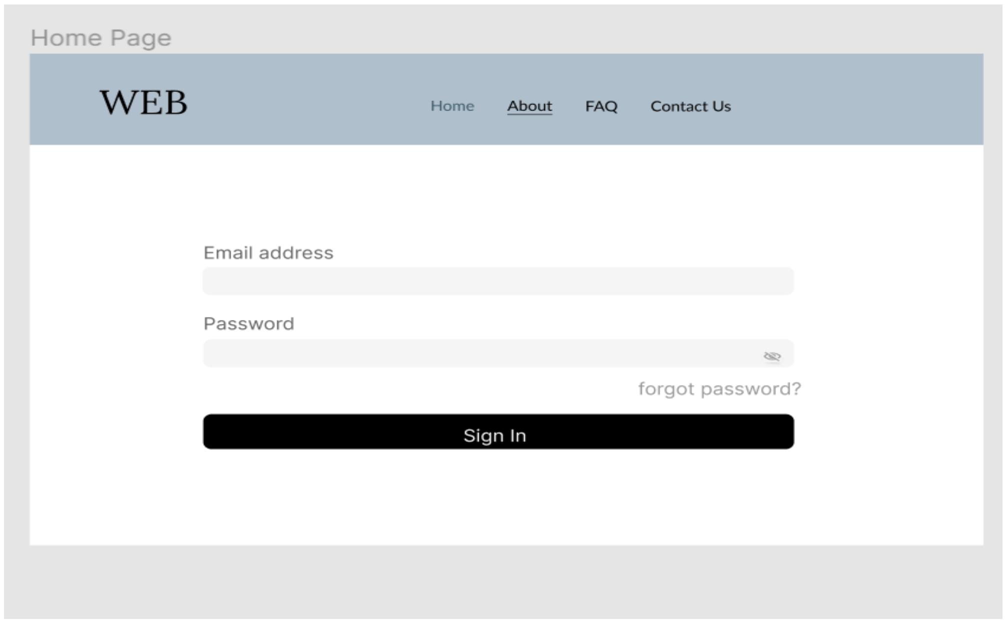
Our success metrics are:

1. For personal use:
    a. # of reports they generate
    b. # of Tweets suggestions they take
2. For marketing purposes
    a. Salse grow
    b. % of potential users turn into actual users

Design and Visualization

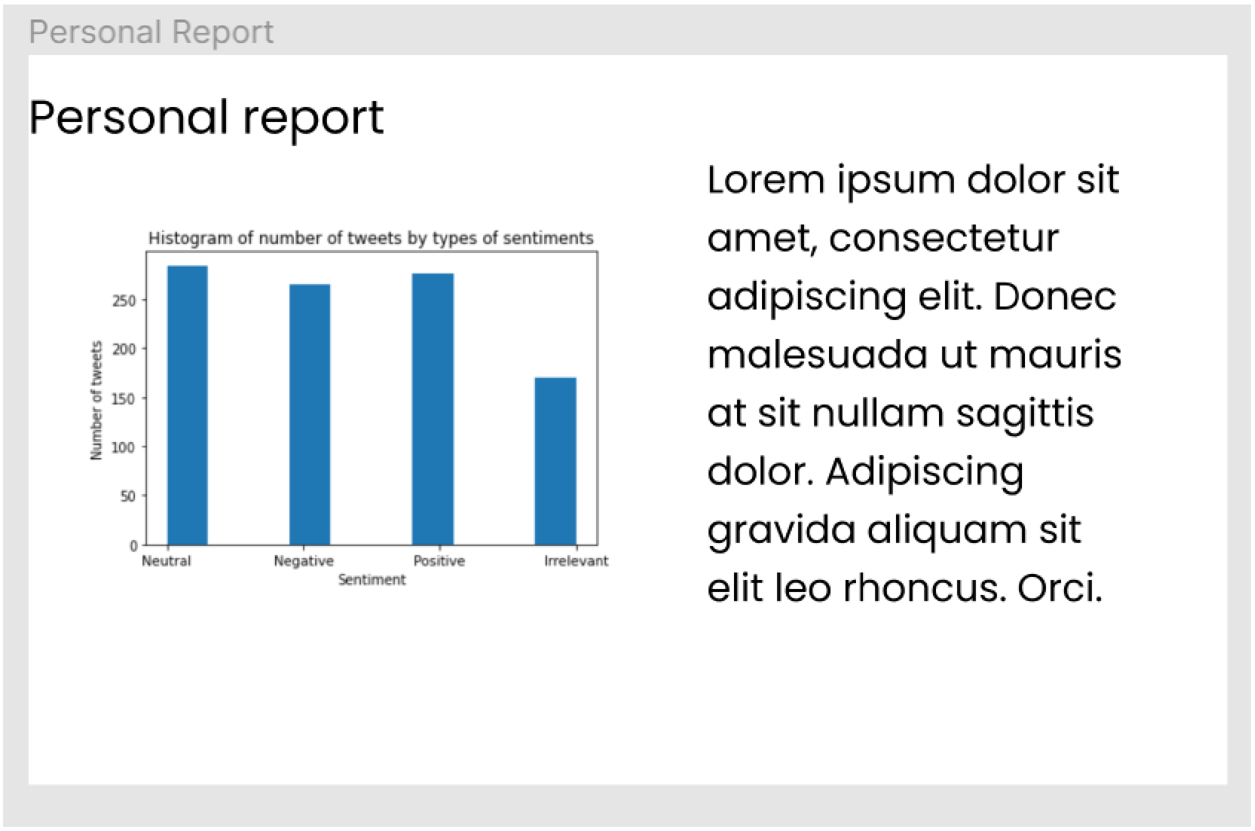
In our design, we focus on different UI for different use cases. We combined design with our visualization

