Name **Mugoya Dihfahsih** **S**tudent no. **2100702353** **Github link:** Link to GitHub Project Repository

Abstractions

An approach to complex problems in which the solution is built in layers. The structures needed to solve the problem (algorithms and data structures) are implemented using lower-level capabilities and given names that can then be used as if they were primitive facilities

The abstractions can be categorized into four levels that is; Bottom level, Second level, Third level and Top level depending on the program, some have one level others two or three while others have all levels

Bottom level; This is a level at which the selectors are being implemented; it is sometimes called the level of pairs. The procedures *cons*, *car* and *cdr* which have been used in the project extensively are already provided by the language. The actual implementation of pairs is hidden at this level

Second level; This is the level of data-science as data objects, at this level the constructors and selectors of cons, car and cdr have been used as procedures and it is at this level where wishful thinking starts because the program does not show how these procedures will be implemented or represented. The implementation of data-science programs at this level is hidden.

Third level; This is the level of service procedures to the data-science program, it is at this level where the procedures such as linear-model, list→sentiment, read-csv, qq-plot, hist and all procedures made available using *provide* keyword are defined. The implementation of these procedures are hidden from the main program or in the code block where they are used.

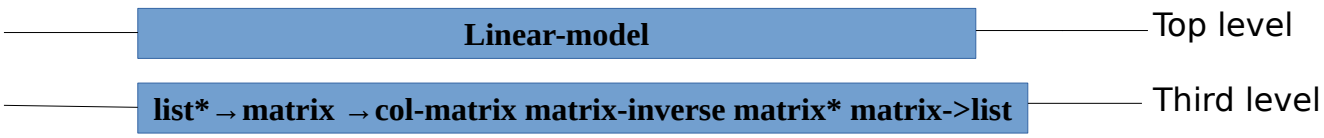
Top Level; Data-Science is now being implemented at a program level, the procedures defined in the program are now being used as if they were ordinary primitives of data.

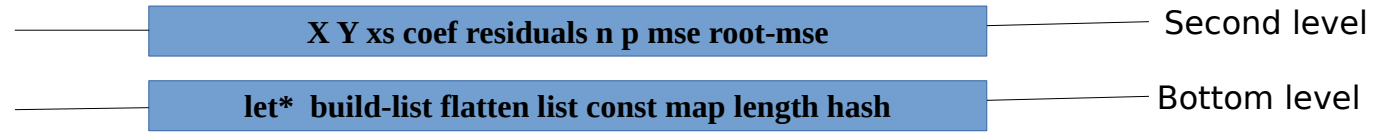
Each level has been designed to hide the implementation details from the higher-level procedures known as abstraction barriers.

QTN.1 Abstractions and procedure levels identified in the *data-science-master* project

The following are diagrams showing abstractions found in each of the identified *linear-model*, *hist*, *hist→sentiment*, *qq-plot** and *csv-reader abstractions*. I have clear identified them in the file called *abstractions_in_data_science_master.scm* in the End_of_semester_project folder. These are the abstraction and procedure levels in the data-science-master project, the lines on each end show that there are more abstraction implementation hidden at each level

