Connect & Discover

Postgraduate Certificate in Software Design with Artificial Intelligence

Advanced Machine Learning and Neural Networks - (AL_KSAIG_9_1)

Major Assignment – Visualising Twitter Data

Student ID: A00267948

Student Name: Daniel Foth

GIT: https://github.com/DanielsHappyWorks/Visualising-Twitter-Data

Brief Description:
Phase 1: Use tweepy to gain a critical mass of tweets regarding a chosen topic and annotate the sentiment of the tweets; positive, negative, neutral.

- Clean and organise the data, remove duplicates and use techniques from text classification exercise such as stopping and stemming to prepare the dataset.

Phase 2: Create a model which can accurately assign a polarity to a group of new tweets within the chosen topic. Test your model against a different domain e.g. politics model on sports or tech.

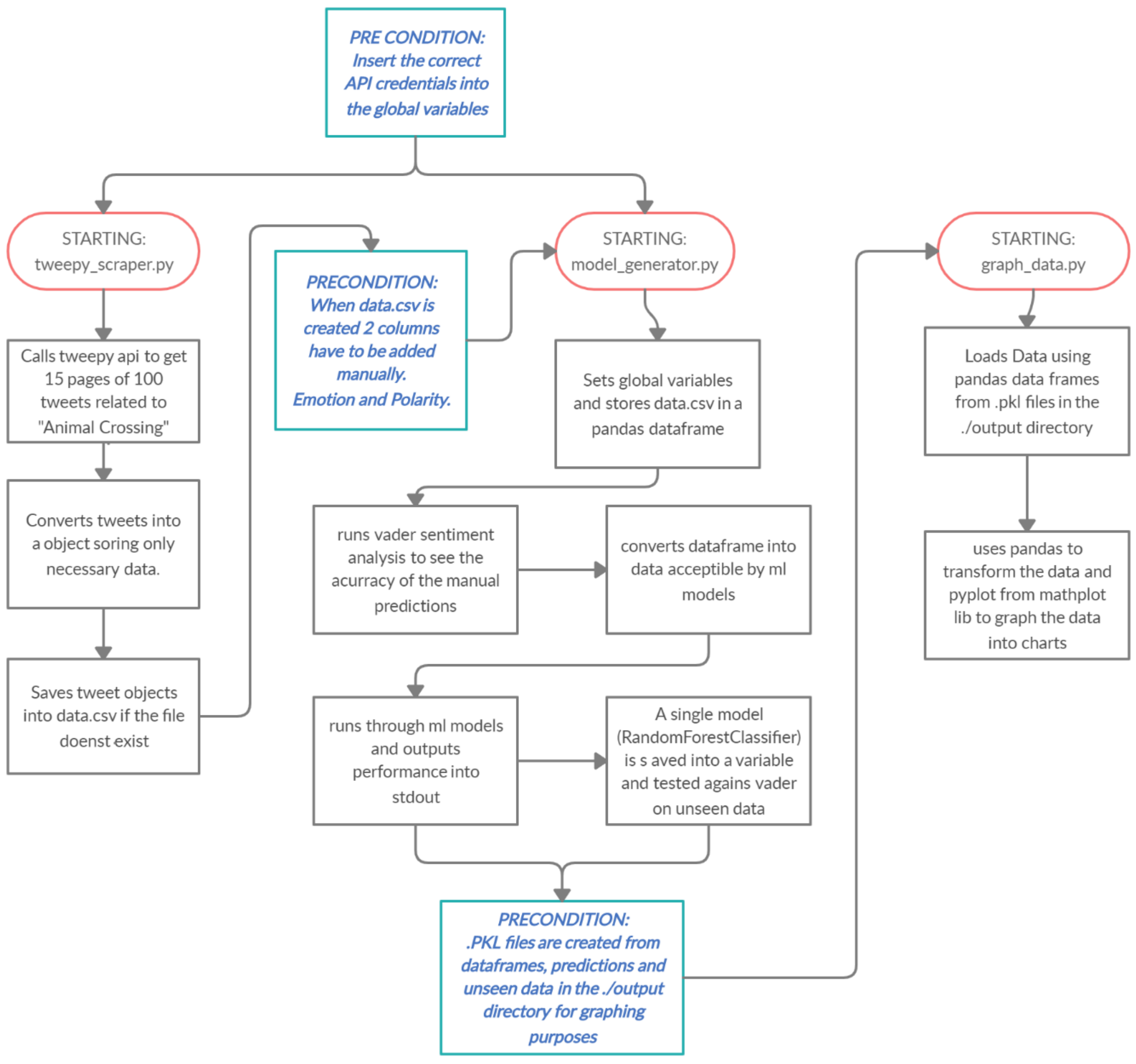
Deliverables

1. A dataset with annotated training data for sentiment polarity
2. A model which predicts polarity of given tweet
3. A report which includes a comparison with VADER (an off the shelf sentiment analysis tool)

