Twitter Analysis DB Details

The main page for the project is **Twitter Analysis DB - OpenCircuits**

Contents

General

Look at the GUI Twitter Analysis DB GUI and just try it out. This page of documentation is both operational, that is using the app, and theoretical, as the ultimate guide to the app is to read the code. Module, class, and method names are fairly stable, but subject to change.

Performance

> I currently run the db on a ram disk, in any case put on your fastest drive.
>
> I have not tuned the db for performance. Got suggestions (that are more than just guesses) Let me know. I am sure that more indexing might help, will try in time.

Debugging

Initial debugging will probably consist of

- tweaking the parameter file for your computer
- Installing missing modules / packages.

The program does not come with an installer You should drop the code into your Python 3.6 development environment.

see: Twitter Analysis DB Configure to Run.

What is in the Database

First the database is really a database, it is not a csv file or a spreadsheet. It is an sql db in sql lite that could be upgraded to a more robust db if worth while.

Currently it consists of three tables:

tweets -- the actual tweet and supporting columns:

- tweet_id -- identifies the tweet
- tweet_datetime -- when tweeted
- tweet -- text of the tweet
- who -- who tweeted -- so far no real support for multiple tweeters, but not too hard to add to the rest of the application
- tweet_type -- a tweet by the author or a retweet
- ....

concord, or concordance -- this is every "word" from all the tweets. A concord.word has be sanitized from the raw tweet, it is always lower case and "dirt" like punctuation has been removed.

- word -- the cleaned up word
- tweet_id -- link to the tweet
- word_type -- just a word, or a "@reference" or a hashtag
- is_ascii -- indicator if the entire word is encoded in ascii, this help identify words that are possibly non-words like emogies.

words -- a list (currently about 300 K) of words and information on the amount of usage.

- word -- the word (again all lower case and normalized)
- word_count -- count of the uses in the analyzed body of text.
- word_rank -- index starting at 1 and acending from the highest_word_count to the lowest.

Tables can of course be joined, tweets to concord on tweet_id, and concord to words on word, the latter join may of course fail (produce a null in one table of the other so outer joins should be used.

Standard sql tools outside of the app can be used to view the data, Sqllite Studio, for example

Building a Database

I am working on providing DB building facilities from the GUI. Since this is sensitive to the input sources it only works with the type of input sources I have used. Not everything is in the GUI as of this writing, this will probably change.

Parameters

First to enable the GUI features you need to adjust the parameter file so

    self.show_db_def = True

Then you also need to point to the input files: (these are in the github repo)

- self.tweet_input_file_name = r"./input/all_tweets_may_16_for_2020.txt" # where tweets are
- self.word_input_file_name = r"./input/english-word-frequency/unigram_freq.csv" # word frequency data from kaggal

Then some processing options:

- self.who_tweets = "djt" # id for who tweets, not much used yet
- self.use_spacy = True # processing words to lemmas

Name a database, I would start with a file name that does not exist, as defining and loading the db will erase any current content. Or if you are only defining some of the tables, or adding to a table start with an existing db.

    self.self.database_name = "tweet_big_words.db" # database with lots of words loaded.

Run the GUI

- <Show Load Parameters> will show you the values of some of the parameters used to load the db. If you do not like what you get edit the parameter file. (and there is a button for that too)
- <Define Tweets Concord> will clear the tweets and concord tables, and initialize the columns in them.
- <Load Tweets File> will load the tweet input file and populate both tweets and concord tables.
- <Define Words> will clear the words table, and initialize the columns in it.
- <Load Words> will load the words input file and populate the words table.
- as loaded the words table does not populate the words.words_rank column, there is another utility to do that that still needs to be added to the GUI

Some Code and Theory

load_tweets_data.py contains the code for defining and loading the db tables. define_table_tweets() and define table_concord methods ... do the defominational work.... read the code.

A class TweetFileProcessor performs the table loads. Again read the code. The general idea is:

- Kick off from the main app from the method cb_load_tweets()
- Create TweetFileProcessor and its helpers TweetTableWriter and ConcordTableWriter, open the db, and input file
- Start processing with TweetFileProcessor.read_process_lines
- Open the file and connect to the db
- Read the lines in the file one by one to process them
- ..... parse the line ... split on tabs to get the individual source fields
- .... create parse_tweet (class ParsedTweet) from the line. ParsedTweet will parse the tweet (just the main text, the datetime.... have already been split off and saved
- .... in the parsing the follow take place.
    - tweet data is shifted to lower case.
    - certain characters are replaced
    - words are categorized (like url's @words, ...)
    - if certain words are present is_covid is set to true.
    - if parameters.use_spacy is True then spacy is used to convert words to lemmas
- * .... Parsed/extracted data is written to the respective tables
- files and the DB are closed.

I have tried to make the above accurate, but read the code.

TweetTableWriter and ConcordTableWriter are helper classes doing the actual table writing