**Third Milestone**

**Abstract.**

In this milestone, we implemented a brute force algorithm to swap abbreviations in a tweet for their meaning. This algorithm has an efficiency of O(kn) where n is the size of the dataset of tweets and k is the size of the dataset of abbreviations. Split of responsibilities for this task was equal, which each teammate contributing to the final product.

**Dataset.**

The dataset utilized in this project is used from Kaggle, this dataset contains a random set of tweets from various topics and users collected at a sample size of 1.6 million tweets. As the dataset consists of various topics and users the algorithm can be utilized to the best of its potential and have reliable data to work with. Due to the dataset being large to this amount we could also push the limits of the algorithm and the code to test the efficiency and have a large enough dataset to plot the graph using the Big O notation the link for the dataset is <https://www.kaggle.com/datasets/kazanova/sentiment140?resource=download>.

In addition, we utilized another dataset which consisted of a list of abbreviations and their meanings which. The dataset is about 300 different meanings and abbreviations and can be easily updated. The link to the dataset is [Common Abbreviations and short forms with meanings | Kaggle](https://www.kaggle.com/datasets/ckapre51/common-abbreviations-and-short-forms-with-meanings).

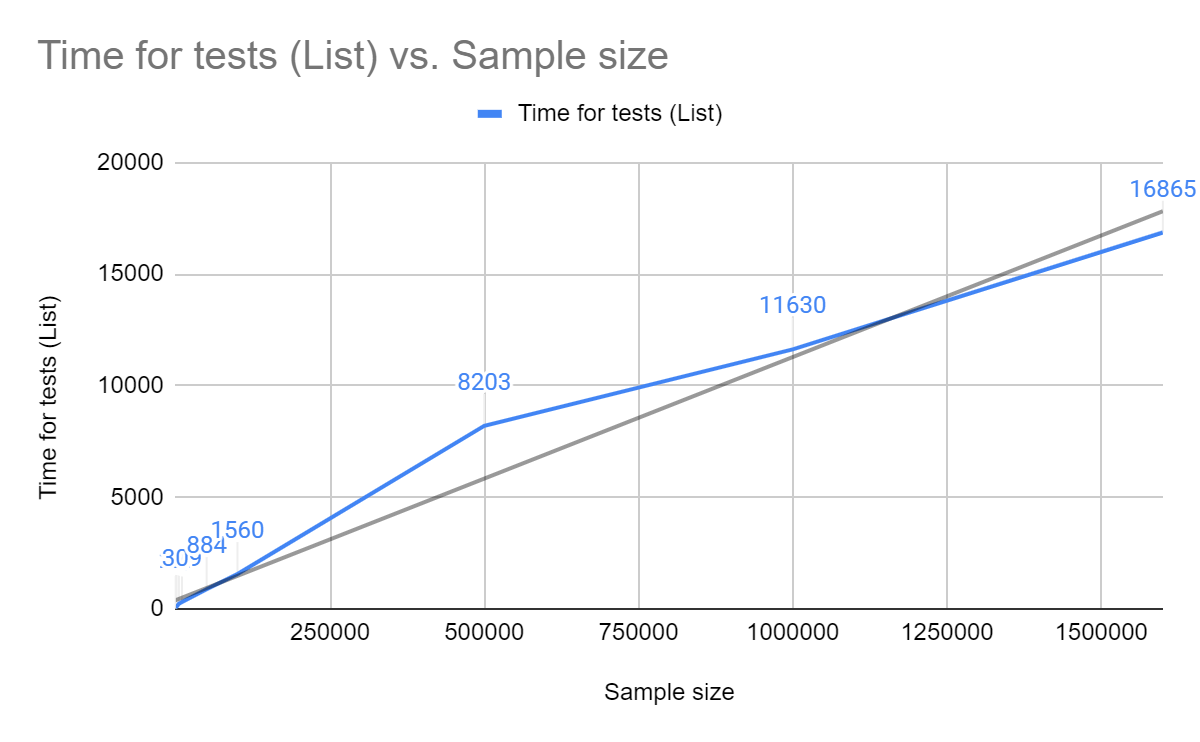
**Implementation.**

To change all abbreviations to their meaning equivalent, we first loaded 2 datasets into the program. For each dataset, we have also implemented a class to hold the data and obey the principles of object-oriented programming. Class "Tweet" has a field of type string that holds the text of the tweet, and class "Abbreviation" has 2 String fields that hold the word and its meaning. Both classes have appropriate constructors and getters and setters. Since data sets are CSV files, we used [OpenCSV](https://opencsv.sourceforge.net/) java library to read from them and write to a new CSV file later. From there, we looped through all the tweets, separating each one into a string of words. We used a delimiter to split the string and account for spaces and possible punctuation signs like commas. For each the splitted tweet, we called the function "swapWordsForMeaning" to change the abbreviation into meanings. This function loops through all the words in the tweet and for each one, it loops