Final Report: Sentiment Analysis on Real-Time Streaming Tweets

Team Ninja
Team Members: **Captain-asankar2@**

1) Kunika Sood: kunikas2@
2) Arvind Sankar: asankar2@

Abstract
Sentiment Analysis is used to identify the sentiments of the text source. Tweets can be a very informative data source to collect data on a variety of opinion holders to sentiment analysis. They can help us understand the opinions and perspectives of various people on virtually any topic. There are over 200 million Twitter users with tweets we can mine in real time to get a real time view of how a sample of the population feels about a certain topic. Holistically, our task was to build an ML pipeline and sentiment analysis model that can be trained on tweets, and to build a system that can use the model to predict the sentiment of tweets as they appear in real time.

Overview
(An overview of the function of the code (i.e., what it does and what it can be used for).

The code is broken up into two main parts, the models and the tool. The models have been trained to run on a dataset of tweets about or related to the show “Jeopardy!”. The tool is a wrapper over several Twitter APIs which allow us to read tweets and metadata about the tweets to train the model.

First, we built functionality into the tool that allowed us to **download** tweets using the Twitter Historical Tweet APIs [1]. Then we used the tool to collect a dataset of tweets on the show ‘Jeopardy’ for a definitive timeline (i.e 2 weeks).

Next, we hand labelled the tweet data and marked each tweet as positive or negative sentiment (1 or 0). We followed up and did analysis, preprocessed and cleaned the twitter data to prepare for models specifically - Naive Bayes model and Logistic Regression model. Both the models created were validated and evaluated with F1 Scores (precision and recall).

Once the models were built, we evaluated them against test data (another day of “Jeopardy!” tweets) and noticed pretty good performance. In order to do this, we added an **evaluate** mode that read tweets from an input file and output the results of the model predictions.

Next, we were ready to try our models on real live streaming tweets. To do so, we used the Twitter Streaming APIs [2] in a new tool mode call **stream**. This allowed us to run the models on live incoming data and store the results of the tweets in an output file.

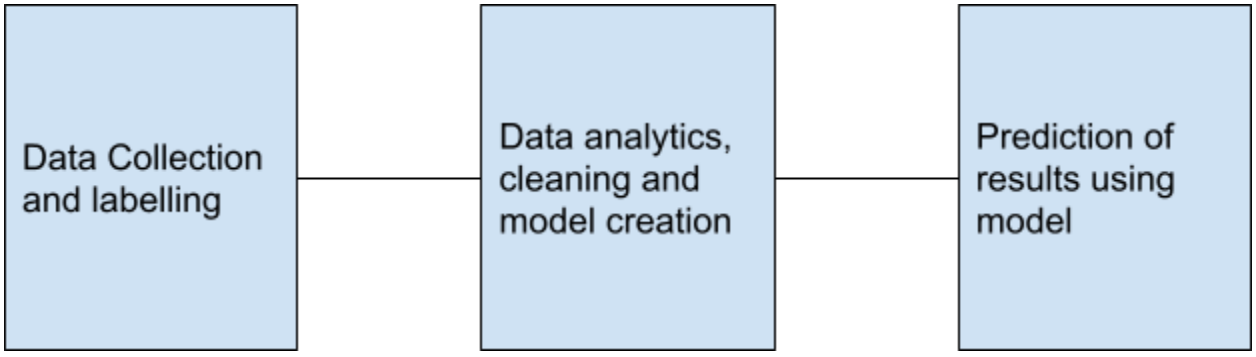
Finally, we wanted to provide a way for the reader (or grader) to reproduce our work in a non deterministic pattern. For this, we created a stream simulator or **simulate** mode from the tool. This mode spins up a separate process that feeds tweets from an input file into our model, as if they were being live tweeted in stream mode. This made it easy to test the model and see its performance, and can easily be replicated by the grader.

This system could allow people to measure audience sentiments in real time. If a show is airing, producers of the show can use this tool to understand what the audience thinks of each scene. This would give them a level of insight that would make subsequent episodes more engaging for the audience. In the case of a show like "Jeopardy!", the producers could read the tweets and gauge sentiment and thereby *change* things that occur on the show for a more positive audience sentiment. For example, if the audience reacts poorly to a category in a Jeopardy game, they could theoretically change the category in future rounds of the game, if needed.