Home page



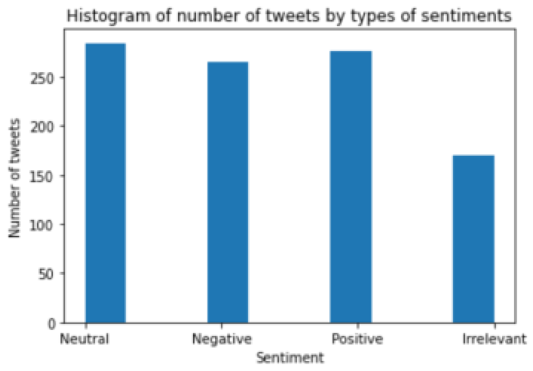
Home page is our opening page for all users which only includes a login feature. However, we will differentiate between individual accounts or enterprise accounts because the databases used for different stakeholders are different.

Personal social media usage



Personal Report

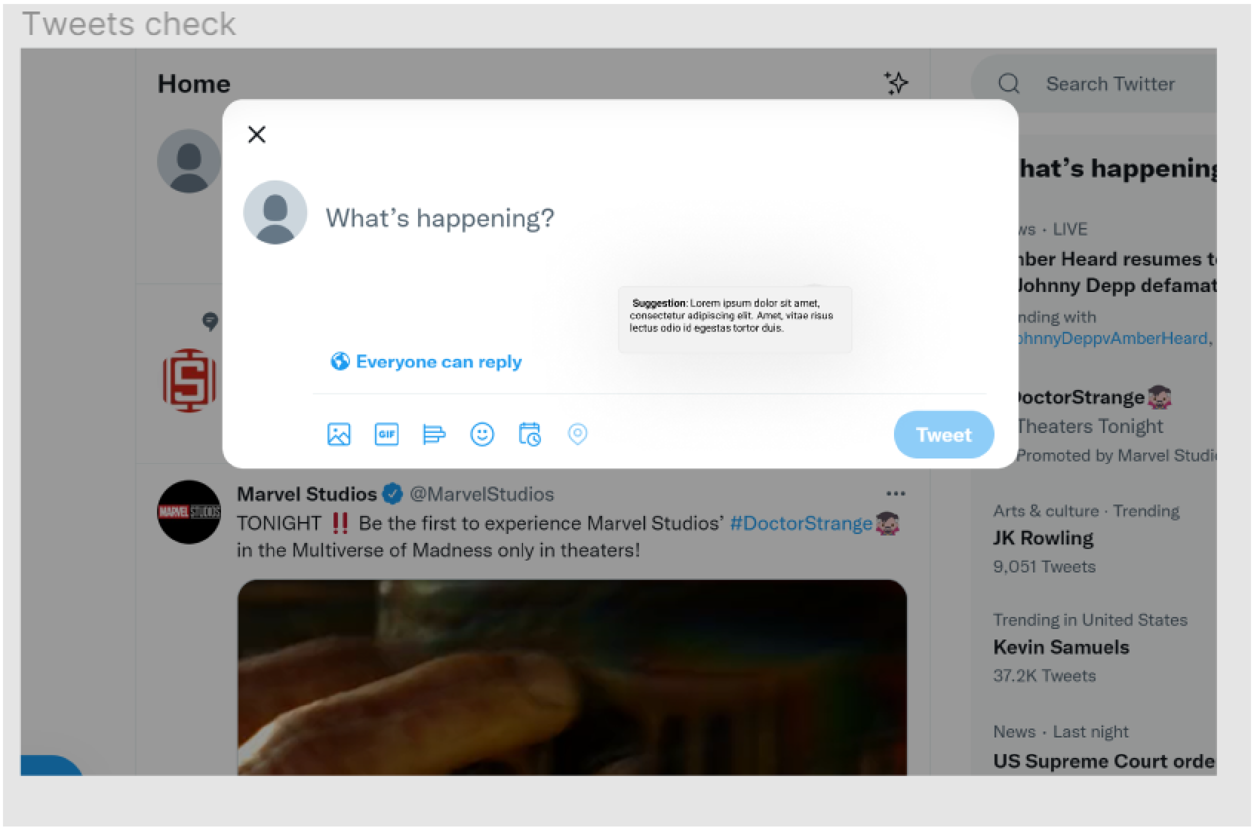
Personal report



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec malesuada ut mauris at sit nullam sagittis dolor. Adipiscing gravida aliquam sit elit leo rhoncus. Orci.

This page serves the use case which for users want to be more retrospective on social media using. We will ask for their permission to get access to gather information from their social media and generate this report. This report will show the sentiments of their tweets.

