

School of Computing

**Honours Report**

***Application of XML tagging to the retrieval of sunbeams from cucumbers***

**Clemence Weiss 2023**

This report is submitted as part of the requirements for the degree of

**BSc (Hons) in Cyber Security**

at Robert Gordon University, Aberdeen, Scotland

**I confirm that the work contained in this Honours project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.**

Signed ..Clemence Weiss................................................ Date....................................

Table of Contents

[Abstract 3](#_Toc127023731)

[Literature Review 3](#_Toc127023732)

[Introduction 3](#_Toc127023733)

[Machine Learning and Sentiment Analysis 3](#_Toc127023734)

[Datasets 3](#_Toc127023735)

[Data Pre-Processing 4](#_Toc127023736)

[Data Augmentation 5](#_Toc127023737)

[Algorithms 6](#_Toc127023738)

[Lexicon based algorithms 6](#_Toc127023739)

[Machine learning Algorithms 6](#_Toc127023740)

[Hybrid Methods 7](#_Toc127023741)

[Conclusion 7](#_Toc127023742)

[Requirements Analysis 9](#_Toc127023743)

[Functional Requirements 9](#_Toc127023744)

[Non-Functional Requirements 10](#_Toc127023745)

[Steps 12](#_Toc127023746)

[Different approach 14](#_Toc127023747)

[Testing – Model Evaluation 18](#_Toc127023748)

[“Class weights 19](#_Toc127023749)

[References 26](#_Toc127023750)

[Bibliography 27](#_Toc127023751)

# Abstract

# Literature Review

## Introduction

As social media use continues to grow, cyber-bullying and hate speech have become more and more prevalent on social media and hate speech is now one of the most common methods for spreading harmful rhetoric and threatening peace (UN, 2021). This is a problem as hate speech has been proven to cause real life harm to people targeted, and to incite physical violence against the groups it is directed at (UN, 2021). It is therefore important to be able to detect this kind of speech to prevent the spread of harmful messages as early as possible.

Sentiment analysis is used to identify the emotional tone behind a piece of text, generally to gain an understanding of the opinions and attitudes people have towards a particular topic, such as a new product from a business. However a less researched area of sentiment analysis can be using detection on individual reactions on social media such as using it to detect potential cyber bullying or hate speech by identifying very negative sentiments in tweets (Ciaburro et al., 2022, Wei, Zou 2019).

Twitter has a very large user base and is often used to express opinions which is very useful for sentiment analysis. The twitter API is also accessible to use to create datasets (Zulfadzli, Haliyana, 2019).

This review will first discuss methods to use to construct a helpful dataset to train and test the classifier on, before discussing different machine learning models and feature classification methods that can be used for sentiment analysis.

## Machine Learning and Sentiment Analysis

While cyberbullying and harmful speech is growing on social media (Kim et al., 2021), there are few accurate and appropriate methods to flag these kinds of speech down. While sentiment analysis is extensively used for opinion mining, it is less frequently used for language detection, however it can be useful for this. Instead of sorting tweets into positive/negative classes, tweets can be sorted on a scale, and very negative tweets are flagged as potentially containing harmful content.

To do this we must first build a dataset with tweets to use, before using an algorithm to train and test on this dataset.

## Datasets

When working with machine learning algorithms, datasets are used to train the machine and to test the models used. Even though Twitter is one of the most common places to collect data, there aren’t many public datasets available. A lot of the datasets are small or

about specific topics such as healthcare or gas prices (Duong, Truong, 2019). Small datasets produce less accurate results since there is less data for the machine to train on and learn from (Zulfadzli, Haliyana, 2019). The largest twitter dataset publicly available is the Stanford Twitter sentiment corpus that has 1.6 million tweets (Saif et al., 2013), however they were labelled as positive or negative automatically instead of manually so are not accurately classified.

### Data Pre-Processing

Twitter is a social media that has a large amount of noise in its data that a machine will have a difficult time sorting through, so it is important to pre-process the data.

