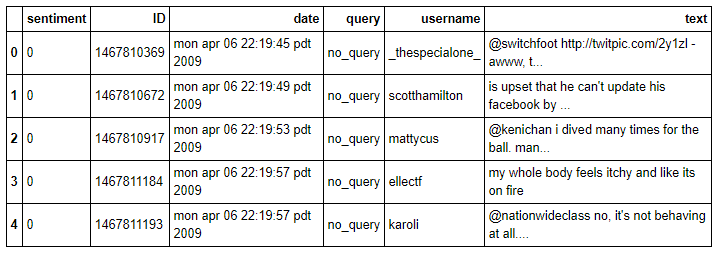
**Introduction**

In our project, we decided to try to work on a sentiment analysis method. Especially in today’s social-media focused world, automated sentiment analysis on blogs, tweets, and other types of social platform media becomes increasingly important for public figures, political campaigns, or a slew of other uses.

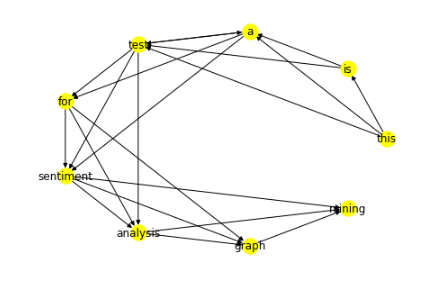
Although there are many existing methods to perform sentiment analysis, from simpler map-reduce methods to complex machine learning algorithms, one area which is not as explored is graph mining. Thus, we chose to attempt to implement sentiment analysis using graph mining. We used a dataset consisting of 1.6 million tweets, which were pre-classified into positive and negative tweets. (Although the dataset we sourced these tweets from claimed that the tweets were separated into three classes- positive, negative, and neutral, we did not find any tweets that were classified as neutral.) We constructed several different graph models that are representative of positive tweets, as well as another set of graph model representative of negative tweets. We also generated graph models of individual tweets in order to train a classifier, of which we used several, such as k-nearest-neighbors, support vector machine, ID3 decision tree algorithm, and also logistic regression. To train the classifier, we used several different methods to generate vectors representing each graph model of each tweet, which we will discuss in further detail later.

**Methodology**

 Our data came in the form of a CSV file, which was loaded into a Pandas data frame. We made sure each letter was lowercase, and ultimately our data, before any of the text was processed, looked like this:

However, attempting to run our project with 1.6 million tweets proved challenging, as we were limited by computing power. Some of the functions which we tried to run, such as building a graph model, took over two days and had not yet completed before we decided to try a different route. We used a keyword likely to garner both positive and negative tweets (“food”), and ended up with 8440 tweets, which is number that is much more easily digestible. Given that we have enough time at the end of project design, we could complete writing all the functions necessary to analyze data, select the best method of analysis, and apply to a larger dataset.

We split our data 80/20 into a training set as well as a testing set. The training set was further split 60/40 into a set to develop our positive and negative representative models, and also a set to train our data mining classifier, respectively. Finally, the set that was set aside to train our positive/negative model was split into positive and negative tweets, which was already classified in the given dataset.

 Before we generated any graphs, we pre-processed the text in each tweet, removing stop words to improve run time, as well as removing any characters that are not letters, words, or spaces. In order to represent a string of words in a graph format, we looked at each word in the string, as well as the words that appeared in its neighborhood, which we called the frame. If we set the frame equal to two, then our algorithm would direct an arrow from the first word to the second word, as well as the first word to the third word. The frame would then slide one word to the right, treating the second word as the current first word, and linking the next two words in a similar manner. If a node representing a word already exists, then our algorithm draws an arrow to and/or from the existing node, without creating a new one.

The diagram on the right shows an example of what the graph of the sentence “this is a test a test for sentiment analysis graph mining” with a frame of four looks like. Note that “a test” is written twice in the above sentence to demonstrate how the algorithm handles existing nodes.

Following, we define several different functions representative of similarity between graphs. First,