Contents

Program Design

Design Diagram


PRE CONDITION: Insert the correct API credentials into the global variables
STARTING: tweepy_scraper.py
Calls tweepy api to get 15 pages of 100 tweets related to "Animal Crossing"
Converts tweets into a object soring only necessary data.
Saves tweet objects into data.csv if the file doenst exist
PRECONDITION: When data.csv is created 2 columns have to be added manually. Emotion and Polarity.
STARTING: model_generator.py
Sets global variables and stores data.csv in a pandas dataframe
runs vader sentiment analysis to see the acurracy of the manual predictions
converts dataframe into data acceptible by ml models
runs through ml models and outputs performance into stdout
A single model (RandomForestClassifier) is s aved into a variable and tested agains vader on unseen data
PRECONDITION: .PKL files are created from dataframes, predictions and unseen data in the ./output directory for graphing purposes
STARTING: graph_data.py
Loads Data using pandas data frames from .pkl files in the ./output directory
uses pandas to transform the data and pyplot from mathplot lib to graph the data into charts

(Created with https://creately.com/)

The Program Files

code/Tweet_scraper.py

It is responsible for scraping tweets from the Tweepy API. It can scrape up to 100 tweets per page. When run it will generate a data.csv file with 1500 tweets on the topic of “Animal Crossing” if it doesn’t already exist.

code/Model_generator.py

It will create models and run one against multiple categories on unseen data. It uses the data.csv file. It requires that 2 extra columns are added in manually. The Emotion and Polarity column. The predictions get stored as data frames into .pkl files along with the tweet data for graphing purposes.

code/Graph_data.py

It graphs the data in the ./output/ directory and stores the graphs as png files under ./output/graphs.

code/Tweet.py

The Tweet class processes a line of tweet data from the API into data that can be used as part of the model creation and graphing. See the data section to understand what is extracted.

Data Set

The data set is stored as data.csv in the code directory. It contains 513 of the 1500 tweets scraped by the tweet_scraper.py script. Some tweets have been removed as they were illegible and manually annotating that many tweets is time consuming, so the data had to be limited.

The tweet data that was saved consists of:

1. *Tweet ID*
2. *Creation Time*
3. *Retweet Count*
4. *Favourite Count*
5. *Source*
6. *Username*
7. *User Location*
8. *Content*
9. *Basic Content – this is the same as the content but stripped of special characters so it can be used with a variety of models.*

The two extra columns annotated manually are:

10. *Emotion – defines the emotion that stands out most in the tweet*
11. *Polarity – a rating of weather the tweet is Positive, Neutral or Negative*

The unseen data used for testing the model predictions against Vader is always retrieved live. Once predictions are made, they get stored as .pkl files for graphing. Alternatively, they can be opened using the pandas “read_pickle” function.
The topics of interest include:

Politics:

1. *Donald Trump*
2. *Boris Jonson*

State of the world:

3. *Covid-19*
4. *Isolation*

New Technology

5. *Virtual Reality*
6. *Ps5*
7. *Xbox Series X*

Data Set vs Vader

Neither predictions are perfect due to machine and human error. We as humans can be very opinionated and my ideas of what is positive or negative can vary from others. I tested my manual predictions against Vader Sentiment Analysis to see what the results were like.

Initially the Accuracy was about 50%. This triggered me to review my datasets polarity with a colleague to see if maybe there are inaccuracies in my predictions. In different cases it seemed like me and Vader were wrong.

After reviewing the data an accuracy of 60% has been reached with Vader. There were mistakes on both sides and not all were easily resolvable. I think that this kind of accuracy is okay since some of the data can't really be classified at all as context is included in pictures and responses to others.

The Results:

(My data set as actual, Vader as predicted)

All Data:

Accuracy: 0.6003898635477583

Confusion Matrix:

121	26	16
70	96	26
47	20	91

Test Data Used on Models:

Accuracy: 0.6233766233766234

Confusion Matrix:

44	5	8
19	27	7
15	4	25

Tested Models

A multitude of models have been tested against this data with different data configurations. The models were:

The Features

The best data configuration must be when the only feature was the tweet content. Other features generally decreased the overall performance or had no affect other than slowing down the processing.