There are three different types of text formats in Natural Language Processing: structured sentiments, semi-structured, and unstructured. Tweets are unstructured sentiments which are the most difficult to work with (Duong, Nguyen, 2021). Duong, Nguyen (2021) applied the most common pre-processing techniques to four different datasets and observed that the F-score of every dataset improved afterwards. However they did not analyse which individual techniques were most effective.

URL removal is a common pre-processing technique as URLs are bulky and often do not add insight into sentiment. While Singh & Kumari (2016) argue that removing URLs is not necessary as it does not improve the later performance of machine learning algorithms, Jianqiang, Xiaolin (2017) show that while removing them might not improve performance it also does not decrease performance the way an important feature would, meaning removing URLs is beneficial as it will improve computational performance and reduce cost. This same method was used to prove that stopwords should be removed as well (Saif et al., 2014).

Lin & He (2009) argue that removing numbers reduces classification effectiveness, however more recent papers show that numbers are mostly irrelevant and do not help with classification, and they can be removed if any emoticons have been turned into words beforehand (Jianqiang, Xiaolin, 2017).

(Duong, Nguyen, 2021, Duong, Truong 2019) observed that replacing negations with a word’s antonym is not an effective technique, and while Duong, Truong (2019) analysed this effect from the point of view of news classification they used identical methods to tweet analysis and therefore their conclusions can be applied to tweet pre-processing.

Singh, Kumari (2016) removes all repeating letters, turning ‘coooool’ into ‘cool’, while Jianqiang, Xiaolin (2017) attempts to maintain the sentiment by conserving one extra letter, turning ‘coooool’ into ‘coool’, however this method had different results on all the datasets it was tried on so is not a conclusively effective pre-processing method.

Duong, Nguyen, (2021) tested different techniques and found that replacing emoticons with words, removing punctuation, stopwords, and numbers, and elongated word handling are effective pre-processing techniques. Their research is done on a Vietnamese dataset which might differ from an English dataset; however their findings are largely supported by

(Jianqiang, Xiaolin 2017, Effrosynidis et al., 2017, Duong, Truong 2019) who all tested on English datasets.

Effrosynidis et al., (2017) created a thorough literature review that found that stemming, replacement of repetitions of punctuation, and removing numbers all had positive effects on the models used. The pre-processing techniques that they observed having negative effects on the model were: removing punctuation, turning all capitalized words into lowercase, replacing slang, replacing negations with antonyms, and spelling corrections.

The literature finds that the pre-processing techniques considered most effective are removing pre-defined stopwords, expanding acronyms, replacing emoticons with text equivalents, removing URLs, removing numbers, and removing repeat letters.

While data pre-processing is overwhelmingly seen as a very important step in NLP, Saif et al., (2014) find that pre-processing leads to a significant reduction of vocabulary size, 62% in their tests. This is a non-negligeable loss of vocabulary; however this can potentially be helped by data augmentation techniques that will insert new vocabulary into the data.

### Data Augmentation

Data augmentation is the process of artificially adding content to a dataset based on the already existing collected tweets. While Duong, Nguyen (2021) state that data augmentation is more accurate on larger datasets as there is more vocabulary to use, (Wei, Zou, 2019, Duong, Truong, 2019) support the fact that the point of data augmentation is to use it on smaller datasets and is particularly helpful on unbalanced datasets (Abonizio et al. 2022). Twitter datasets can be unbalanced, Gaind, et al. (2019) states that most tweets are positive, however they used a small sample size and only collected tweets in India.

One of the most frequently used data augmentation methods is the Easy Data Augmentation (EDA) method. This method consists of four techniques that can be applied individually or together to a dataset: random insertion, synonym replacement, random swap, and random deletion.

(Wei, Zou, 2019, Duong, Nguyen, 2021) use this method in their research to analyse the effect and note that using EDA improves performance. Wei & Zou (2019) also noted that when using 50% of the available data in the dataset paired with EDA provided the same performance as when using 100% of the data without DA, meaning that EDA is very effective in augmenting data. While Wei & Zou (2019) did not look specifically at sentiment analysis they did text classification that can be applied to sentiment analysis. However, since longer sentences can absorb more changes without losing their meaning, there is a possibility this would not be as effective on tweets where the text is very short.