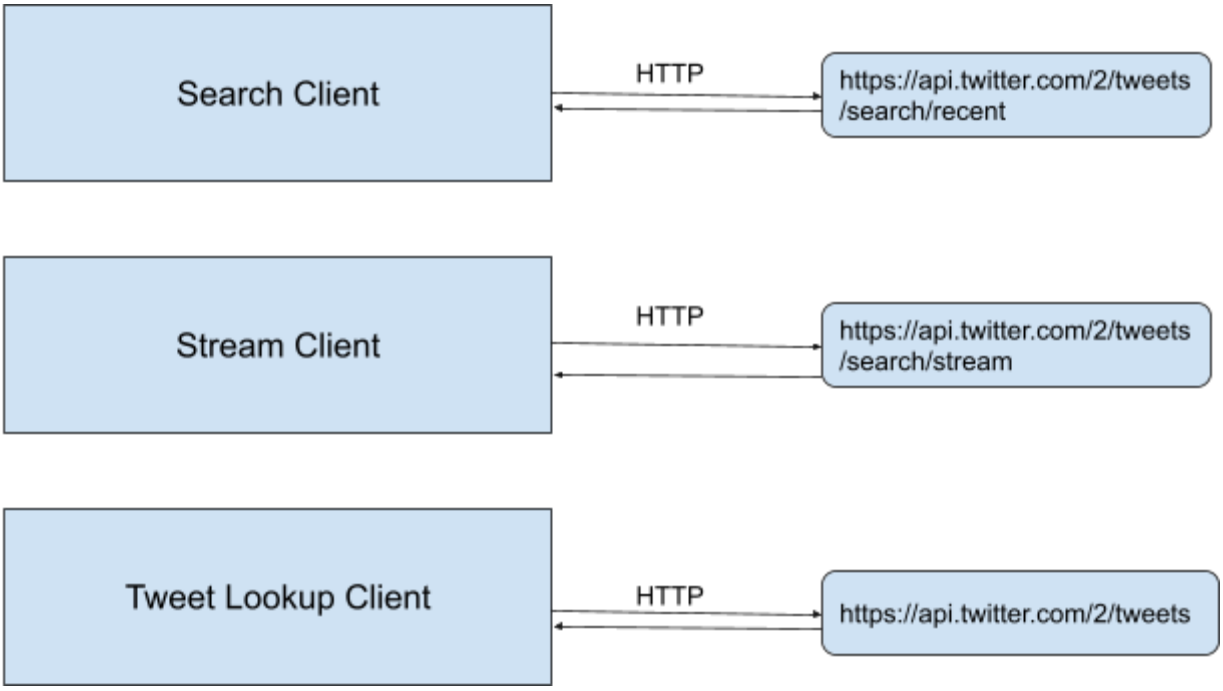
Implementation Details

(Documentation of how the software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.)

The software is implemented in 3 important steps:



To accomplish each of our three steps, we built Twitter Clients that could interface with different Twitter APIs and collect tweet data and metadata for us.



Search Client was used to collect historic data about a certain topic in a certain time frame. We used it to collect tweets about Jeopardy! over a two week window.

Tweet Lookup Client was used to get more information about the tweets that we downloaded. We got info such as time of day, location of tweet, author, etc.

Stream Client was used to download tweets in real time from twitter. This was implemented with the python requests library and this is how we are able to run our model in real time.

Our tool's **download** and **stream** mode use each of these clients to work with the twitter API to accomplish its tasks.

Once we implemented the infrastructure, we were able to collect the data and train the models.

Procedure:

1) **Data Collection and Labelling**
    - Determine TV schedule and show to run on
    - Run **download** mode of the tool to collect all tweets and related metadata and store it in a csv file.
    - Then, each person goes through each row of the csv and marks each tweet as positive or negative in a way the model can read it.

2) **Data analytics, cleaning and model creation:** This part involves usage of collected data from the above step. The data collected is raw and needs to be analysed and cleaned for generating models. The various steps involved are
    - Importing dependencies
    - Loading prefetched and labelled dataset
    - Data Analysis
    - Data preprocessing and cleaning
    - Splitting data into training and testing subsets
    - Using TF-vectoriser to transform data
    - Evaluation using model
    - Creation of model
    - Predicting results using testing subset
    - Predicting results using fresh dataset
    - Serializing model using sklearn package
3) **Prediction of results using models**
    - Run **download** mode again to select a new set of tweets that the model will predict on.
    - Run **evaluate** mode of the tool to collect the model's decision on this new data. A human can parse the decisions the model made and decide if it is ready or not.
    - When the model is ready, we can use **stream** mode to run the model on incoming tweets as the show is airing and read the decisions from a file.
    - Finally, we can use **simulate** mode to test the model and see if it is able to consume the twitter stream and measure things like QPS served.