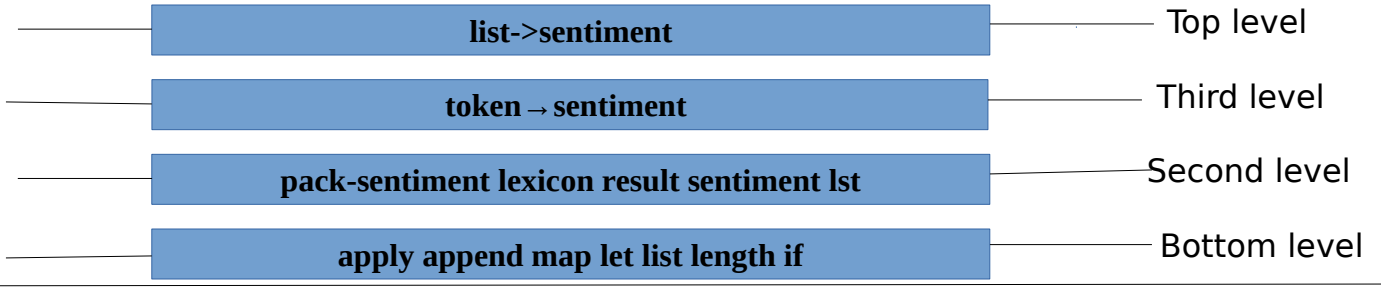
The Linear-model Abstractions



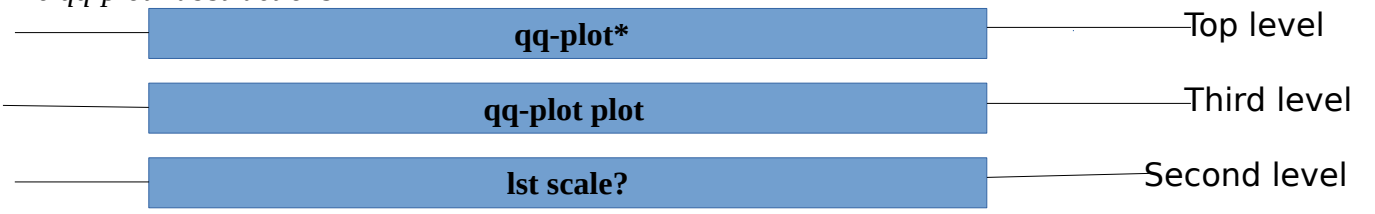


The histogram Abstractions

The list→ sentiment abstractions



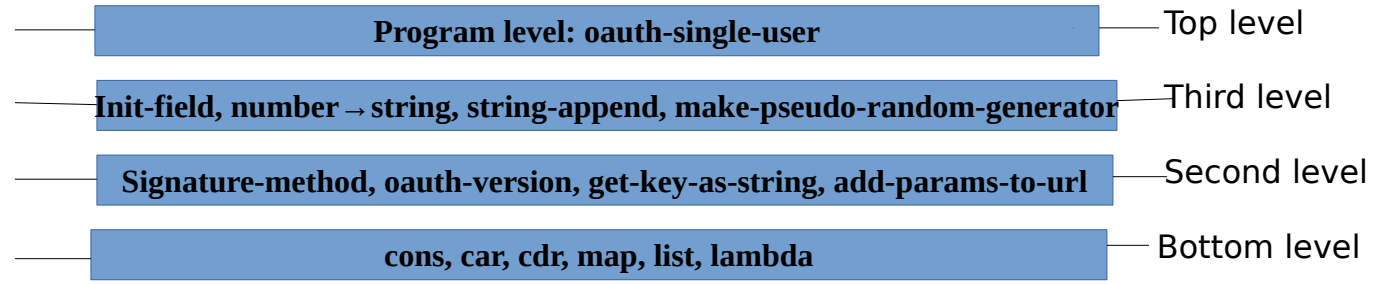
The qq-plot abstractions*



QTN.2 The New Abstractions

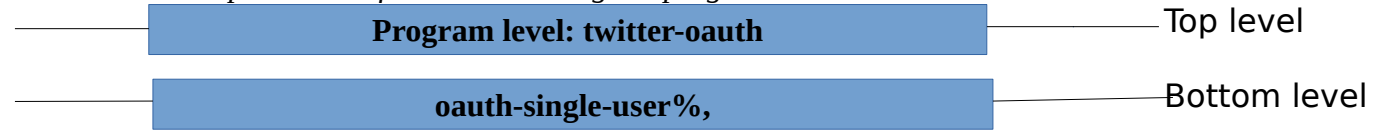
1. OAuth 1.0a for Twitter API authentication and data fetching

The following abstractions are found in the file *pick_data_from_twitter.rkt* in the project directory.The new abstractions start with picking data from twitter using the Twitter API abstraction. The authentication to use the twitter API starts with submitting the API request through http://developer.twitter.com.

