TALKING DATA

INTRODUCTION

The talking data (TD) dataset comes from a competition hosted on Kaggle several months ago. The general idea was to take the huge amount of accumulated data files and predict in which category (age and gender group) the user will be most likely to reside in. The categories (n = 12) are as follows:

```
'F23-', 'F24-26','F27-28','F29-32', 'F33-42', 'F43+',
'M22-', 'M23-26', 'M27-28', 'M29-31', 'M32-38', 'M39+'
```

DATA

The data that is available for prediction is as follows:

- In-app events (Event ID, timestamp + coordinate pair)
- Events (event ID, timestamp + coordinate pair)
- Event ID labels
- Phone brands/models

Test and training data labels

The data contains a lot of noise: e.g. missing data, 0/0 or 1/1 coordinate pairs, and so forth.

EVALUATION

The classification is evaluated using multi-class logarithmic loss. Each device will only be labelled with one true class.

OUR IDEAS

- Geolocation (reverse geocoding) – to trace back the coordinate pairs to cities, municipalities, provinces etc. Perhaps this could play a role in classification.
- Additionally we could treat the coordinates as a form of clustering
- Joining the data together
- We are still looking for a cross-validation structure, perhaps you could advice us?
- A Random Forest variant named the Mondrian forest seems interesting
- Failing that, RF and SVM, deep nn
- We want to keep using github for collab

OUR PROGRESS THUS FAR

- Geolocation

The initial idea with the geocoding was to reverse geocode the coordinates. This is a practice that has some experience in the group, though never at this volume. However, it quickly appeared impossible to use standard APIs:

- Google API is limited to 2500 requests a day, but not for paying users. At a rate of 50 cents per 1000 requests this would cost roughly (~36 million entries) 17900 dollars.
- Tigerlines has a somewhat accurate API, but at the moment it only recognizes up to 18 million locations and most are in the US.
- Openstreetmap's API (Nomatim) is limited to one request per second, which would mean we would still be going for close to 10000 hours. See you next year!
- We could clone Openstreetmap and run a copy with a local API, but this might take a fair bit of coding in an unfamiliar project.

My (Jeffrey) hunch is thus the following: it seems that for a good responsive API we are tied to third party services. However, from a past project I know that MongoDB allows you to query for polygons (shapes) such as city borders. Since MongoDB is pretty well established with big data, I expect less problems than running it with spatial joins (something I do not have experience with!) in python. I therefore propose the following pipeline

- I write a small script to load the data into MongoDB (as GeoJSON or plain CSV)
- I make a second script (JS) to read in polygons from a different folder, query the data using MongoDB's Mapreduce framework or Aggregation framework, and output the data to a separate collection for each polygon
- I run a third script to label each collection with the administrative area they are in, and publish these in the desired data format

The obvious problem with this is that it might take long to query plus that there is a little bit of coding time and making APIs work involved. Using a database for a bit of data manipulation raises the question why it can't be done in pre-processing.

DATA FUSION

Due to the fact that the dataset consisted of several csv files. One of the activities that needed to be done for this project, was to combine these different datasets into one universal dataset. This resulted into the following datasets:

- Checking Apps/Events: All apps that are combined with the events and their respective label;
- Phone Gender: All mobile phones combined with their respective owners, gender and location;
- Universal: All attributes combined into one universal database;
- Device Properties: All information concerning the mobile devices, location and installed apps.

OUR GROUP

A quick summary of our group and backgrounds (in no particular order)

- Jordi (point of contact) – HBO Business, IT and Management, Data Science
- Jeffrey – HBO Software Engineering, Data Science
- Gerdriaan – TRU/E security
- Roel – Data Science and Chemistry (Double master)
- Thijs – Data Science

We have named ourselves *classified*.

Raw assignments are so far:

Geo stuff: Jeffrey
Data joining: Jordi
Machine Learning algorithms: Gerdriaan and Roel
Deep learning algorithm: Thijs

(EXTERNAL) RESOURCES

https://arxiv.org/abs/1406.2673 - the Mondrian forest paper

https://www.kaggle.com/c/talkingdata-mobile-user-demographics/data - TalkingData Data Kaggle Link

https://docs.mongodb.com/manual/reference/operator/query/geoWithin/ – Geospatial querying in MongoDB