Back translation is another popular data augmentation method, where a tweet is translated into another language and then back again to create a different sentence with the same meaning. Yu et al. (2018) showed this method was effective, however it uses more computation effort for close to the same improvement as using EDA.

Type swap can also be used, this method replaces words in the text by other words of the same type (Raiman, Miller, 2017). Yu et al. (2018) argue that text samples will it be diverse

enough as the sentence structures do not change with this method, but it improves the accuracy.

## Algorithms

There are two approaches to classifying tweets: lexicon-based or machine learning.

### Lexicon based algorithms

Lexicon based approaches do not use datasets, they use lexicon dictionaries to count the number of positive and negative words in a text and base the classification on it. This makes them a very easy to implement and cost and energy efficient solution. However Hamdan et al. (2015) commented that lexicon-based approaches weren’t optimal since the performance depends on how good the dictionary is, and that it cannot classify nuances in languages accurately such as sarcasm and negation. Their literature review was conducted in 2015, however Zulfadzli & Haliyana (2019) also support this analysis. Lexicon based approaches are particularly unsuitable on their own for twitter sentiment analysis since twitter has a large amount of slang that will not be able to be captured by the dictionary unless it is kept up to date regularly.

Venkateswarlu et al., (2019) classify reviews and argue that lexicon-based approaches are better since machine learning classifiers are trained on limited data and may struggle to classify outside of the dataset that it was trained on, however previous research has established that a larger dataset can minimize that issue, and lexicon-based classifiers are also trained on limited data due to the limited capacity of a lexicon dictionary.

Lexicon approaches can still be useful for hybrid classification methods, Mody et al. (2018) try to identify tweets using lexicon-based approaches to classify the tweets based on emoticons present, and then using machine learning methods on these pre-classified tweets. This method gave good results and shows that using hybrid methods can be effective for tweet sentiment analysis.

### Machine learning Algorithms

Machine learning is different from lexicon approaches as it uses datasets to identify the sentiment of the tweets. Tweets must first be turned into a machine-readable format by turning them into numerical vectors called vectorization.

#### Vectorization

The two main vectorization models are TF-IDF and Bag of Words. While Bag of Words simply counts the amount of time a word is present and adds this to a vector, TF-IDF also considers how often the word is used in other tweets to give a weight to each word. Abubakar et al., (2022) finds that TF-IDF perform better than the Bag of Words approach, however they performed this study on book reviews which might differ from tweets as reviews will usually be much longer. Naf’an et al. (2019) used TF-IDF to flag potential cyber bullying tweets using a strong methodology and found that TF-IDF was an effective vectorizing method, however they did not compare it to Bag-of-Words. TF-IDF is good for more nuanced rankings of sentiments that have more classes than just ‘positive’, ‘negative’, and ‘neutral’, as Naf’an et al. (2019) does.

TF-IDF generally performs much better than Bag of Words vectorization, Rakhmanov (2020) used a 5-class classification and several different machine learning algorithms on a very large dataset and TF-IDF performed better on all of them. While this study was focused on student comments rather than tweets the sentiment analysis can be applied to tweets, however there might be differences again concerning the length of the text used.

Ismail et al. (2016) looked at whether Bag of Words or TF-IDF was better for twitter sentiment analysis, and they found an inconclusive result. However they performed these tests on a small dataset of only 369 tweets and using data from 2009, since tweets are now longer than they were back then this might not give accurate results for our purpose.

#### Machine Learning Approach

There are several machine learning approaches that come up the most often in the literature: Naïve Bayes, Logistic Regression, Support Vector Machine, and Maximum Entropy. Literature disagrees on which one is best for classification tasks with many papers finding different results. Zulfadzli & Haliyana (2019) found that SVM and Naïve Bayes, which are the most common models for sentiment analysis, perform very similarly to each other. Naïve Bayes is more successful on well-formed data which would indicate that it is not ideal for tweets that often have slang and poor spelling or structure, however this could be improved with good pre-processing.

