Be able to be connected to the Stanford Sentiment Corpus, this one was chosen as, in accordance with the literature, it is the largest publicly available twitter dataset with 1,6 million tweets. It is however not the most accurately sorted as it is all sorted automatically. It also has a scaling system where the results are not strictly binary. This is good for our purposes as this is what I wanted to do.

The results will be looked at and compared to our results. They have been collected from 2009 twitter, which means potentially the differences between the two datasets are: difference in slang that might make results less accurate. Also difference in tweet length, the tweets changed from max 180 to 260 in 2017, while TechCrunch says that that didn’t really change the length of tweets they looked at it one year later and did state that it might be that people are still used to tweeting as they used to [1].

There is also another study that says that it barely changed the length of what people were tweeting, but it was performed on Dutch tweets and was performed only two weeks after the change in character limits [2]. So is probably not super accurate.

The Stanford sentiment dataset is sorted automatically and assumes that any tweet with 😊 is positive and any tweet with ☹ is negative. This is obviously flawed. However it doesn’t actually matter that much as the algorithm used can just use the data and remove the column that sorts the tweets.

This dataset will be used for the formatting of ours, will use the same type of formatting so that both can be compared.

The dataset wasn’t necessarily completely objective on which tweets selected as it used specific queries to collect the tweets. The paper about the dataset can be found here [3].

The different fields on the csv: the polarity of the tweet (0=negative, 2=neutral, 4=positive), id of the tweet, date of the tweet, the query, the user, the text of the tweet.

While only the text data will be used to classify the tweets, the other fields will be included in order to be able to connect the dataset built to the Stanford one.

**Does it matter that the username is in the data?**

Is hybrid sentiment analysis machine learning techniques that you use lexicon method to sort tweets first into sentiments, then can be evaluated by me, and given scores, and then put through the machine learning algorithm.

API key secret: lQF726OQxRqvFoGfUFEBgPSrV4vH8phzRhrmMt2WbHfuVfX4c4

API Key: HM2L20WSoeiDOGuBpWMBhO35Y

Access token: 1604068859706855424-yHTZzJEtlJJbCYiv21RwfxMPmhTRYI

Access token secret: WMwVZ2V7I3sNEQqmISUVy3BrczkCMyfo6oLaDlO6VWRXO

Bearer token: AAAAAAAAAAAAAAAAAAAAABI%2BkgEAAAAAXWj%2B%2Bf6FxPZ9fu9kE1wu0NPZiQM%3DauGTgGLIg4RuJ0rj54Z5I5vVFFHOOm7b4tu21dopElGbR6GhFK

Start with twarc2, I have the keys and the app on twitter, then downloading python 3, and instal twarc using pip on command control.

Use twarc to authorize app to access my account made for this.

Graphical user interface, text

Description automatically generated

There you go.

# Steps

**Step 1 -** Collect Tweets

* Identify how I want that to be done, what requests to send to the API
* Identify how to format properly
* Where to find the json or csv file afterwards

**Step 2 –** Build Dataset

* Figure out what data is collected when tweets are collected
* Data pre-process by removing unnecessary columns, only keeping the ones that are on the Stanford Sentiment Corpus
* Set aside all the data that is on the sentiment corpus that is not strictly tweet content
* Vectorize tweets using TF-IDF
* Use Easy data augmentation techniques
* Figure out how to correctly store a dataset

**Step 3 –** build lexicon dictionary

* Figure out how to build lexicon dictionary
* Figure out which words I want to use
* Figure out how it all works
* Figure out how to assign a number to the tweets

**Step 4 –** Put dataset through lexicon

* Put all the tweets through the lexicon to assign them temporary sentiment assignment
* Go through the tweets afterwards to manually review the sentiment assigned

**Step 5 –** Put through the machine learning algorithms

* Try a couple different machine learning algorithms to determine which one is the most efficient
* Select the best one

**Step 6 –** Build GUI

* Design GUI and implement it
* Have it accept user input
* Figure out how to connect the frontend of the application to the backend
* Take user input of keys and API call and select their tweets to run through the trained machine learning algorithm

Twarc has plugins that can convert line oriented json to csv, which is what we will want.

To start with twarc, create application on API and attach it to project on Twitter Developer Portal.

