RoundU – Create your own friend recommender on Twitter (analysis)

Garcia, Marc – Alonso, Aitor

Kschool

Finding good friends might be as hard as finding good data

Abstract

In this project we try to find the optimal course of action to develop a people recommender bot based on twitter activity. That so, we will explore the best way to extract and storage twitter data using SQL DataBases, apply Natural Language Processing to Raw Data (LDA & W2V) and recommend users based on topic similarity and multidimensional vector distance scores.

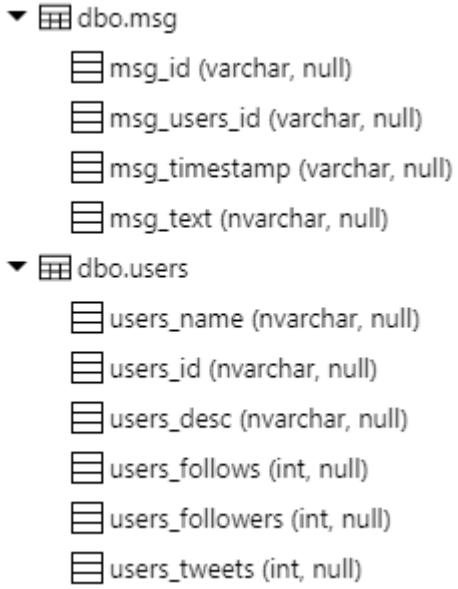
Keywords: Twitter, LDA, Word2Vec, Gensim, SQL, vector distance measuring

Project goal

As Twitter gathers user's interests, reactions and thoughts in natural language, we contemplate the possibility of mapping user's characteristics by analyzing their outputs to create a people recommender in order to facilitate users to make new friends in this increasing individualistic world. For this, we structure the steps to be developed as follows:

Infrastructure set-up

In order to be able to set up a twitter streaming listener, we create a Developer Twitter Account which access tokens and keys will be used to connect to Twitter's datahose. These credentials will be stored in *configLocal* file in order to ensure security on our process. This file, not supposed to be accessible to anyone outside the developer team, will be stored in Github for predictability purposes.
We will be storing our data in an Azure SQL DataBase. That so, we create an account and the Database with the following structure:



The connection string to this database shall be:

Driver={ODBC Driver 13 for SQL Server};Server=tcp:roundu.database.windows.net,1433;Database=RoundU_DB;Uid=roundu@roundu;Pwd={password};Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30;

Twitter Data extraction

As every data science project, the quality of the results depends directly from the quality of the data. That so, we write a python script to be executed in a PC for two months. This script creates a Tweepy streaming object able to absorb tweets geotagged in Barcelona Metropolitan Area and identified by Twitter as written in Spanish.

In the same object we set a function to extract json data claiming tweets to be in extended mode to avoid "140 character limitation". Also, in every tweet reception, two SQL queries are executed, uploading extracted data to SQL Database.

During this phase, having done different tests, we learnt that only ~2-3% of tweets are geotagged, generally coming from the same users who might have geotagging active in their smartphones. Also, despite having Twitter tagged a tweet as Spanish written, it might be wrongly tagged due to Spanish similarity to other languages such as Portuguese, Catalonian or Italian in some cases.

At the end of this phase, 45.000 tweets (aprox.) from 7.000 single users (aprox.) are gathered in the SQL Database, which gives a mean of 5 tweets/user that encourages us to move on to the next phase.