through the list of abbreviations and checks if any of them are equal. If they are, it swaps the word for its meaning. At the end, it returns an array of words where abbreviations are swapped. We then call the function "makeStringOutOfWords" on this returned array to get the array back into the shape of a string using StringBuilder. After that, we printed out a sample of 10 tweets to test the program and wrote all new tweets in the CSV file.

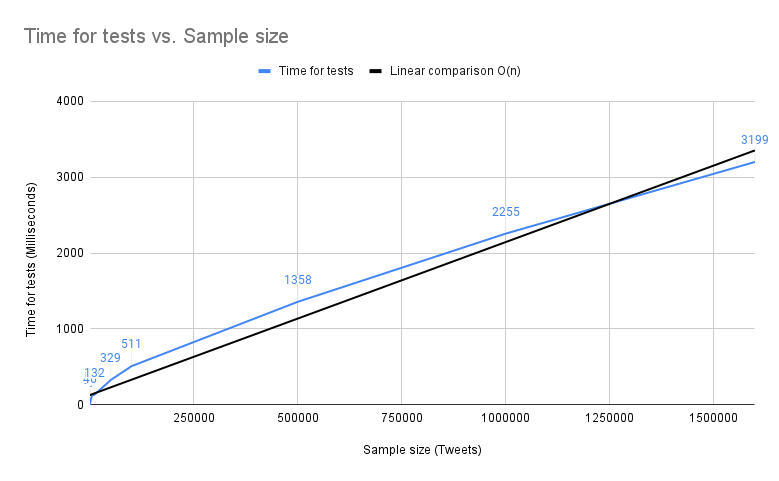
To test the time it takes to run our program for different input sizes, we used a system function to get time in milliseconds before and after running our algorithm and subtracted start time from finish time. We then manually changed how many tweets is "readTwFromCSV" function reading from the dataset and reported values for input sizes varying from 10 to 1.6 million and plotted the graph using an Excel sheet.

**Results.**

Our algorithm was determined to run in O(kn) time, where k is the length of the abbreviations list and n is the number of tweets analyzed. Since we used a brute force method, the result of our algorithm is approximately what we expected. The algorithm is designed to look through the entire length of the abbreviation list and the entire length of each tweet (up to 280 characters) therefore, this algorithm takes much longer to analyze the samples given. For example, at n = 500000, the time to execute this code takes nearly 8x longer than our original algorithm.



Picture 1: Plot for the new algorithm O(kn). Black line represents k\*n function.



Picture 2: Plot for the old algorithm O(n)

**Unexpected Cases/Difficulties.**

We had attempted to account for common abbreviations being part of words however, we encountered an issue with words that are the same as common abbreviations. For example, a tweet that might have read “I’m so happy” would change to “I’m significant other happy”. We have also run into issues with data set having abbreviations that are not very commonly used completely changing the sentence structure. For example, in the dataset letter "I" had meaning "eye" which is not commonly used, so we fixed that by cleaning up the dataset from such examples.

**Task Separation and Responsibilities.**

Andela focused on making an alternative algorithm and approach to handle the data, and Andela and Mackenzie helped implement the program. Arya helped with analyzing and collecting data. Everyone helped create and document for this milestone.