Can use archive and start-time just like a regular search command and can end up with full archive of all tweets for first day of 2020

twarc2 searches --archive --start-time 2020-01-01 --end-time 2020-01-02 animals.txt animals.json

Maybe if I remove the json stuff it will collect every tweet from that day

Stream command: collect tweets as they happen.

**USE PY -M FOR ACCESS TO ANYTHING PYTHON**

Downloaded the csv plugin for twarc using py -m pip install twarc-csv command in visual studio code.

This is so that the data collected can be automatically saved to a csv.

**Use 3-legged Oath for authenticating other users?**

WERE USING TWEEPY NOW

Using streaming HTTP protocol which makes it so that there is only one connection and every time a match is found it goes through that connection instead of opening a new one each time.

Find how to stream tweets in real time through this [4]. By default what I will get back is the id of each tweet and the text of each tweet but to make it comparable to the Stanford sentiment corpus I need to add the date, the query, and the user.

the 1% random sample of public Tweets provided by the sampled stream endpoint can meet this need since it provides a small subset of data relative to the total amount of public Tweets. Additionally, the data is sent to you in real time as it happens, which will meet the requirement of the data being current.

Install pandas so that we can then collect data into csv files.

Create config file to save all tokens and stuff in it meaning that I can share my files without people seeing all the keys, keeps it safer [5].

# Libraries imported

Pandas, tweepy, datetime, logging.

Use stream class from tweepy to read tweets in real time from twitter

It is possible to use that class to collect tweets, however, there needs to be a filter or a keyword to collect the tweets, this isn’t what I’m looking for as I want tweets that are perfectly objective please.

Also possibly need to evaluate the fact that I will be collecting tweets on European times, could do two waves to capture the American time zones as well however this will not include the entire English-speaking world.

it is possible to track tweets without adding any filter but would have to be done using twitter’s firehose option, which is rarely given out, so instead of using that I will filter out tweets by using very common words such as ‘I’, ‘a’, ‘you’, ‘it’. the most common words in the English language. I will also filter by English language since we are looking at English language tweets [6]. Using the most used words on twitter [7].

Sort by keywords + language, keywords are an OR situation not an AND one.

Since I only have an essential account for twitter I have to use the V2 of the twitter api, making things a little bit different to what we could do with an elevated account. But it’s all good.

We have run into a problem, that is that while using the most commonly used words is very convenient, the tweets will not be fetched unless there is at least one non-stopword in the filters. This is fine because here is the solution: will try using ‘new’ as it is still in the list of most commonly used words and isn’t a stopword I think

# Different approach

We are now using client because it’s a lot easier, won’t accept more than one stopword I think so we’re putting I in there and hopefully that will work well, using happy and upset as search terms too and that will help hopefully get good tweets with sentiments in them

Okay so that’s sorted

Then we iterate over it 3 times to collect enough tweets

Okay so then we’re pre processing

Using NLTK to find synonyms for both random insertion and synonym replacement in EDA. Nltk wants us to cite the book if used I think, although it says if publishing stuff: Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O’Reilly Media Inc.

Using tf-idf to vectorize tweets, it’s a small amount of data on each tweet so there are alternate ways of vectorizing that some people prefer, however there is evidence that tf-idf is the best method even for tweets as they are considered all as one. Word2vec is a popular one but it doesn’t work as well. Can use bag-of-words approach but that was shown to not be the most effective in literature review.

Install sci-kit learn

Tf-idf is used I THINK after the text has been classified, it evaluates the most important features in a document while turning it to machine readable format and therefore can help the algorithm draw conclusions about what features are important in deciding which class a text belongs to [8].

Gonna do feature creation with 2-gramsso as to be able to capture things like do not and all that. We are going to do 2-gramfeatures to encompass negations but we could do more, could be interesting in future to see if more has more effect on anything.

ngram\_range(1,2) meaning that n grams considered will be unigrams and bigrams [9].

Using vader for lexicon classification because ‘**VADER (Valence Aware Dictionary and sEntiment Reasoner)** is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media’ [10].

Vader is good but not super accurate, this way of classification can only ever be relatively accurate because a machine will never reach the level of nuanced understanding that a team of people could. For example when classifying ‘Star Wars’ VADER classified wars as negative so missed the context being that Star Wars is a movie.

Problems:

There are some very short texts that cannot get accurately classified because once they are pre-processed there is basically nothing left of them.