Creating the Models

The prefetched dataset consists of around 2000 tweets and have been prefetched using download mode and is used to train and test the models. The various columns in data are-

- **tweetId:** unique Id of tweet
- **date:** tweet date
- **polarity:** sentiment of tweet. Either 1 or 0
- **fulltweet:** the text of the tweet

For creating the models suggest running the notebook provided under the notebook folder. Start Jupyter notebook. Go to notebook (`K_load_preprocess_training.ipynb`) and start to run it. The notebook can be run in a step wise manner as directed here - *https://github.com/arvindsankar/CourseProject/tree/main/notebook*

Install dependencies if there are any issues installing using `requirements.txt`

```
(Optional- Install if dependencies didn't install correctly with requirements.txt)
conda create -n my-conda-env                 # creates new virtual env
conda activate my-conda-env                   # activate environment in terminal
conda install jupyter                         # install jupyter + notebook

conda install numpy
conda install pandas
conda install nltk
conda install -c conda-forge matplotlib
conda install -c conda-forge wordcloud
conda install seaborn
conda install scikit-learn
```

Start notebook.

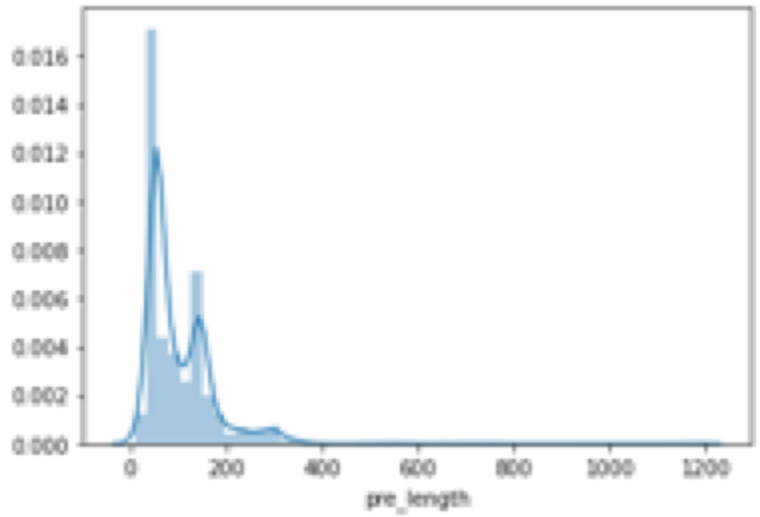
```
jupyter notebook                  # start server + kernel inside my-conda-env
```

Importing dependencies for creating models-

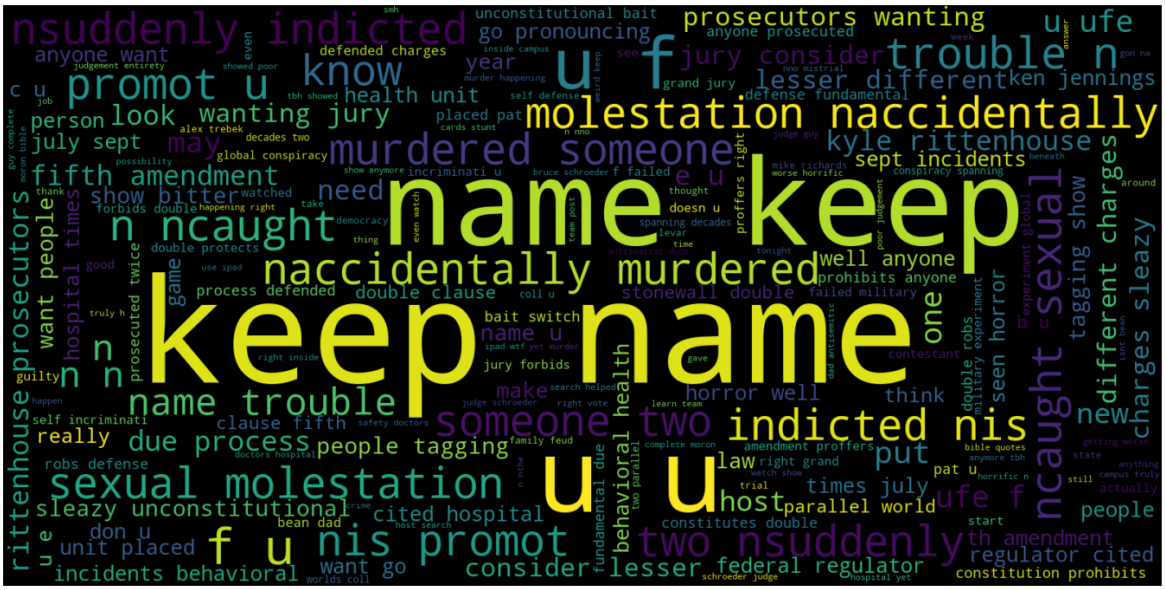
```
import csv
import re
import html
import csv
import nltk
from csv import writer
from ast import literal_eval
import numpy as np
import pandas as pd
import pylab as pl
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud, STOPWORDS
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
#Machine learning
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score
```