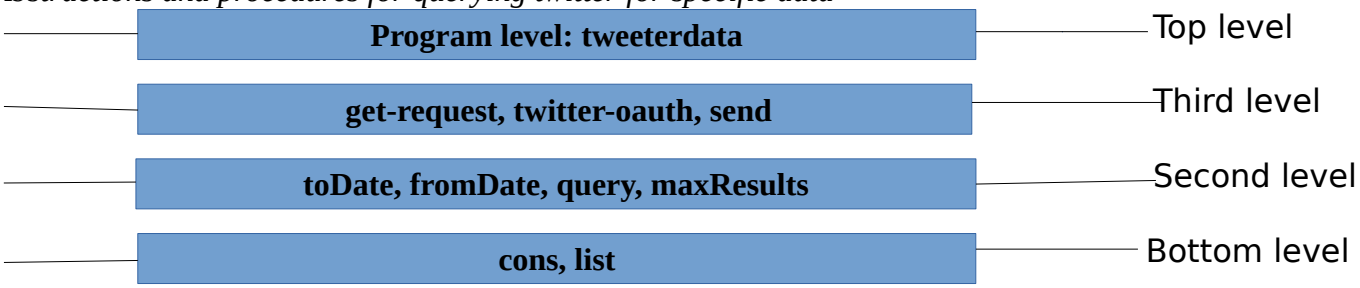


The new abstractions use *oauth-single-user* procedures to initiate the connection to twitter using the access keys and tokens are abstractions that were received from twitter developer account. The *init-field* service package from racket makes these abstractions available and ready to be used in the program

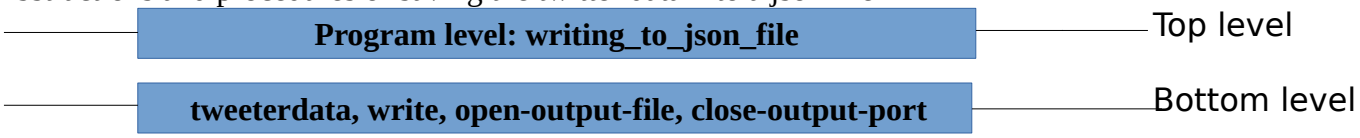
Abstractions and procedures for authenticating the program to twitter API



Abstractions and procedures for querying twitter for specific data



Abstractions and procedures of saving the twitter data into a json file



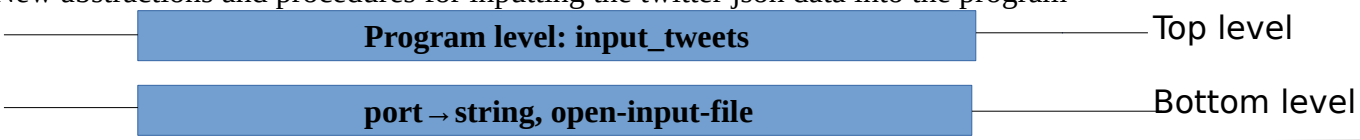
tweeterdata procedure uses *twitter-oauth get-request* abstractions to fetch 12 months data from twitter and query parameters such as *place_country*, *maxResult*, *fromDate* and *toDate* are used to limit the data to only Uganda and in a specified period of time as appended to the *url* request abstractions.

Procedure *writing_to_json_file* uses the *open-output-file* abstraction from racket to create a new file called data.json, the *write* abstraction then implements the *tweeterdata* procedure to save the fetched data to the json file that was created by *writing_to_json_file* procedures. Using *close-output-port* abstractions releases lower-level resources such as file handle to be used by the rest of the program at the analysis stage.

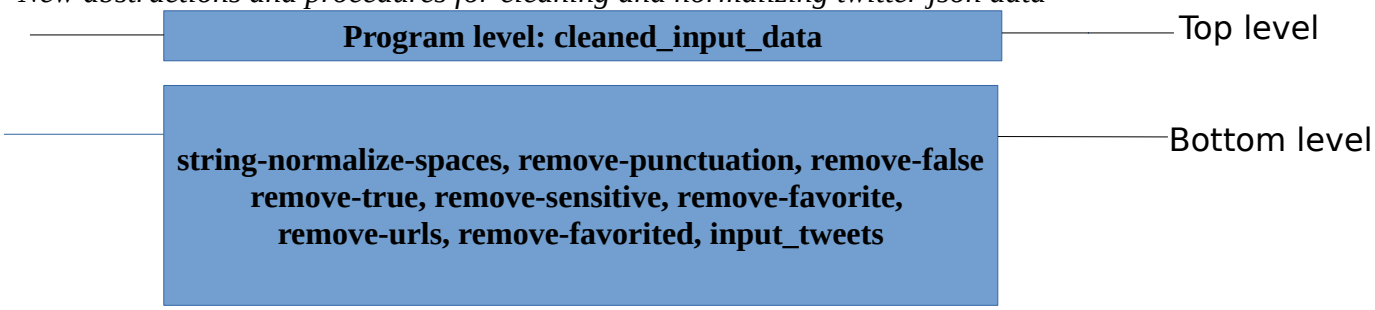
2. Analyzing the data picked from twitter

