# Introduction

The purpose of this report is to complete a cursory overview of different sentiment analysis tools to see which one would likely work best for a client’s social media application to test the public’s sentiment towards Artifical Intelligence. Sentiment analysis is a technique that looks at written text by an individual and attempts to score it from a positive reaction through to a negative one.

Sentiment analysis can be quite helpful identifying where a consumer has had an issue with a product and thus allow a business to quickly respond hopefully remedying the situation. It also shows the larger audiences reading the responses that the company is out trying to make amends thus hopefully persuading them to engage with the organization. Sentiment analysis can als be used to gage how the overall public is responding not just to products, but politicians and other professionals placed the spotlight.

Sentiment analysis can also be a bit of a tricky topic as subjectivity of what constitutes negative and positive can arise. Much work has been done within both the data science and linguistic communities collectively to create dictionaries of words and their sentiment strengths. The various tools can either up play this work by providing a magnitude or downplay it by provide a meager positive or negative result.

That in itself is not highlight unreasonable. A positive or negative tag helps a business quickly identify those individuals with which they need to have further discussion, while the magnitude may help a politician or product development group gauge the reaction to a newly provided idea. The tools discussed within are free to use, but are they good enough to help a client decide if they are strong enough to be deployed.

# Analysis and Models

## About The Data

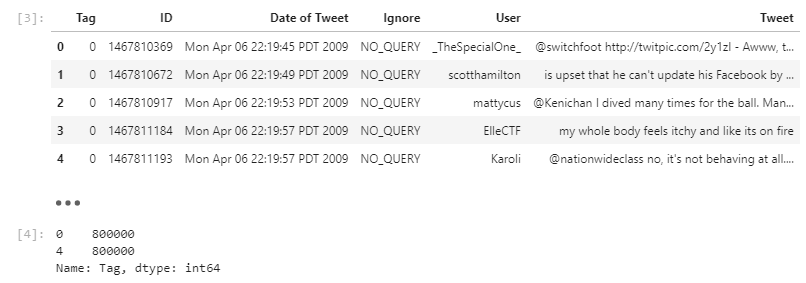
For this exercise a data set of various Tweets was selected from Kaggle. This data contained 1.6 million Tweets gathered from the spring and summer time in 2009. The Tweets have been tagged with 0 for negative sentiment and 4 for positive sentiment. The documentation describing the data set indicates that a tag of 2 should indicate a neutral tweet. Using Python’s Value Counts function, it was shown that there were know.

Along with the tag the data set contains the date of the tweet, the id of the tweet from the twitter database, a flag field which all values were the same, the user, and for this task most importantly the tweet from the user.

The preparation for the sentiment analysis was as follows. First the data was imported from the CSV file provide. Second the columns which were lacking a name were update per the documentation. Next the records were checked to see the tags and how many of each there were. As mentioned earlier, while the documentation for the set claims a tag of 2 there were none present in the data set.

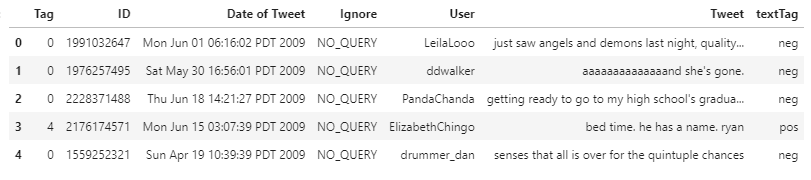
The data set was already balance with an even number of negative sentiment and positive sentiment records. Given the balance the relatively short text in a tweet, a decision was made to utilize all records for the training purpose.

Next using python’s text here package the tweets were set to all lower case letters. White space and diacritics were removed from the text. The models were tested both with and without the punctuation removed. As the 2nd model had a higher result with the punctuation left the decision was made not to adjust the punctuation. With twitter data this can be a little touchy. Hash tags a commonly used to point users towards certain topics and these can be used in other analysis including twitters famous trending functions. In addition all user names are seen with an @ sign in front of them and utilized in tweets to send notification to the selected users.



A snapshot of the cleaned text

As the final two steps the data was randomly sorted using python’s sample function and sampling all the records. The randome state argument was used to ensure the same results can be achieved by any party looking to reproduce the work. Lastly the tags translated to ‘pos’ and ‘neg’ by adding a new column to the data frame. This just helps make the comparison’s easier later on.