Once the dependencies are imported we can read and load the labelled dataset provided under folder ‘notebook’ (week1_labelled_tweets2.csv and week2_labelled_tweets2.csv).

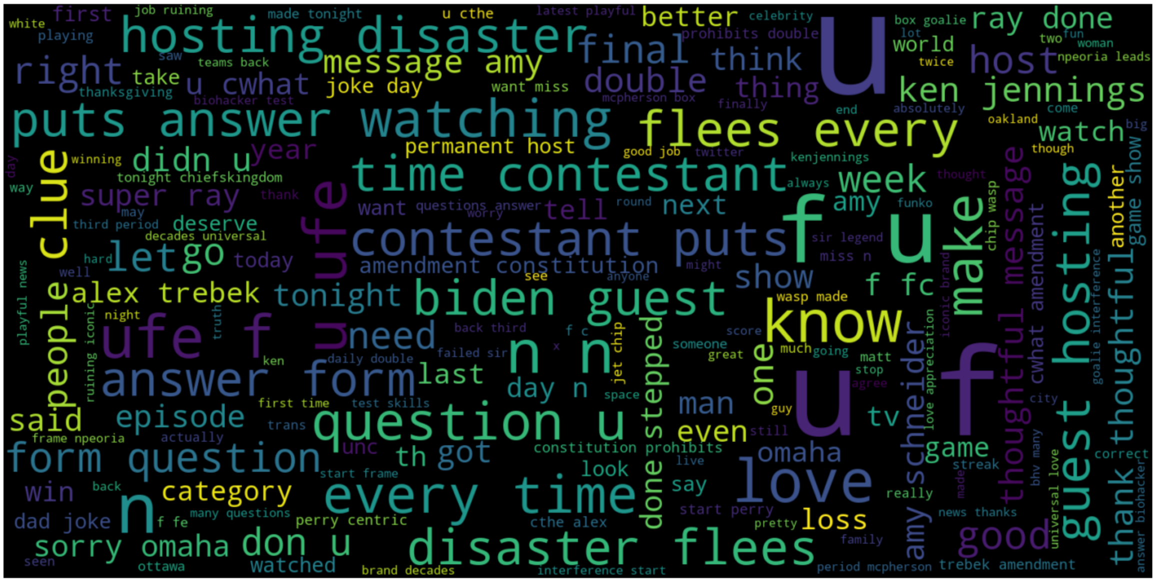
The data is analyzed and cleaned. Cleaning is done by removing urls, html codes, hashtags, mentions, stopwords. Stemming and Lemmatization is then applied on data. Dist plot showing length of data used for model generation.



Wordcloud plotted with negative tweets-



Wordcloud plotted with positive tweets-



After cleaning, the data is split into training and testing subsets. The Vectorizer provided is used to transform a string of text into something more similar to a bag of words model. (Our example one uses TF-IDF weighting that we learned in the Text Retrieval section of our class!).

```
X=df.fulltweet
y=df.polarity
X_training, X_testing, y_training, y_testing = split_into_training(X,y)

vectorize_train_test = TfidfVectorizer(ngram_range=(1,2), max_features=1000)
vectorize_train_test.fit(X_training)
X_training = vectorize_train_test.transform(X_training)
X_testing  = vectorize_train_test.transform(X_testing)
```

Once the models are trained they are used to evaluate test data.