NLP - Data analysis

At this point, we analyze 2 different strategies to extract relevant information from users that could help us to categorize them:

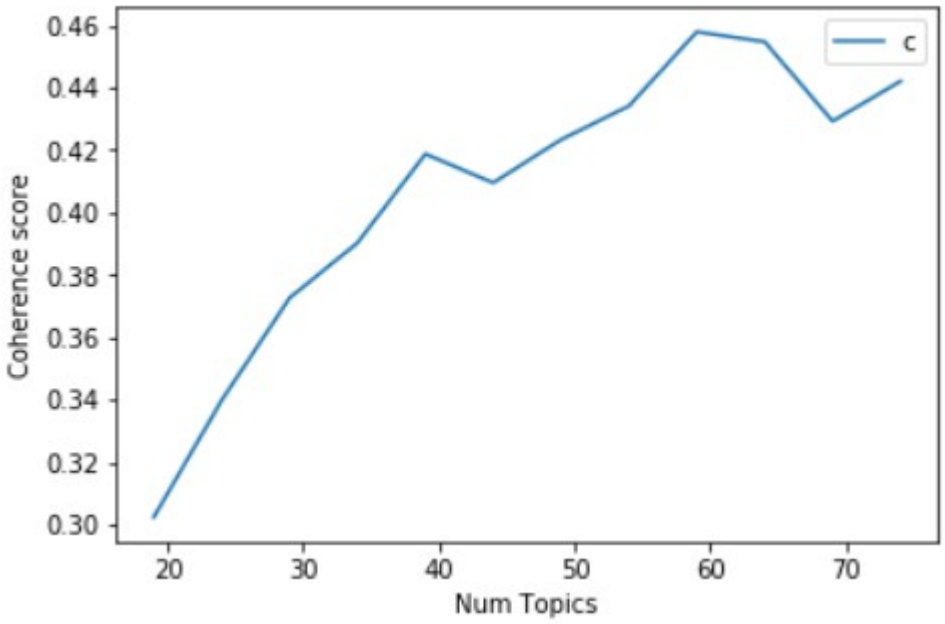
LDA [1]

In this approach, our next guess is that users who frequently mention a specific group of words in their tweets tend to be interested in same specific topics. That so, we find LDA fitting for our purpose. This algorithm treats received elements (corpus of user's tweets for us) as a bag of words where order is irrelevant in contrast to the occurrence frequency of every element in corpus.

This way, LDA understands every corpus as a mixture of categories where every word as a lead of the corresponding category or categories (overlap might occur). In this project we set LDA to distinguish 64 different word categories. This decision has been made based on distance between topics visualization, through picking a number of categories so every side of category spectrum has several representatives and avoiding them to group in one side of the visual quadrant. Also, we take care of not generating unnecessary category overlap, so classification is properly made.

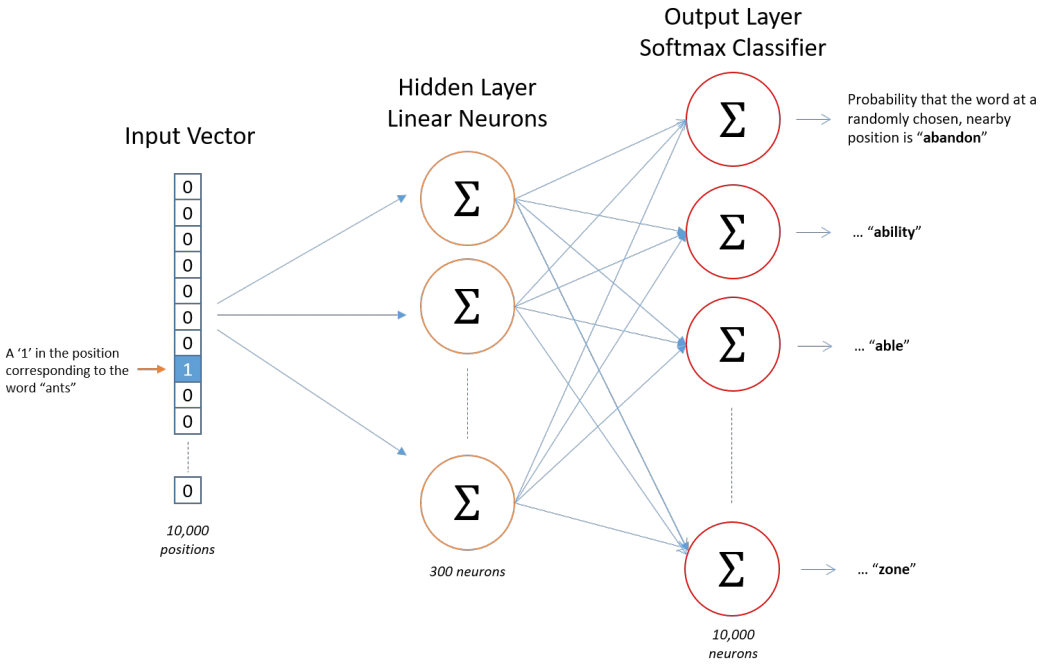
Other key element taken into account at setting optimal category number has been the coherence coefficient of every instance in our LDA model. The coherence coefficient keeps increasing as we train LDA with bigger amounts of categories until it stabilizes around 60 categories. That so, the biggest coherence coefficient we can achieve its on 0.45, which we consider enough.