When comparing SVM and Naïve Bayes on airline reviews, SVM performs much better than Naïve Bayes, Rahat et al. (2019) test this on tweets and used good pre-processing methods, however the datasets were mostly negative tweets which could skew the results. Naïve Bayes and SVM both have good precision among other machine learning algorithms (Alasaeedi, Khan, 2019), however they recommend hybrid classification as it seems to have the best performance.

### Hybrid Methods

While simple machine learning models are not proven to be necessarily more efficient than others, literature is clear that using hybrid approaches with both machine learning algorithm and a lexicon algorithm produces the best results. This is shown by Alasaeedi & Khan (2019) and Hamdan et al. (2015) who test with and without a hybrid approach and consistently find that hybrid approaches perform best.

While Dhaoui et al. (2017) did not find that hybrid methods were necessarily more efficient, they used a small Facebook dataset and used two different algorithms to classify positive and negative comments separately, making it difficult to reach an accurate conclusion about their performance.

## Conclusion

Research has been done to classify tweets into positive or negative sentiments, however not much research has been made into using this method to detect tweets based on a sliding scale of positivity/negativity instead of a binary approach, even though this could be helpful for cyber bullying detection among others. There are also few large datasets to use for sentiment analysis that can be used to produce accurate results. While this project will be

creating a dataset to test sentiment analysis algorithms on, it should also be able to interact with another public dataset to be able to compare results and ensure the classification method does work.

# Requirements Analysis

The requirements analysis are organised following the MoSCoW method. This method helps organize requirements and priorities of a project, by sorting requirements into Must, Could, and Shoulds. Must-haves are mandatory needs, the classifier will not work without them, should-haves are needs that are important but not vital, and could-haves are requirements that would be nice to have but are subject to being added in only if there is enough time.

## Functional Requirements

* Must collect tweets from twitter
  + Must use the twitter API to collect tweets
  + Must collect at least 300 to have a large enough dataset, the data will then be augmented to make the dataset larger
  + Must put all tweets into one csv
  + Must remove all unnecessary columns in the csv such as the dates, usernames, and tweet metadata, leaving only the raw text of the tweets.
  + Should use community libraries to help collect the tweets.
    - Could use twarc2, a community library that collects tweets based on the twitter API.
* Must use tweets to create a dataset.
  + Must pre-process tweets by removing URLS, replacing emoticons to word equivalents, removing numbers, punctuation, repeat letters and pre-defined stopwords, and expanding acronyms.
  + Must vectorize the tweets using the Term Frequency-Inverse Document Frequency method to assign a weight to the words in a tweet.
  + Must anonymize the tweets.
  + Should split the dataset 80/20 for training and testing so as to train the classifier on enough data to get better accuracy.
  + Should use Easy Data Augmentation techniques to augment the number of tweets collected.
    - Should use random insertion, synonym replacement, random swap, and random deletion to augment the data.
    - Should augment the data to end up with a dataset of at least 800 tweets.
* Must classify the sentiment in each tweet on a scale from positive to negative.
  + Sentiments in tweets must be recorded on a scale from 1 to 5.
  + Must use a hybrid algorithm, one that uses lexicon and machine learning algorithm methods.
  + The two algorithms used must be lexicon and Naïve-Bayes based.
  + Should automatically flag tweets with a score of 4 or higher.
  + Should be able to interact with datasets that are publicly available.
  + Could identify if a negative tweet is directed at a person using Part of Speech tagging, indicating a higher likelihood of cyber-bullying or harassment.
  + Could offer two different types of hybrid algorithms, one with a hybrid lexicon-Support Vector Machine algorithm and one with a hybrid lexicon- Naïve Bayes algorithm
* Should have an accuracy equal to or over 70%
* Should have an F-score equal to or over 0.7

## Non-Functional Requirements

* Must use python for the algorithm as it is simple and flexible and has a lot of useful libraries for machine learning projects
* Must respect GDPR guidelines for privacy and data collection by keeping all data collected anonymous
* Must document the progress of the project to have explainable project results
* Must use Git to keep track of version history of the project
* Must be tested by using the accuracy measure and the F-score as accuracy alone will not provide indicative results
* Must be stable, must produce the same results when tested multiple times on the same data
* Should be explainable, an algorithm’s decision should have a clear reason behind it
* Should use the twitter API to collect tweets, this is because the twitter API is available and easier to use, it also scrapes tweets more efficiently than other methods
* Should be scalable, usable on twitter outside of the datasets used to train and test on
* Could use the sci-kit python library when creating the dataset
* Could use cross validation for testing