Training Naive Bayes model

```
# Training Naive Bayes model
NB_model = MultinomialNB()
NB_model.fit(X_training, y_training)
evaluate_model_values(NB_model)
y_predict1 = NB_model.predict(X_testing)
print('Accuracy Score of Naive Bayes:',accuracy_score(y_testing, y_predict1))
```

Training Logistic Regression model

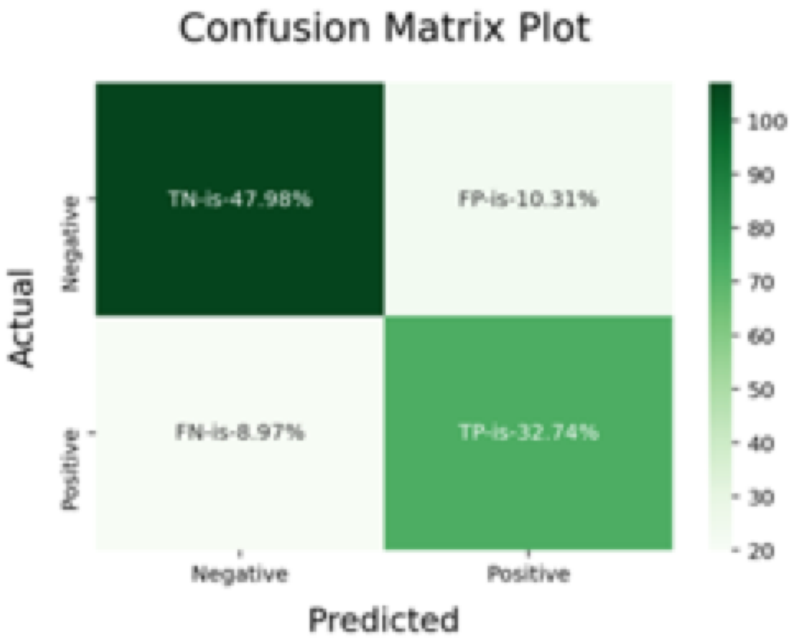
```
# Training Logistic Regression model
LogisticR_model = LogisticRegression(solver='lbfgs')
LogisticR_model.fit(X_training, y_training)
evaluate_model_values(LogisticR_model)
```

```
y_predict2 = LogisticR_model.predict(X_testing)
print('Accuracy Score of LogisticRegression:',accuracy_score(y_testing,
y_predict2))
```

The following are the precision, recall and F1 scores generated by Naïve Bayes and LogisticRegression models we created-

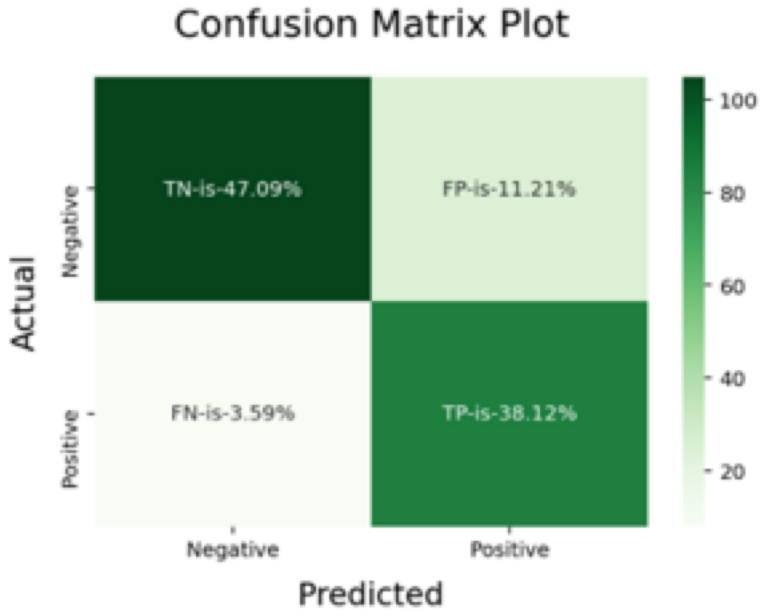
	precision	recall	f1-score	support
0.0	0.84	0.82	0.83	130
1.0	0.76	0.78	0.77	93
accuracy			0.81	223
macro avg	0.80	0.80	0.80	223
weighted avg	0.81	0.81	0.81	223