The following are the new levels of abstractions and procedures that I came up with while building on the data-science-master project. These are found in the file *twitter_mode_analysis.scm* in the project directory,

New abstractions and procedures for inputting the twitter json data into the program

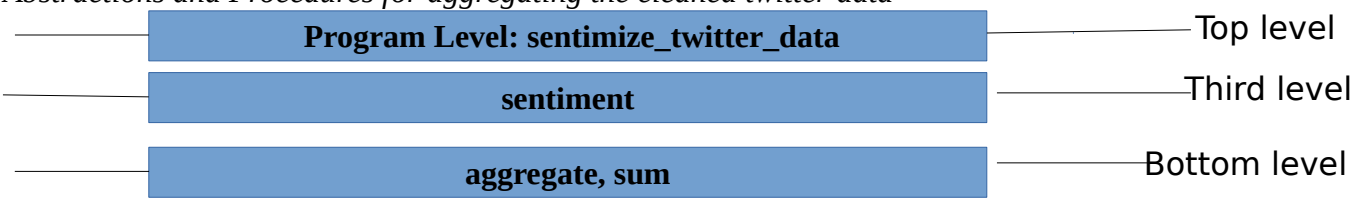


New abstractions and procedures for cleaning and normalizing twitter json data

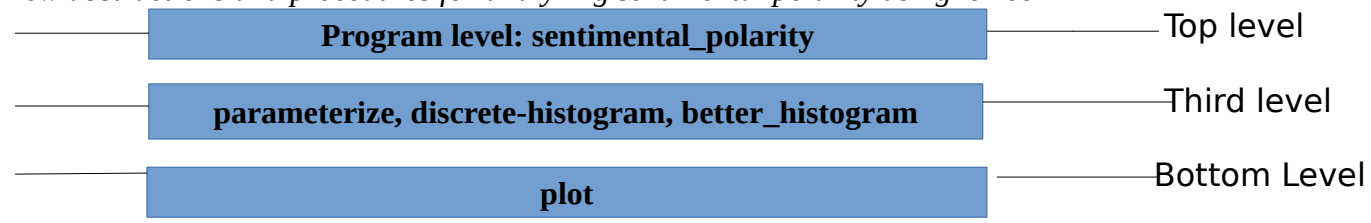


The abstractions *remove-punctuation* and *remove-url* have their procedures implemented in the data-science-master project and in this *cleaned_input_data* new abstraction they have been used with other new remove abstractions to clean the json twitter data.

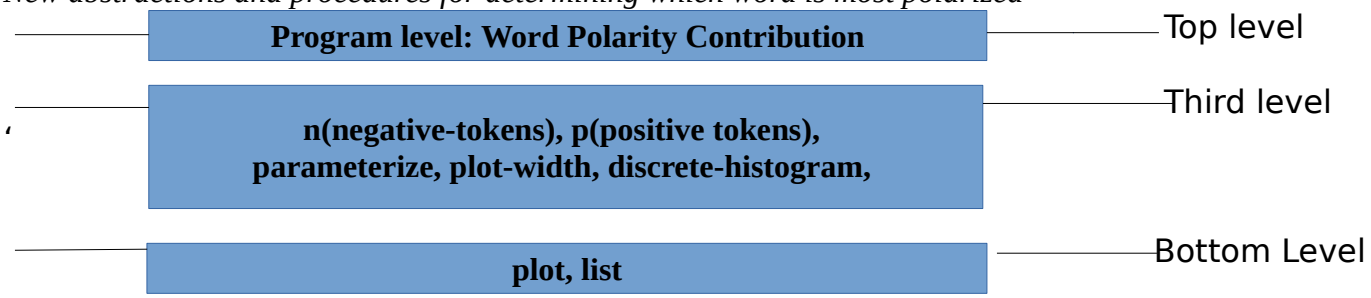
Abstractions and Procedures for aggregating the cleaned twitter data