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

The subject finding thing is not working very well because of the removal of stopwords, i can find a way to get clean tweets without removing the stopwords but the literature review shows that removing stopwords doesn’t improve classification performance, just improves the computational load. Therefore in our situation it is easier to just leave the stopwords in.

**WIREFRAMES**

Problem: the dataset appears to be repeating the same few tweets over and over again, this is because we’re running the search recent tweets over and over again and there isn’t enough time between the queries for there to be more ‘recent tweets’. To fix this

We will be using F-score, the confusion matrix, and precision and recall to evaluate the classification algorithm as those are the best ways to evaluate an unbalanced dataset like the one we are using [12].

“Precision is the number of true positives divided by the number of total positive predictions. In other words, precision finds out what fraction of predicted positives is actually positive.” [13] Precision formula: (True Positive)/(True Positive + False Positive).

“The recall is true positive divided by the true positive and false negative. In other words, recall measures the model’s ability to predict the positives.” [13]. Recall formula: (True Positive)/(True Positive + False Negative).

* “**Precision** — among all the positive predictions, count how many of them are really positive.
* **Recall** — among all the real positive cases, count how many of them are predicted positive.
* **Accuracy** — among all the cases, count how many of them have been predicted correctly.” [15]

# Testing – Model Evaluation

On my dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | Categories | Lemmatization | Weighting | Precision | Recall | F-score |
| 1 | 5 | No | No | 0.7626059291615871 | 0.5667185861427105 | 0.5793190406764861 |
| 2 | 2 | No | No | 0.8663333129247576 | 0.839261776503462 | 0.8418861500378643 |
| 3 | 5 | No | Yes | 0.7246101596406151 | 0.6696557754248114 | 0.6345077927445895 |
| 4 | 2 | No | Yes | 0.8663333129247576 | 0.839261776503462 | 0.8418861500378643 |
| 5 | 5 | Yes | No | 0.7724712521472578 | 0.5359956183744464 | 0.5596684670745148 |
| 6 | 2 | Yes | No | 0.8663333129247576 | 0.839261776503462 | 0.8418861500378643 |
| 7 | 5 | Yes | Yes | 0.7266700576483489 | 0.660751846491658 | 0.6212992203333438 |
| 8 | 2 | Yes | Yes | 0.8634002919024895 | 0.8525961823549488 | 0.8480923086331483 |

On the Stanford dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | Weighted | Precision | Recall | F-1 Score |
| 1 | No | 0.7650593153764582 | 0.7628786424439681 | 0.762392519131747 |
| 2 | Yes | 0.7650593386712401 | 0.7628785980107688 | 0.7623925082721987 |

Weighted and unweighted calculations will have the same confusion matrices and precision + recall + f1 scores as the only thing different is how the final results are calculated.

|  |  |  |
| --- | --- | --- |
| Test No. | Confusion Matrix | Visual Results |
| 1 + 3 |  |  |
| 2 + 4 |  |  |
| 5 + 7 |  |  |
| 7 |  |  |
| 8 |  |  |

Stanford dataset

|  |  |  |  |
| --- | --- | --- | --- |
| Test No. | | Confusion Matrix | Visual Results |
| 1+2 |  | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Weighted | Precision | Recall | F-Score |
| Stanford Dataset | Yes | 0.7650593386712401 | 0.7628785980107688 | 0.7623925082721987 |
| Stanford Dataset | No | 0.7650593153764582 | 0.7628786424439681 | 0.762392519131747 |
| My Manually Classified Dataset | No | 0.8106120268620269 | 0.7938690476190476 | 0.7862015368725439 |
| My Manually Classified Dataset | Yes | 0.8322485684813271 | 0.8190804597701151 | 0.8099738907034595 |