The Processing of Text

All models use a TfidfVectorizer to process the text into values that can be easily fed into the model. The stemmer and stop words from the nltk library are used to pre-process the text and make sure we pass in the best possible input is passed to the model. Some characters like the # or @ are removed from the text as the nltk tokenizer would treat them as words which would affect the predictions in a weird way.

The Tested Models

The tested models include:

1. MLPClassifier
2. SVC
3. RandomForestClassifier
4. LinearSVC

5. MultinomialNB

Output for all of these can be obtained from the stdout by running model_generator.py
All algorithms performed around (Post Optimisation):

- 80-86% Accuracy on all data from data.csv
- 45-60% Accuracy against 30% test data
- 45-55% Accuracy against Vader predictions on all data from data.csv
- 45-55% Accuracy against Vader predictions on all data from data.csv
- 30-50% Accuracy against Vader predictions on 30% test data

Since the results were all so similar, another approach needed to be taken when selecting the best algorithm overall. It is how well the classifier performed on other topics which is described in the section below.

The Chosen Model

The model that performed best overall must be the RandomForestClassifier. It performed like all models but performed way better on unseen data from different topics. Most models would predict all unseen data to be only Neutral or only Negative (Especially the NN and SVC classifier) this was very visible from the confusion matrix output. Testing this involved manual scanning through tweets for each model to see how accurate the predictions were.

Within the model_generator.py code, to see performance in unseen data, move the "model =" from line 234 (RandomForestClassifier by default) to a different classifier.
Line 226 for MLPClassifier
Line 230 for SVC
Line 238 for LinearSVC
Line 241 for MultinomialNB

The RandomForestClassifier did its best to classify tweets into their correct categories. This needs more testing as the live unseen data has never been categorised and the only comparison we have is Vader which is only 60% accurate vs the dataset the models learned on.

Unseen Data from Different categories

Each category of unseen data has up to 500 tweets; 100 tweets for 5 pages if available.
The categories include:

Politics:
8. *Donald Trump*
9. *Boris Jonson*

State of the world:
10. *Covid-19*
11. *Isolation*

New Technology
12. *Virtual Reality*
13. *Ps5*
14. *Xbox Series X*

Vader comparison: Donald Trump

Accuracy 0.306
Confusion Matrix:

22	128	9
1	118	6
8	195	13

Vader comparison: Boris Jonson

Accuracy 0.3389830508474576

Confusion Matrix:

5	18	0
1	11	0
1	19	4

Vader comparison: Covid-19

Accuracy 0.28857715430861725

Confusion Matrix:

29	168	7
2	108	0
10	168	7

Vader comparison: Isolation

Accuracy 0.194

Confusion Matrix:

44	121	11
1	31	0
30	240	22

Vader comparison: virtual reality

Accuracy 0.48414376321353064

Confusion Matrix:

25	155	17
8	196	0
9	55	8

Vader comparison: ps5

Accuracy 0.352

Confusion Matrix:

16	203	31
3	147	7
0	80	13

Vader comparison: Xbox Series X

Accuracy 0.396

Confusion Matrix:

14	182	43
1	173	4
1	71	11

Conclusion

In conclusion, all models performed okay on tweets about “Animal Crossing”. Based on reviewing tweets, Vader isn’t perfect. Tweets out of context may not make as much sense as written text so it can be very hard to categorise them. This is why it’s not surprising that mine and the Vader sentiment analysis tool answers can be so different.

On unseen data, the models didn’t perform as well. RandomForestClassifier did best but even it was able to only classify around a third of the tweets in the same way as Vader. There is a chance that some tweets are classified incorrectly by Vader and correctly by this tool but even then, the increase would probably be insignificant. I would estimate the average score to probably be around 35-45% if the tweets were classified by me.

“Animal Crossing” was not a good choice in topics too. A lot of tweets in relation to covid-19 and Isolation would have been positive as far as the dataset is concerned. This is because people enjoyed playing the game while all of this is happening. It is letting them take their mind off the serious issues plaguing the world right now. This could have impacted the model in a way where it is categorising some words normally associated with negativity or positivity incorrectly overall.

Lastly, the model was small with only 513 rows. Ideally it would be bigger so the models could learn better and be able to distinguish any incorrect categorisation in the dataset if enough data was provided. Also having more people categorise the data could make the model more accurate as tweets would be more accurately categorised by polarity.