New abstractions and procedures for analyzing sentimental polarity using lexicon



New abstractions and procedures for determining which word is most polarized



With motivation and design of *remove* procedures from the data-science-master project, I came up with the procedures to remove “false” “true” “favorite” “sensitive” key words from raw twitter file and replace them with an empty string “ ”. This was done so that the data analyzed is not affected by outliers.

After Normalizing the data, I used the *open-input-file* abstraction to create a procedure *input-tweets* to read the json file that has data to be analyzed

I used the procedure *cleanedinputdata* to combine the remove procedures as abstractions at this level to implement the cleaning of data in *input-tweet*

The *cleanedinputdata* procedure outputs quoted strings in the list. The order of the strings is not maintained from input to output, further cleaning is done by removing stop words using the defaults lexicon, *cleanedinputdata1* which returns a list of pairs, each pair consisting of a unique word or token from str with its frequency the number of times the word occurs in the tweets selected.

With the motivation from data-science-master project of data sentiment, I created a procedure called *sentimentize_twitter_data* that implements *cleanedinputdata1* procedures to return the aggregated twitter data which can now be visualized using various plotting tools . The following are the graphs visuaizing the cleaned twitter data using the new abstractions and procedures

Figure 2, shows the a bar plot of abstractions using bing lexicon to determine the ratio of positive words to the negative ones

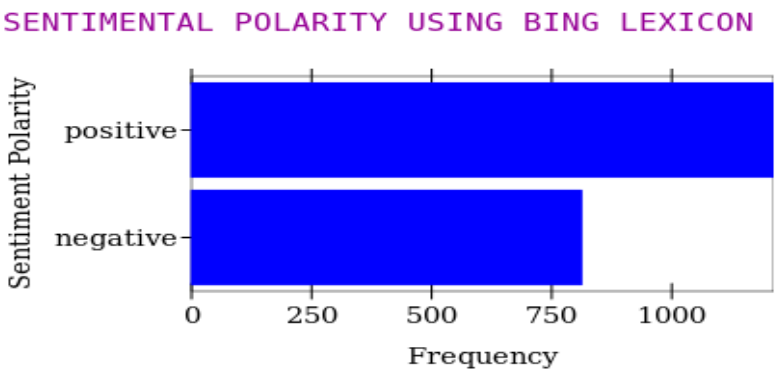


Figure 2: Using the bing lexicon to determine the ratio of positive-to-negative words

```
'(("positive" 5660)
  ("trust" 3790)
  ("disgust" 572)
  ("fear" 682)
  ("negative" 1350)
  ("sadness" 604)
  ("anticipation" 1335)
  ("surprise" 598)
  ("joy" 322)
  ("anger" 117))
```

Figure 1: Sentimental word analysis

Figure1 shows the sentiment for each word in the json file that have been aggregated and summed using same sentimental label

Figure3 shows a discrete histogram that visualizes the sentimental data abstractions using the automatic means of picking the bins which is an improvement on the hist abstractions implemented in the project.