Once LDA finds bags of words with biggest or lowest occurrence frequencies between them, its able to estimate the degree of belonging for each category for every new message given. This way, LDA will compare different tweets belonging to same or different categories. Therefore, going through every user's corpus, we are able to find users speaking about the same categories amongst them and on which frequency.



Word2Vec [2]

Our second strategy is to represent every word (from now on, token) in a vectorized space that could represent the semantic distance between tokens as numeric distance. To do so, we feed tokens as one-hot vectors to a neural network with a single standard fully-connected hidden layer of N neurons, so we can latter extract the weights as word embeddings of N dimensions. The output layer (softmax activation function) outputs probabilities for the target words from the vocabulary.



At the end of the training process, we will get a matrix of N dimensions for M unique tokens. This will help us on setting a measurable categorical space for every user.

The process to get this done will start by tokenizing every word in every tweet using regex functions and storing these tokens in a dictionary for every user. Then, in order to feed the model, a vocabulary of unique tokens and their frequencies will be needed.

Once the model has been trained, it can be visualized on Tensorboard, which will represent our multidimensional vectors through PCA on a 3D space.

In this project, we get 200 dimensional vectors for 10900 (aprox.) unique tokens.

At this point, its obvious that our model might not be accurate enough as in visualization similar meaning tokens appear in opposite sides of the space. That so, we bypass this weakness by loading a pretrained 100 dimensional model [3] trained on 218M tweets ($2.3x10^{12}$ words) collected between 2009 and 2019 in 333 different cities from Latin America, Spain and USA.

Applying TF/IDF[4] to every token in every user's corpus, we get a coefficient to calculate the weighted average vector of every user.

TF/IDF as term frequency–inverse document frequency measures the relevance of a token in a given corpus, defined as:

$$\mathbf{tf}(t, d) = \frac{f_d(t)}{\max_{w \in d} f_d(w)}$$

$$\mathbf{idf}(t, D) = \ln\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

$$\mathbf{tfidf}(t, d, D) = \mathbf{tf}(t, d) \cdot \mathbf{idf}(t, D)$$

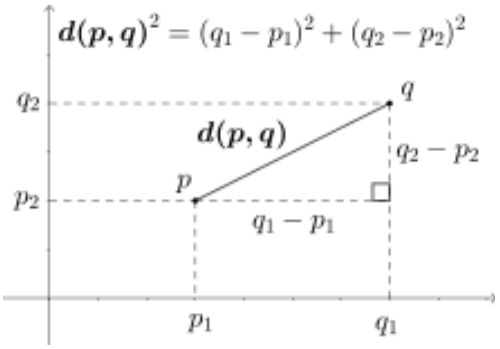
$$\mathbf{tfidf}'(t, d, D) = \frac{\mathbf{idf}(t, D)}{|D|} + \mathbf{tfidf}(t, d, D)$$

$f_d(t) :=$ frequency of term t in document d

$D :=$ corpus of documents

At this point, the matrix of 7.000 users x 100 dimensions (vectors) created will be used to calculate the distance between users:

Euclidean distance: it might be inertial to use Euclidean distance to calculate user similarity and this might be an error as vectors, while being continuous do not represent continuous information but categorical information located in a vectorized space.