Accuracy Score of Naive Bayes: 0.8071748878923767



	precision	recall	f1-score	support
0.0	0.93	0.81	0.86	130
1.0	0.77	0.91	0.84	93
accuracy			0.85	223
macro avg	0.85	0.86	0.85	223
weighted avg	0.86	0.85	0.85	223

```
Accuracy Score of LogisticRegression: 0.852017937219731
```



Once the models are trained they can be stored and used for analysis on streaming tweets.

```
from joblib import dump, load
dump(NB_model, 'NB_model.joblib')
dump(LogisticR_model, 'LogisticR_model.joblib')
dump(vectorize_data_file, 'vectorizer.joblib')
```

The Vectorizer provided is used to transform a string of text into something more similar to a bag of words model. (Our example one uses TF-IDF weighting that we learned in the Text Retrieval section of our class!) Here are some example commands you can try to use our models to evaluate some tweets.

Tool Usage Documentation

(Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run a software, whichever is applicable.)

Please use: https://github.com/arvindsankar/CourseProject to read this same information, but rendered in markdown.

We have written a wrapper around our system for the grader to use to run the code to collect tweets, evaluate the model, stream tweets, and simulate evaluating in real time.

There are four modes to use our tool:

```
foo@bar % python tool.py --help

usage: tool.py [-h] {download,evaluate,stream,simulate} ...

positional arguments:

{download,evaluate,stream,simulate}

optional arguments:

-h, --help  show this help message and exit
```

Download Mode (Requires Twitter API keys)

```
usage: tool.py download [-h] --topic TOPIC --start_time START_TIME --end_time
END_TIME [--output OUTPUT]

-h, --help  show this help message and exit
--topic TOPIC Topic for tweets to download.
--start_time START_TIME Start time to download tweets in ISO format.
--end_time END_TIME End time to download tweets in ISO format.
--output OUTPUT File path to store downloaded tweets.
```

As the names and arguments suggest, we can use this mode to download tweets from a given topic in the time range (START_TIME, END_TIME). This is implemented with the Twitter Search API. NOTE: the Twitter Developer API keys only allow us to retrieve data from up to 7 days ago. If you need data from even earlier, you will need to use an Academic Researcher API account.

Remember to pass the start time and end time in ISO format! Here is an example command you may use (adjust the date so you are retrieving data from within the last 7 days).

```
python tool.py download --topic="Jeopardy" --start_time=2021-12-06T03:00:00
--end_time=2021-12-06T03:30:00 --output demo_tweets
```

Evaluate Mode

```
usage: tool.py evaluate [-h] --model MODEL --vectorizer VECTORIZER --input INPUT
--output OUTPUT

-h, --help  show this help message and exit
--model MODEL File path to Sklearn model in .joblib format.
--vectorizer VECTORIZER File path to Sklearn vectorizer in .joblib format.
--input INPUT File path to tweets to evaluate against in format.
--output OUTPUT File path to post tweet_id and sentiment detected.
```

We can use this mode to evaluate pre-trained models on an input file of tweets and produce an output file that contains the output of the model.

The models we will used are stored here:

```
(base) foo@bar % ls models
LogisticR_model.joblib  Naive_Bayes.joblib  vectorizer.joblib
```

Here's what the input data looks like:

```
(base) foo@bar CourseProject % head examples/input_data.csv
Mayim Bialik is back #Jeopardy! https://t.co/Vz5FhkUBYG
@ReallyAmerican1 Garland is building a case. \nwe don't want to go in all HalfAss.
\nDouble Jeopardy applies  \neven in this case
RT @OccupyDemocrats: BREAKING: Amy Schneider becomes the first openly trans person
in history to make it into Jeopardy's "Tournament of Cha\u2026
RT @Ian_R_Williams: @dispositive? I know him! Beloved Law School Professor Making
Jeopardy Debut Tonight https://t.co/XXGQTs3n5x
The latest My Playful News! https://t.co/OMPtg2O2n3 Thanks to @GF_Films
@CallMeCarsonYT @_thejeopardyfan #jeopardy #funko
Ok  have seen both hosts now Ken  and Mayim Bialik.\n\nPrefer Mayim \n\n#Jeopardy
@2ndHandBookery Multiple people said Clue and it makes me so happy. That\u2019s on
my list  also Death Becomes Her  The Big Lebowski  Goodfellas and Double Jeopardy.
So many more but I\u2019ll stick with those \U0001f602
```

```
Another article. https://t.co/kfTaCSLmFb
@Bree_AndersenXO @FeelFreeToPanic He's just toying with  them now  He really thinks he won with CPS.He is putting that baby in jeopardy of being taken away and he doesn't care
@bklvr70 @ryanlcooper I'm of this opinion. I can understand  though  that drs and hospitals are reluctant to put themselves in legal jeopardy by refusing  esp. with the red legislatures eager to lynch medicos for Fox points.
```