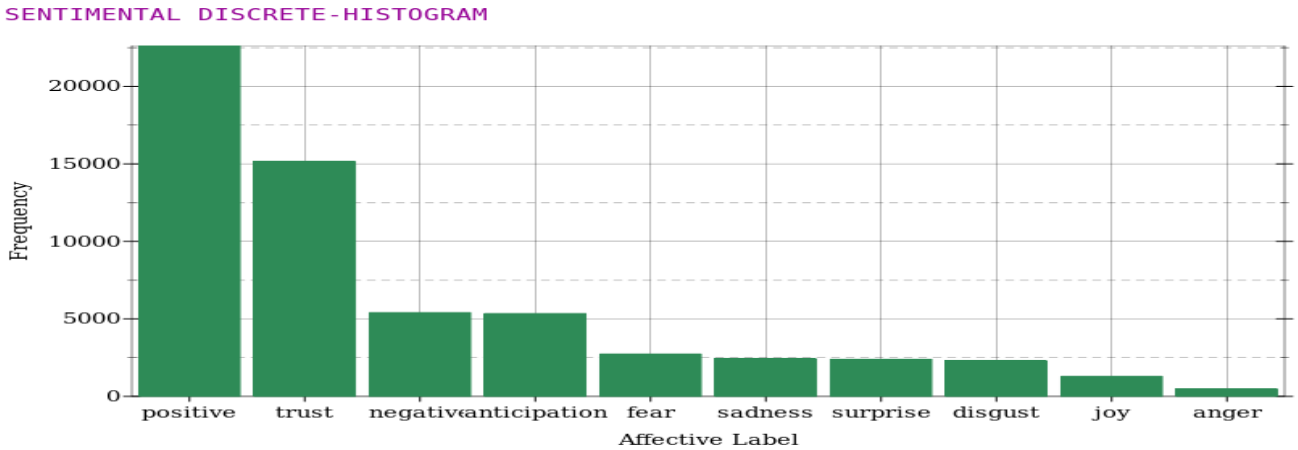


Figure 3: Visualizing the results as a barplot (discrete-histogram), a better hist

Figure 4 shows a plot of data abstractions of words that are mostly contributing to negative and positive sentimental scores, a sample of 20 influential positive and negatives were analyzed from data abstractions.

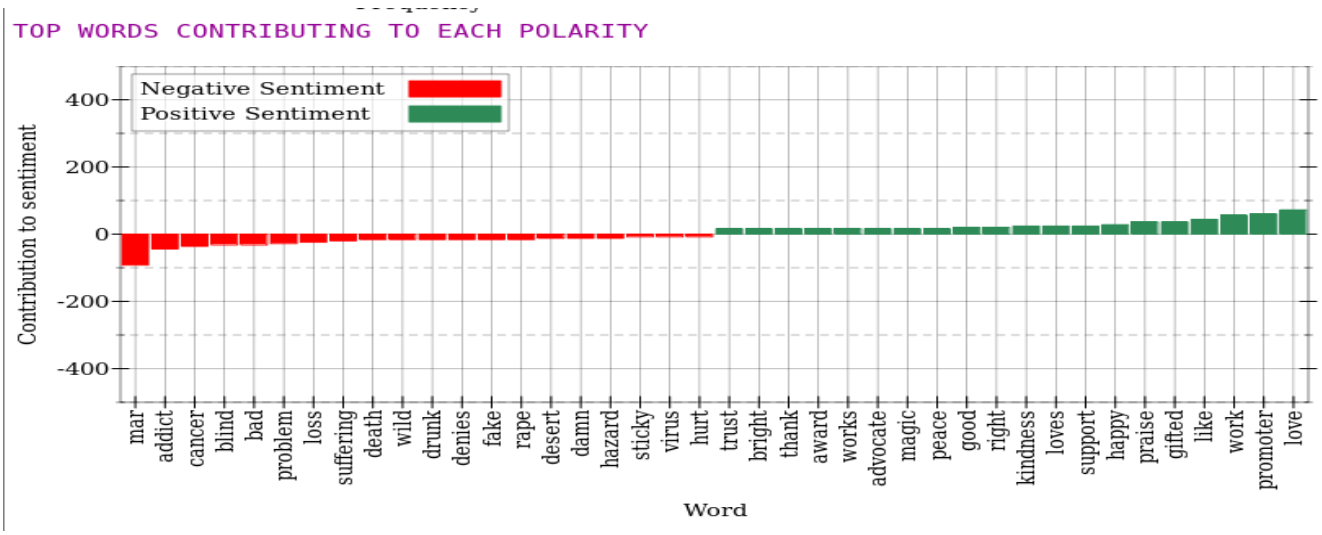


Figure 4: 20 positive and negative influential words contributing most to sentiment scores