**Web App Testing**

The noted benefits of using SUS include that it:

* Is a very easy scale to administer to participants
* Can be used on small sample sizes with reliable results
* Is valid – it can effectively differentiate between usable and unusable systems [18]

Steps and questions:

* I think that I would like to use this system frequently.
* I found the system unnecessarily complex.
* I thought the system was easy to use.
* I think that I would need the support of a technical person to be able to use this system.
* I found the various functions in this system were well integrated.
* I thought there was too much inconsistency in this system.
* I would imagine that most people would learn to use this system very quickly.
* I found the system very cumbersome to use.
* I felt very confident using the system.
* I needed to learn a lot of things before I could get going with this system. [18]

**TESTS:**

**Tests for testing the web application: using SUS evaluation:**

* Design a test which gets users to try the application and encompasses the three criteria: efficiency, time, overall satisfaction with the tool

Get users to look for 50 tweets using specific keywords - SEQ

Get them to display the negative tweets - SEQ

and then display the personal negative tweets -SEQ

Get them to look at tweets classified by the tool, get them first to classify if they think these tweets are positive or negative, then display what the tool says

Then get them to complete a questionnaire about the SUS stuff, then a questionnaire bout how easy they thought the task was to complete.

Then SUS Test.

So with the current standards on vader classification, with it being perfectly equal, the dataset is now at this many of each category:

“Use weighted macro-averaging score in case of class imbalances (different number of instances related to different class labels). The weighted macro-average is calculated by weighting the score of each class label by the number of true instances when calculating the average.” [16]

## “Class weights

The weights from the class\_weight parameter are used to **train the classifier**. They **are not used in the calculation of any of the metrics you are using**: with different class weights, the numbers will be different simply because the classifier is different.

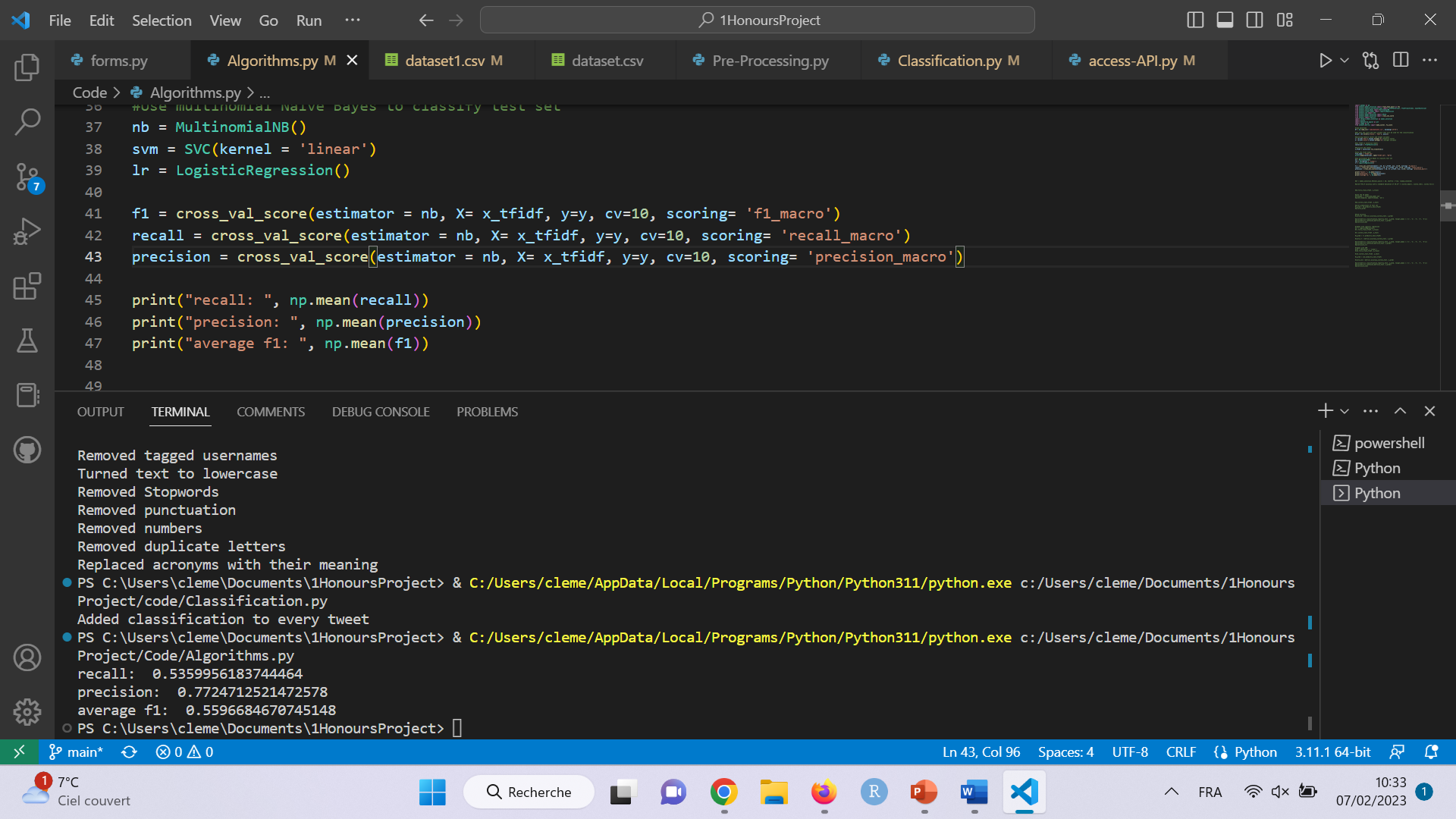
Basically in every scikit-learn classifier, the class weights are used to tell your model how important a class is. That means that during the training, the classifier will make extra efforts to classify properly the classes with high weights.  
How they do that is algorithm-specific. If you want details about how it works for SVC and the doc does not make sense to you, feel free to mention it.” [17]