Mahalanobis distance: as is suited to scale variations (remember that on each dimension, polars might refer to different scale of distance between meanings), vectors represent categorical data and takes into account correlations between dimensions.

$$D^2 = (\mathbf{x} - \mathbf{m})^{\mathbf{T}} \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m})$$

D^2 = Mahalanobis distance
$\mathbf{x}$ = Vector of data
$\mathbf{m}$ = Vector of mean values of independent variables
$\mathbf{C}^{-1}$ = Inverse Covariance matrix of independent variables
$\mathbf{T}$ = Indicates vector should be transposed

Bhattacharyya / Cavalli-Sforza[7]: used in genetics, allows us to measure distances between multidimensional categorical vectors, it is considered to be more reliable than the Mahalanobis distance, as the Mahalanobis distance is a particular case of the Bhattacharyya distance when the standard deviations of the two classes are the same:

$$D_B(p,q) = -\ln(BC(p,q))$$

where

$$BC(p,q) = \sum_{x \in X} \sqrt{p(x)q(x)}$$

$D_B(p,q)$ is the Bhattacharyya distance between p and q distributions or classes,
σ_p^2 is the variance of the p-th distribution,
μ_p is the mean of the p-th distribution, and
p, q are two different distributions.

At this point, for any new user, its text will be tokenized, vectorized and calculated its distance to other users, being recommended those to minimum distance.

Conclusions

After developing all this project, we get to the following conclusions:

- Twitter data extraction might present some difficulties and **restrictions imposed by Twitter itself or its API**.
- Regarding data, the main problem we have found is the difficulty of working with very fragmented, decontextualized texts, with very poorly-normalized and colloquial vocabulary. Also, due to Twitter User's dynamics, lots of times messages do not mean anything by themselves (ej: links to webpages, fast responses in a conversation with other users…). Given this, we strongly believe a previous strong data cleaning and filtering would be needed. We have observed that, otherwise, **the language analysis models are very vulnerable to noise, and offer non reliable data**.
- As for every model analyzed, data must be calculated for every pair of users, deploying this methodology in production environment might present challenging, requesting much time and computational effort.

Recommendations:

For further advances, we make recommendations as follows:

- Data gathering period should be continuous and possible parsing errors between twitter API and SQL database should be corrected.

- Implement a dictionary to transform emojis to significative tokens
- Increase dimensionality in Word2Vec model to 200-300.
- For LDA, generate a corpus of broadly-used words, like very common verbs on the language, to eliminate them from the messages on the preprocessing in order to reduce the noise given to the model.
- Discard users with less than 8 tokens (10% lowest percentile) or with low token appearance frequency in user's corpus.
- Analyze user similarity based on vectorized distance from hypernyms extracted form entities in corpus.

Next Steps:

Based on the analysis developed, we will continue by creating **a BOT (supported by Twitter's API) that will publish a message mentioning recommended similar users**, due to LDA topic grouping or lowest vectorized distance.

Having this done, we will analyze its accuracy and fiability through two metrics:

- Ascending evolution on User-bot interaction
- Rate of follows and follow-back between users.

References

1. https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
2. https://en.wikipedia.org/wiki/Word2vec
3. https://www.datos.gov.co/Ciencia-Tecnolog-a-e-Innovaci-n/WORD2VEC-Twitter-Espa-ol-para-Latinoam-rica-Espa-a/79c6-2d7z
4. https://es.wikipedia.org/wiki/Tf-idf
5. http://halweb.uc3m.es/esp/Personal/personas/agrane/ficheros_docencia/MULTIVARIANT/slides_Coorp_reducido.pdf
6. Encyclopedia of Distances, Michel Marie Deza, Elena Deza (Springer 2009)