Project Outputs:

There are two folders in the project: 1) data and, 2) results

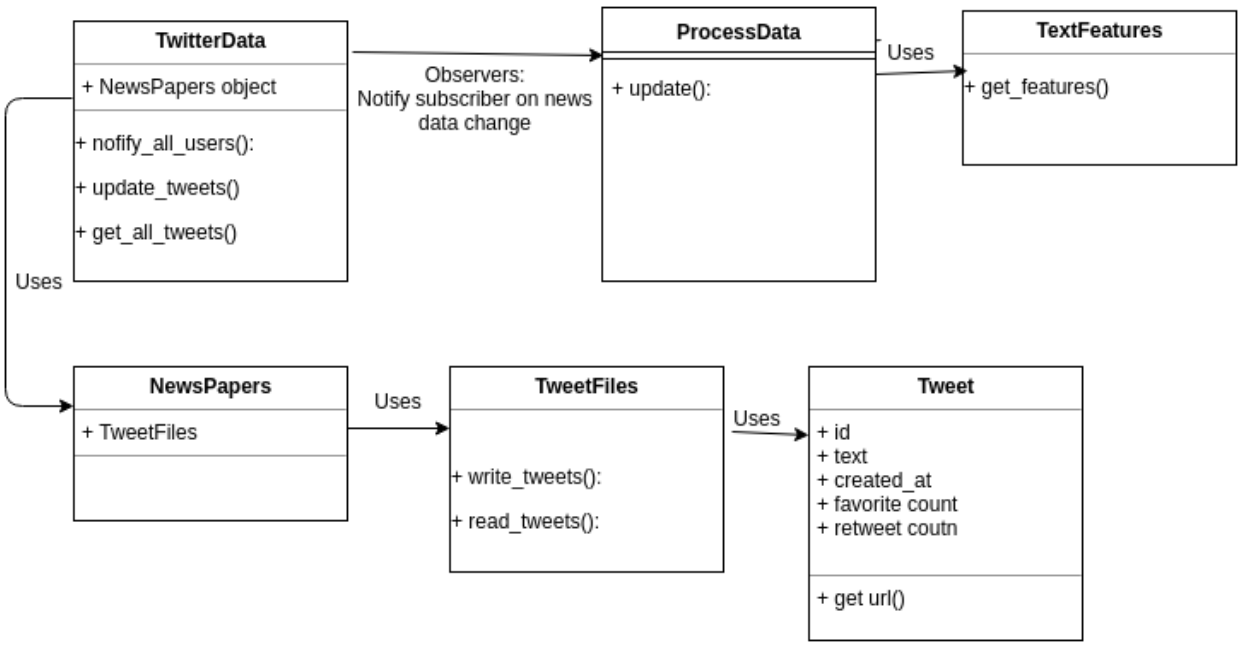
1) Data folder contains all the raw tweets from different news agencies. Each news agency has their own file which contains tweet with different features (likes, retweet, url, text). Each file is updated in 10 minutes interval and keeps the tweets which are recent. It deletes the tweets which are older than 120 minutes. For the first time, when app is executed, it fetches tweets which are posted within last 60 minutes.
2) Results folder contains three text files. One is clusters.txt which contains all the clusters of news in last 60 minutes (which can be set to any value by changing TwitterData.store_limit value). The second one is best_tweet.txt which contains current best tweet which is ranked based on number of favourite count and number of retweets count. The other file is send_user.txt files which contains the tweet which we want to send the users. If it is empty that means we do not want to send any tweets from last 60 minutes.

Project Design:

I tried to implement the project in modular fashion. I used **observer design pattern** so that when we get a update (after 10 minutes in our case), it will automatically inform the subscriber (ProcessData class is our subscriber in our case).

In our project we have two main classes. One is **TwitterData** which is the subject and other is **ProcessData** which is the subscriber. Whenever we call the `update_tweets` methods of the TwitterData class it automatically informs the ProcessData class which then updates data processing and write the results in the text files.

Fig: UML diagram



UML diagram: I didn't include all the connections between the classes for simplicity.

Main.py: It's the main file to execute.

Clustering Algorithm:
I used only nouns and verbs from each tweet to represent the tweet (gave up the stock words). Then created a feature dictionary using all the unique words in all the tweet texts. Here I used similarity score to find out whether a word is unique comparing with all the existing words in the dictionary. Then created feature for each tweet using the Bag of Words (BoW:if-idf) technique. Then used the hierarchical clustering algorithm to cluster tweets into different clusters.

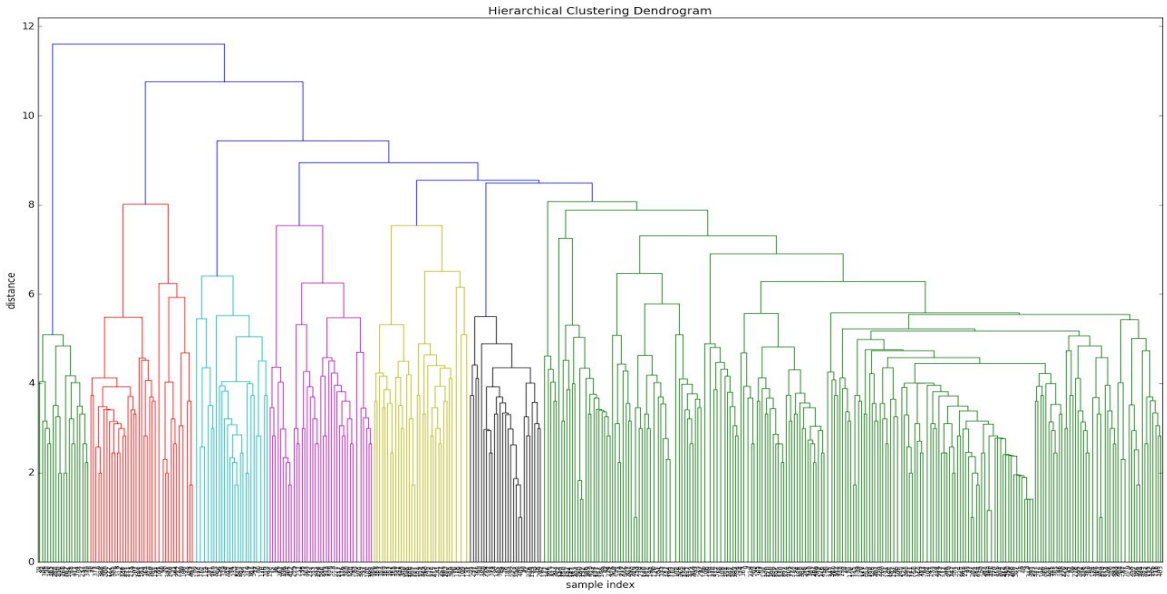


Fig:Dendrogram of the hierarchical clusters

Results:
Currently, my algorithm doesn't have a very good result. However, I believe if I do more tweaking with the parameters (like different threshold parameters) then the performance will improve.
Future Improvements:
Since we are using raw words so If a tweet has more words than the average tweet lenght then the distance to all other tweets become larger. To overcome this I need to use a semantic meaning of the sentences. Like if I can get some tags from a sentence like does this sentence is related to politics, social, sports, accident and so on. Then I can use these as features along

with original. To retrieve tags from sentences, I can use deep neural network to train from existing news website with their given category as a ground truth.