Use this command to run our model and vectorizer on this input and store the output to a file called 'output_tweets'.

```
python tool.py evaluate --model models/Naive_Bayes.joblib --input examples/input_data.csv --output output_tweets --vectorizer models/vectorizer.joblib
```

Here's what the output data looks like:

```
(base) foo@bar CourseProject % head output_tweets

0,Mayim Bialik is back #Jeopardy! https://t.co/Vz5FhkUBYG
0,@ReallyAmerican1 Garland is building a case. \nwe don't want to go in all HalfAss. \nDouble Jeopardy applies  \neven in this case
1,RT @OccupyDemocrats: BREAKING: Amy Schneider becomes the first openly trans person in history to make it into Jeopardy's "Tournament of Cha\u2026
0,RT @Ian_R_Williams: @dispositive? I know him! Beloved Law School Professor Making Jeopardy Debut Tonight https://t.co/XXGQTs3n5x
0,The latest My Playful News! https://t.co/OMPtg2O2n3 Thanks to @GF_Films @CallMeCarsonYT @_thejeopardyfan #jeopardy #funko
1,Ok  have seen both hosts now Ken  and Mayim Bialik.\n\nPrefer Mayim \n\n#Jeopardy
1,@2ndHandBookery Multiple people said Clue and it makes me so happy. That\u2019s on my list  also Death Becomes Her  The Big Lebowski  Goodfellas and Double Jeopardy. So many more but I\u2019ll stick with those \U0001f602
0,Another article. https://t.co/kfTaCSLmFb
1,@Bree_AndersenXO @FeelFreeToPanic He's just toying with  them now  He really thinks he won with CPS.He is putting that baby in jeopardy of being taken away and he doesn't care
1,@bklvr70 @ryanlcooper I'm of this opinion. I can understand  though  that drs and hospitals are reluctant to put themselves in legal jeopardy by refusing  esp. with the red legislatures eager to lynch medicos for Fox points.
```

The integer 0 or 1 before the tweet on each line denotes the sentiment the model gives. In our case, the 0 denotes a negative sentiment and 1 denotes a positive one.

Stream Mode (Requires Twitter API keys)

```
usage: tool.py stream [-h] --topic TOPIC --model MODEL --vectorizer VECTORIZER
--output OUTPUT [--duration DURATION]

  -h, --help            show this help message and exit
  --topic TOPIC         Topic for tweets to stream.
  --model MODEL         File path to Sklearn model in .joblib format.
  --vectorizer VECTORIZER
                        File path to Sklearn vectorizer in .joblib format.
  --output OUTPUT       File path to post tweet_id and sentiment detected.
  --duration DURATION   How long to stream tweets for (in seconds).
```

We can use this mode to evaluate pre-trained models on a stream of incoming tweets and produce an output file that contains the output of the model. Note we use the joblib sklearn package to serialize and deserialize our models. Documentation on this package can be found here: https://joblib.readthedocs.io/en/latest/

The Vectorizer provided is used to transform a string of text into something more similar to a bag of words model. (Our example one uses TF-IDF weighting that we learned in the Text Retrieval section of our class!) Here are some example commands you can try to use our models to run models on a stream of tweets.

```
python tool.py stream --topic="cats" --model models/Naive_Bayes.joblib --output
output_tweets  --vectorizer models/vectorizer.joblib --duration 30
```

This command will download a stream of *real-time* tweets for 30 seconds on the topic "cats". This means any tweet with the word "cat" will get parsed, evaluated and the result will be stored in the output_tweets file.

The integer 0 or 1 before the tweet on each line of the output_tweet file denotes the sentiment the model gives. In our case, the 0 denotes a negative sentiment and 1 denotes a positive one.