Auth token for github: ghp\_HmZQxIx3j11g3CdV2kcrf6payZWmkp1Od7VG

A screenshot of a computer

Description automatically generated

This is with stricter measures for vader scores, 1 and 5 only worth 0,2 points. And 3 was at 0,1. Recall: 0,49979, precision: 0,78346, f1: 0,5426



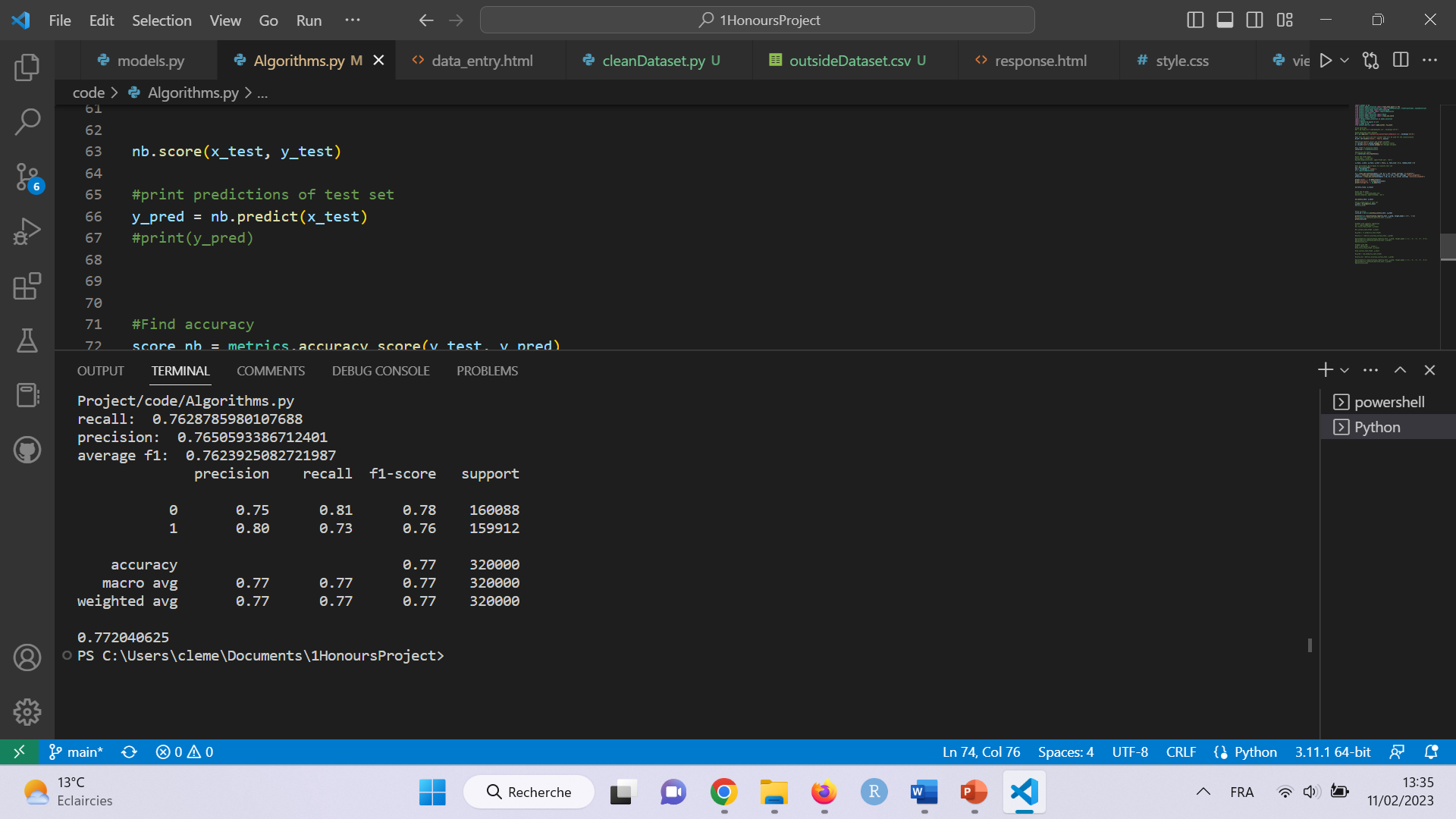
This is with equal repartition of scores with vader: precision: 0,77247, average f1: 0,559668, recall: 0,535999

A screenshot of a computer

Description automatically generatedusing only 2 categories.

A screenshot of a computer

Description automatically generatedusing weighted evaluation



Results after the stanford dataset spread on 2 categories.

Due to the removal of free access to the twitter API the web application was not able to be appropriately user tested, this maens that there is likely faults and errors that could make it difficult to navigate the layout of the web application or to use it prpoerly. However special care was taken into designing the web app with consideration for the user, such as adding validation to fields so as not to take vales that would not be accepted as proper values by the API. Also good contrast and stuff for disabilities.

Things to do:

* Test with another dataset that only has two columns and that are already classified
* Try to take a small subset of my dataset and classify it myself and then test that
* Make sus test
* Get it working on pythoneverywhere
* Create a scenario for people to test it on
* Have a way to give them tweets first and then have them classify them and then ask whether or not they agree once they are actually classified

# 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>

[12] <https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/>

[13] <https://regenerativetoday.com/learn-precision-recall-and-f1-score-of-multiclass-classification-in-depth/>

[14] <https://machinelearningmastery.com/k-fold-cross-validation/>

[15] <https://towardsdatascience.com/model-evaluation-in-scikit-learn-abce32ee4a99>

[16] <https://vitalflux.com/micro-average-macro-average-scoring-metrics-multi-class-classification-python/>

[17] <https://stackoverflow.com/questions/31421413/how-to-compute-precision-recall-accuracy-and-f1-score-for-the-multiclass-case>

[18] <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>

# 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

<https://stackoverflow.com/questions/61783499/remove-a-word-if-it-contains-a-specific-letter>

<https://subscription.packtpub.com/book/data/9781838987312/2/ch02lvl1sec16/extracting-subjects-and-objects-of-the-sentence>

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

<https://stackoverflow.com/questions/48606087/getting-values-of-queryset-in-django>

<https://www.machinelearningplus.com/nlp/lemmatization-examples-python/>