The data Sorted

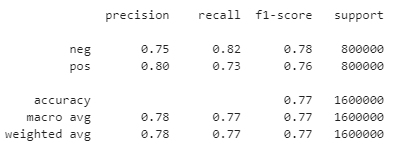
## The Models

### Model 1

For the first model a Naive Bayes classifier was trained on the text tags from the data set. Prior to initialing the training, the first step was to take the cleaned data from the prior section and convert the text reviews into Term Frequency Inverted Document Frequency matrix.

What this technique does is take each of the words in each review and score them by how frequently they show up in the review and penalizes them for showing up to frequently in other reviews. As mentioned in the exploration section, this would minimize the impact of the stop words on the model. For English, words like the, a, that, and so forth which provide structure to the sentence, but provide no real meaning, leaving words with strong emphasis and that happen less frequently with higher numbers for the model to learn.

Additionally for this train, the decision was made to use cross fold validation instead of the holdout method to train the model. 10 folds were used for this analysis. Break the data in 90% and 10% remaining. The 90% was used to train the model and 10% to test. This process then repeats with another break of 90% and 10% till all 10 folds have been used to test the varying models. The model’s average score accuracy is returned, for this model it was 77%. Precision and recall both around the average as well.



Confusion Matrix of Naive Bayes Model

This model learns from data has be classified previously and attempts to learn from the data set it is fed. It is limited in the ability to really grow beyond the classification tags that it is provided. As an example, the model cannot decide the magnitude of the sentiment if the tags are just positive and negative. However, the model can be easily retrained using other data or new data with these features.

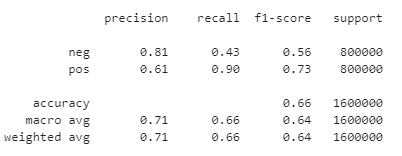
### Model 2

The second model chosen as a comparison is the Vader sentiment classifier. Unlike Naive Bayes this model is based on a dictionary of words that have been classified by their level of sentiment. Negative, Neutral, and Positive words ranging from 0 to 1. 0 being the weakest possible sentiment in the category to 1 being the strongest.

Per the documentation on Github, the three aforementioned scores are then aggregated to create a compound score for the sentence that expresses if the not only if the sentence is positive, negative, or neutral. It also describes how strongly the sentiment is expressed from 1 being the most positive to -1 being the most negative. The documentation also stresses that the range for neutral range greater than -.05 to less than .05 which subjectively is quite narrow.

The model itself works by pass the text through the model it returns the 4 scores in python as a dictionary. Once applied to the data frame, and the lambda function was used to extract the compound value to its own column in the data frame and the bands from the documentation provided on Github used to calculate the predicted label. Due to the fact there were no neutral tags in the main data set, the neutral rang was absorbed into the positive and negative. With positive ranging from 0 to 1 and negative ranging from -1 to less than 0.

While the first few records do not appear to be very promising from the output but since there are over 1 million records in the data set. Similar to the Naive Bayes model covered in the prior section, a confusion matrix will provide an overall reality of how the vader model has preformed. Not the compound value mentioned earlier in the second to last column shows not only the sentiment but how strong the vader model has assessed the sentiment.



Confusion Matrix for Vader Model

The model preforms with a 66% accuracy better than the sample seen above. However; since the original measurement was limited to just negative and positive the strength of this method is partially lost in the comparison.

This model being built on an underlying dictionary may also prove to have a few challenges in a social media environment as the rapid invent of new slang may lower the performance, however this model is better suited to answering the clients question about the publics’ AI perception.

## Conclution

Both the Naive Bayes and Vader model for reviewing the sentiment of social media texts on twitter have preformed well. Naive Bayes has out preformed the model by recognizing patters in the word frequency that allow it to recognize how the human tags for positive and negative sentiment were applied. This model does have some limitation in that if the task at hand for the client requires not only knowing positive or negative but some measure of magnitude this model does not provide such an outcome.

The Vader model preformed slight worse than the Naive Bayes Model. Being built on a dictionary of sentiment may limit its performance with less common words seen on social media. It also does not work by learning the patters but measure preset levels of sentiment of given words in a sentence. This can provide useful information when testing out new ideas and seeing how the reception of the idea is fairing. A positive ID with a lukewarm sentiment might not be worth the energy of pursuing. While a strongly negative sentiment may be a sign that an idea needs to be entirely rethought.

With the client’s goal of measuring the public sentiment towards AI, the recommendation would be to deploy the Vader model to do this. It decent accuracy coupled with the ability to capture the magnitude make it the perfect tool to not only measure the sentiment but with enough data be able to segment it to the public. The recommendation also comes with the recommendation for additional research into method to help tune the model, but at this deployment level it will provide a reliable measure of sentiment on real world sentiment.