Simulate Mode

```
usage: tool.py simulate [-h] --model MODEL --vectorizer VECTORIZER --input INPUT
--output OUTPUT [--duration DURATION]
```

```
-h, --help  show this help message and exit
--model MODEL File path to Sklearn model in .joblib format.
--vectorizer VECTORIZER File path to Sklearn vectorizer in .joblib format.
--input INPUT File path where tweets are read from.
--output OUTPUT File path to post tweet_id and sentiment detected.
--duration DURATION How long to stream tweets for (in seconds).
```

Since the stream mode requires the user to have Twitter API keys, we have created another mode for the reader to try our system that does not require them. This mode is just like the Stream Mode but it uses a stream simulator to read tweets from a file and send the tweets to our model process over some duration in a random way. For details on how this is implemented, please read the project documentation.

```
python tool.py simulate --model models/Naive_Bayes.joblib --output output_tweets
--input examples/input_data.csv --vectorizer models/vectorizer.joblib --duration 30
```

This command will send the tweets in the example directory to the model process and record the results to the output_tweets file with the sentiment recorded.

Here is an example with the sentiment:

```
(base) foo@bar CourseProject % head output_tweets

0,@JeopardyGuesser @missmayim #blindguess #Jeopardy going with
Dadaism\t\t\t\t\t\t\t\t\n
1,RT @HowardU: Tune in TOMORROW\tDec. 7th! HU\\u2019s very own Dr. John
Harkless\tassociate professor\tDepartment of Chemistry\twill appear as a
co\\u2026\t\t\t\t\n
0,@bklvr70 @ryanlcooper I'm of this opinion. I can understand\tthough\tthat drs and
hospitals are reluctant to put themselves in legal jeopardy by refusing\tesp. with
the red legislatures eager to lynch medicos for Fox points.\t\t\t\t\t\n
1,RT @QuakeMedia: "Jeopardy!" contestant Amy Schneider (@jeopardyamy) making
history as the first trans-woman to qualify for the
#TournamentO\\u2026\t\t\t\t\t\t\t\t\n
1,RT @madimoukas: Coming out of tweeting retirement for @KenJennings. For the love
of god please make this man the host and stop toying with\\u2026\t\t\t\t\t\t\t\t\n
0,@richjhewitt #Jeopardy https://t.co/NyI7bwtTKR\t\t\t\t\t\t\t\t\n
1,Watching the news is depressing. I never thought in my lifetime that we could
lose our democracy in the United States! Yes\tthe USA is in jeopardy of a take over
of our government and the Republican party and a hugh portion of Americans seem to
be okay with it. Wake up!\t\t\t\t\t\t\t\n
1,@Jeopardy @missmayim. It is a common error\tbut Farsi is neither the official
language of Persia nor the same as Persian.  Persian is Persian and Farsi is Farsi.
```

```
Today\\u2019s game perpetuates the misunderstanding that Farsi is a common name for
Persian. They are not interchangeable.\t\t\t\t\t\t\t\n
1,Girl Groups is a category on Jeopardy right now and all I wanted was for The
Runaways to be an answer\tit was the $2000 clue and NO ONE GOT IT.\t\t\t\t\t\t\t\n
0,@Jeopardy Yes\tit was great!\t\t\t\t\t\t\t\n
```

Contributions

Brief description of contribution of each team member in case of a multi-person team.

Task	Details	Team member who worked
1. Collect historic tweet data	Create a Twitter developer account, learn which APIs we need, etc.	Arvind Sankar
2. Fetch tweets relevant to topic at a defined time interval	Write code that uses API to fetch tweets relevant to a pre defined topic at a predefined time interval	Arvind Sankar
3. Data Preprocessing	Label Training Data	Arvind Sankar and Kunika Sood
4. Data Analysis and Preprocessing to Prepare for Specific Model	Apply stemming, tokenization, stopword removal, etc.	Kunika Sood
5. Split data as Training and Test Subset	Split labeled data as Training and Test set in N samples (90/10 ratio)	Kunika Sood
6. Model Creation with Training	Train models	Kunika Sood
7. Model Validation with Test Data	Validate the models with test data	Kunika Sood
8. Run Experiments on Streaming data	Write code that fetches data / applies preprocessing / sends to the model / records result	Arvind Sankar
9. Evaluate the results of the model on Streaming data.	Label Testing Data	Arvind Sankar
10. Presentation and demo video	Record a demo of model running evaluations on a live twitter topic.	Arvind Sankar and Kunika Sood