Also there should be a better way to find positive and negative tweets than using the keywords like that because the datasets don’t end up being particularly accurate, especially since the tweets will be heavily skewed towards whatever the first words in the keyword lists are.

A screenshot of a computer

Description automatically generated

Okay so new new plan, I had not understood TFIDF and it’s not going to work by multiplying things together due to the fact that vader already pretty much only rates positive and negative words for logical reasons so it doesn’t actually matter which word in the tweet is important because vader will probably be ranking it at zero anyway. So TFIDF can still be used but will probably be used later as a way of helping identify which tweets are particularly bad/good.

Okay so when running that it works well but there are no 1s which is not good because we need more or less the same amount of data for all classes so that the ML algorithm can learn accurately.

Actually I had messed up my code, but there are still only 24 instances of number 1 classifications.

The algorithms used for multi-class classification can be categorized into the following categories primarily:

1. Binary Transformation
2. Native Multi-Class Classifier
3. Hierarchical Classification [11]

For future work/how to make it better: make the data pre-processing steps easily reversible in case things have to go back to what they were.

Removing apostrophes in dictionary of contractions, can work but can also create inaccuracies, like how do you know when ‘its’ is it is or is just its, and hell that could be shortened he’ll or literally hell

Superuser profile for database:

Username: admin

Email: [clemence.weiss1010@gmail.com](mailto:clemence.weiss1010@gmail.com)

Password: Password123

**WIREFRAMES**

# References

[1] <https://techcrunch.com/2018/10/30/twitters-doubling-of-character-count-from-140-to-280-had-little-impact-on-length-of-tweets/?guce_referrer=aHR0cHM6Ly93d3cuc3RhcnRwYWdlLmNvbS8&guce_referrer_sig=AQAAANdNcmjwbwX0KyhrqCCoj3ZQhEQf2GdKsqXkEPlUVeRn1zivikMJHBGPkQLigA93kKehD7eVNAWWc8_KskonpGhy4H3sbPPWe94Yh2-cux1ezFw7_cZEJcGwdFSJP2n7hUiHSG0QV_tD71b6hwh23lxGXuTTPmAohOvLvxa5agzC&guccounter=2>

[2] <https://www.nature.com/articles/s41599-019-0280-3>

[3] <https://www-cs-faculty.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf>

[4] <https://developer.twitter.com/en/docs/tutorials/stream-tweets-in-real-time>

[5] <https://www.youtube.com/watch?v=Lu1nskBkPJU>

[6] <https://stackoverflow.com/questions/26890605/filter-twitter-feeds-only-by-language>

[7] <https://techland.time.com/2009/06/08/the-500-most-frequently-used-words-on-twitter/>

[8] <https://monkeylearn.com/sentiment-analysis/>

[9] <https://towardsdatascience.com/leveraging-n-grams-to-extract-context-from-text-bdc576b47049>

[10] <https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/>

[11] <https://www.projectpro.io/article/multi-class-classification-python-example/547#mcetoc_1fpjsn4g8b>

# Bibliography

<https://towardsdatascience.com/an-extensive-guide-to-collecting-tweets-from-twitter-api-v2-for-academic-research-using-python-3-518fcb71df2a>

<https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a>

<https://thesai.org/Downloads/Volume12No7/Paper_30-LSTM_VADER_and_TF_IDF_based_Hybrid_Sentiment.pdf>

<https://link.springer.com/content/pdf/10.1007/978-3-319-69900-4_48.pdf?pdf=inline%20link>

<https://er.ucu.edu.ua/bitstream/handle/1/2042/Babenko_Determining%20Sentiment%20and%20Important.pdf?sequence=1&isAllowed=y>

<https://www.diva-portal.org/smash/get/diva2:811021/fulltext01.pdf>

<https://link.springer.com/content/pdf/10.1007/978-3-319-69900-4_48.pdf?pdf=inline%20link>

<https://www.holisticseo.digital/python-seo/nltk/wordnet>

<https://iq.opengenus.org/naive-bayes-on-tf-idf-vectorized-matrix/>

<https://budibase.com/blog/web-application-development/>

<https://openclassrooms.com/en/courses/6967196-create-a-web-application-with-django/7349237-capture-user-input-with-django-forms>

<https://docs.djangoproject.com/en/4.1/intro/tutorial02/>

<https://www.deploymachinelearning.com/django-models/> - for the ml deployment