Architecture/Cleaning

Alun

11 December 2015

Links to Literate Code Dcouments

Below are links to my github page and r markdown documents (documents which have code embedded and executed when published to ensure reproducibility) relating to preprocessing and querying of the data written the raw files of which can also be found on the github page.

- Github Page
- Data Preprocessing
- Query Code

Pipeline

The pipeline of the data was the following processes:

- Downloading and extracting the data
- Type conversion of file types
- id
  - A number (approx 1000) entries were found to have tweets stored in their "id" field. This judged to be full tweets (and often coordinates) which had filled from the left of the database so ended up in the wrong columns. These tweets were moved to the correct variable and other values set to NA. This process was repeated for the geo-location data which was also in the wrong columns for many of these entries.
  - Ensured that all values of id were unique to make sure multiple copies of the same tweet wern't present.
  - Ensured that the values were in an appropriate range. No negative values and values of the same approximate order, unlike the id_member attribute all these values should be (and were) around the same order of magnitude because they were created around the same time.
- id_member
  - Checked the range of values present. Some values were negative. These were just changed to be the positive equivalent. This was done because it didn't affect the number of unique users so I hypothesised that the error could have been just been the error in the sign not the value. Although you would have to do some more thorough research to verify this for stream based data and be careful about the forward compatability of that assumption.
- Text
  - Ensured length was in the correct limited boundary. This should be a length of characters 1:240 but some tweets up to 240 characters in length were found. The reason for this I believe is that non-standard characters like emoticons or chinese characters are sometimes interpreted as sequences of characters, because the values of these tweets were mainly made of non-stardard characters like accented characters and punctuation. Because of this I have left the tweets as they are. It would be very difficult to distinguish between the non-standard characters that were intentionally added and the ones introduced through these errors because its perfectly acceptable to send a tweet that is just "!!!!!" etc. This decision will impact on the length of tweets and average number of hashtags queries and given more time I would certainly of made efforts to clean some of these issues for those queries in particular.

    - Removed tweets that were empty, of which there were very few.

- Timestamp
    - Converted to POSIct type with 'UTC' time this is because the rmongodb package ensures interpretation of that type as the Date/time type in mongodb.
    - Ensured values were between correct limits. Did this by plotting a histogram to find the period of time in which tweets were taken and upon seeing this range make sure there were no outliers.
- Geo_lng
    - Made sure that the values were within the range of degrees possible. Upon inspecting the data found that all values were in a much smaller range corresponding to the UK, made sure there were no outliers to this.
- Geo_lat
    - Same as above

Although only the above changes were made I explored all the variables for the boundaries of possible values as I saw them.

Wherever any changes were made they the action was stored in an additional field called "anom". Therefore all the data was retained and could be reintegrated or changed whenever the situation arrises. For the most part however

It was the intention to pipe the dataset directly from R to mongodb using the 'rmongodb' package. This requires converting to BSON first and then uploading with an insert function. However there were sustained and robust errors concerning some entries not "being BSON" which crashed the uploads.

Attempts were made to export to mongoDb directly from R both as part of the data processing pipeline but problems were encountered using the interface between this package and mongodb (converting the data to BSON using mongo.bson.from.list then mongo.batch.insert doesn't seem to recognise some entries as BSON) so the cleaned data was saved to csv and imported using mongoimport.

Queries

The package 'rmongodb' was used to create the queries. As stated before there were a lot of issues encountered using this interface which impacted the queries as well despite spending many hours trying to find the cause/solution to the problem like encoding or using a cursor instead of batch functions etc. This prevented me from completing some of the queries but the code I was using to try and conduct them were left in place commented out (neccessary due to using Rmardown/knitr).