Tweets check

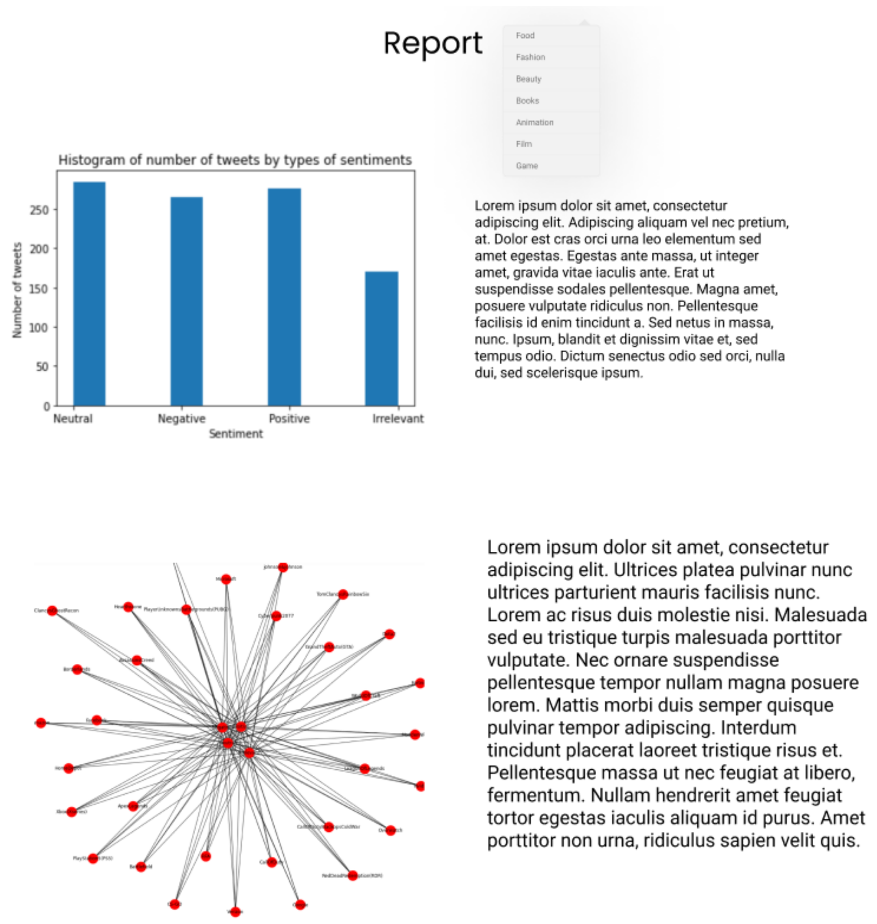


Suggestion: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Amet, vitae risus lectus odio id egestas tortor duis.

Marvel Studios @MarvelStudios
TONIGHT ‼️ Be the first to experience Marvel Studios' #DoctorStrange in the Multiverse of Madness only in theaters!

We reference the presentation feedback for this use case. For this use case, while users are typing their tweet, this feature can detect if there's any red flag of word choices and offer potential suggestions to users.

Marketing Report



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Adipiscing aliquam vel nec pretium, at. Dolor est cras orci urna leo elementum sed amet egestas. Egestas ante massa, ut integer amet, gravida vitae iaculis ante. Erat ut suspendisse sodales pellentesque. Magna amet, posuere vulputate ridiculus non. Pellentesque facilisis id enim tincidunt a. Sed netus in massa, nunc. Ipsum, blandit et dignissim vitae et, sed tempus odio. Dictum senectus odio sed orci, nulla dui, sed scelerisque ipsum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ultrices platea pulvinar nunc ultrices parturient mauris facilisis nunc. Lorem ac risus duis molestie nisi. Malesuada sed eu tristique turpis malesuada porttitor vulputate. Nec ornare suspendisse pellentesque tempor nullam magna posuere lorem. Mattis morbi duis semper quisque pulvinar tempor adipiscing. Interdum tincidunt placerat laoreet tristique risus et. Pellentesque massa ut nec feugiat at libero, fermentum. Nullam hendrerit amet feugiat tortor egestas iaculis aliquam id purus. Amet porttitor non urna, ridiculus sapien velit quis.

This feature serves for enterprises which have marketing purposes. We filter out specific users whose posts are relatable to certain topics. This feature also contains visualizations of databases grouped by sentiments and relationships of each keyword.

