**Ideas/Questions**

* Build an index of all business groups using phones as tracking identifiers. (Anybody can change their names, but they will keep their phone numbers to allow people to find them.)
* Represent businesses in the map. From landline phones in the ads extract addresses via ETECSA db and find coordinates through some address-to-coordinates-service
* Figure out how to extract the price of a product and alert people when a product is bellow certain price (that would require immediacy, though, with constant scrapping, for instance)
* Do the people in different categories share more phone numbers?
* Use TF-IDF to extract the most important terms in each ad and also to calculate ads similarity by some kind of dot product between TF-IDF vectors
* Start exploring named entity recognition on the ads text to see what comes up.

1. **DB**

* Create ad table
* Create users table
* Write ad data in the DB
* Write user data in the DB
* Read from the DB

1. **Scrapping the data**:

* User name
* User Phone number(s)
* ID of the ad
* Title of the ad
* Classification, term/subterm (eg autos/alquiler)
* Content of the ad
* Price on the ad
* Date-time of the ad
* Is the ad “autorenovable”

1. **Cleaning**

* Clean each of the scrapped fields
* Remove duplicate ads
* THINK what to do with ads that have the same content but different title.
* THINK what to do with text inserted by agencies that has nothing to do with the ad.

1. **Feature extraction**

* All ads published by the same business person group (identified by co-appearing phones)

1. **Exploring data**

* Distribution of ad per time of the day and day of the week

1. **Modeling**

* Use a decision tree or a random forest to predict the classification of an advert from its title and content. Probably a good way would be to give the tree a bag of words with title and content merged. Although the site has a lot of misclassified ads, most of them are good. Perhaps in the future we could find a way of cleaning misclassified ads and training a better model.

1. **Code Performance Optimization**

* Profile the code, find bottlenecks
* Make the code parallel, capable of using more resources to speed up