Ethics

Of course, privacy is our biggest concern in each use case. Our main functions work around the fact that we are gonna collect users' social media information including what they talked about, viewed, liked or even clicked which is not secure to a lot of users.

This concern mostly happens in our marketing feature because it involves big corporations. Another side of this issue is that we collect data from the social media platform, so we do all our functions based on the privacy agreement of social media platforms and then we run our algorithm to have the outcomes.

For the service we provide for personal use like a social media activity report for each user who installs our extension, we surely are going to ask for permission if any activities involve privacy concerns. Beyond the privacy concern, we think as social media have an increasing effect on how we perceive information and how to spread information in our daily life, having a retrospective review of social media use for each user can be helpful for the whole social media environment.

Citation

Artificial Intelligence at UCI. (2020, November 23). Twitter sentiment analysis (naive Bayes classifier) - youtube. YouTube. Retrieved April 28, 2022, from https://www.youtube.com/watch?v=OsSkjrNjqNI

Goyal, G. (2021, August 27). Twitter sentiment analysis: Implement twitter sentiment analysis model. Analytics Vidhya. Retrieved April 28, 2022, from https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/

Li, S. (2018, April 23). *Multi label text classification with Scikit-Learn*. Medium. Retrieved March 31, 2022, from https://towardsdatascience.com/multi-label-text-classification-with-scikit-learn-30714b7819c5

Li, S. (2018, February 20). *Multi-class text classification with Scikit-Learn*. Medium. Retrieved March 31, 2022, from https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f