実験詳細

1. Twitterで有料モードで収集されたツイートからジオタグ付きのツイートを取ってくる。counting\_geo.py

有料モードで収集されたものはrawファイル。そこからファイルごとにカウントする。

count\_geo\_per\_file method で集められたのがTotal 3004877でユーザ毎に上で作られたファイルから（geotag付きのみ）ツイート数をカウントする。

sum\_count\_geo method そこからトップユーザー４万人を選出する。ジオタグ付きのツイートをよく発しているトップユーザー。top\_geo\_existing\_user\_id\_40000.json

blacklistはbotとかが入ってる。つまり2000ツイート以上を発しているアカウント。

Output files: top\_geo\_user\_id\_40000.json

Blacklist.json

1. check\_users.py で実際に生きているユーザーリストを作る。db.lookupsに保存された。67268 existing users (como guarde encima del experimento anterior es mas de 40000 usuarios. Tengo que hacer de vuelta???)

そのdbからjson fileが作られた。

Output files:

top\_geo\_existing\_user\_id\_40000.json

Db.lookups (mongodb)

1. collecting\_tweet\_with\_API.py ~~でこれらのユーザのツイートを2018 年4月13日から2018年4月21日までの期間収集した~~． Top\_geo\_existing\_user\_id\_4000.jsonから

保存されたCollectionは以下の通り：

db.tweets\_from\_follow\_users0: 2926 registros

db.tweets\_from\_follow\_users1: 1459 registros

db.tweets\_from\_follow\_users2: 1021 registros

db.tweets\_from\_follow\_users3: 890 registros

db.tweets\_from\_follow\_users4: 665 registros

db.tweets\_from\_follow\_users5: 2672 registros

db.tweets\_from\_follow\_users6: 1949 registros

db.tweets\_from\_follow\_users7: 839 registros

Se puede recolectar en paralelo utilizando 4 cuentas de API cambiando los parámetros de la cuenta, el nombre del archivo

El comando line se escribe de esta forma: python collecting\_tweet\_with\_API.py 0

API 1

0: 0 a 5000

1: 5000 a 10000

API 2

2: 10000 a 15000

3: 15000 a 20000

API 3

4: 20000 a 25000

5: 30000 a 35000

API 4

6: 35000 a 40000

それらのツイートを取った後collect\_follow\_relationships methodでフォローユーザーのデータもとる。そしてdb.idsに格納。これもまた複数のAPIアカウント並行で行う。

それからmake\_user\_list\_for\_experiment methodで選ばれたユーザのツイート群（db.tweets\_from\_follow\_users）を4つのタイムウィンドウに分割してjson fileに出力する。ユーザーごとにツイート本文やGeotag付きの場所、タイムスタンプなどをまとめたもの。それがuser\_list.json。実際にこの中で生きているユーザ（いまだにツイートを発している人たち）をフィルタリングしなきゃいけないのでそれをuser\_link2.jsonに入れる。ですが実際はuser\_list.jsonをそのまま使うのが正確。なぜならこのリスト（user\_link2.json）を作るには全ての工程が終わった後でありこの時点では未来のツイートを取集できてるのは不可能であるから。

Output files:

Db.tweets\_from\_follow\_users0, Db.tweets\_from\_follow\_users1, Db.tweets\_from\_follow\_users2, Db.tweets\_from\_follow\_users3, Db.tweets\_from\_follow\_users4, Db.tweets\_from\_follow\_users5, Db.tweets\_from\_follow\_users6, Db.tweets\_from\_follow\_users7

Db.ids (user\_id ごとにfollowing してる人たちのリスト、twitter APIを使ってとってくる。逆にfollowerの方を簡単にとって来れないか検討！ただここでそれをすれば正解なのかが分からない)

ただcollect\_follow\_relationship()ではlookup\_url (users/lookup.json)を使っていてそれだとfollowingしてる人を取れないんじゃない？collecting\_tweets\_with\_API234-247らへんでrelationshipが分かるlink.jsonを作っているけど仕組みがよく分からない。なぜ [GET friends/ids](https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/api-reference/get-friends-ids)

 [GET friends/list](https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/api-reference/get-friends-list)を使わなかったのか。

1. classifier.pyで分類器にかける。そのためにはまずdictionaryを作らなければならない。

Corpus()クラスでBag of wordsを作る。そのCorpusには発信地推定器の学習データとして，**収集対象地域内**と**それ以外**の日本国内から発信された**ジオタグ付きツイート**それぞれ1万件入っている。もし指定されたareaがtsukubaだったならtrain\_include\_tsukuba (つくばから発信されたトレーニング用のツイート)、train\_exclude\_tsukuba (つくば以外から発信されたトレーニング用のツイート)、test\_include\_tsukuba (つくばから発信されたテスト用のツイート)、test\_exclude\_tsukuba (つくば以外から発信されたテスト用のツイート)となる。

実際にはareaごとにCorpusが作られてトレーニング、テストされるのだが、詳細はこちら

train\_clf method でトレーニングセットはランダムに使われる。もともとトレーニングに使われるデータセット＋テストで使われるデータセット。get\_tweets というメソッドで指定されたareaで使われたワードのセットが作られる。

ちなみにこのランダムに生成された順番（random\_indices）をログファイルに保存している。そうしてできたTweets\_insideとTweet\_outsideをマージする。それを元にdictionaryを作る。この作られたdictionaryからユーザごとにまとめたデータセットに適用する。~~そのユーザーが発したツイートの中にdictionaryにある言葉があればbagofwordsに追加される。ツイートをベクトル化する。~~

ここではトピックモデルのためのライブラリを使っている。Gensim

トピックモデルとは簡単にいうと、文章を内容からトピック（カテゴリ）を推定して分類するモデルです。

https://qiita.com/shizuma/items/44c016812552ba8a8b88

Doc2bowというgensimのメソッドは単語IDと頻度のタプルに変換出来ます。

<http://sucrose.hatenablog.com/entry/2013/10/29/001041>

そうやって前処理を施したトレーニング用データセットを分類機にかける。

この論文の場合はnaivebayesが良かったらしい。

ハイパーパラメータ設定はまた別のところでやっている。

目的変数seria la variable predictora y 説明変数 son las demas variables

そうやってscikit-learnで学習した分類器を保存する場合、joblib.dumpを使用する。

Test\_clf メソッドではテストデータで分類する。

ここでobtenerできるのは各クラスに属する確率がサンプル毎に取得できる。

例えばあるツイートがつくばに分類される確率とつくば以外に分類される確率。予測結果ですね。

<http://universityofbigdata.net/competition/tutorial/5681717746597888>

そしてclassification\_reportも作る。クラスごとにprecision, recall, f1-score, supportの結果を出してくれる。

5- Calc\_prob\_per\_tw.py ではモデルとdictionaryを作った後に確率を計算する。tweetごとに確率を計算してる。

Output files:

Db.tweets\_per\_user\_long, db.tweets\_per\_user\_long2, db.tweets\_per\_user\_long3, db.tweets\_per\_user\_long4

6- Db\_to\_json.py で足りないデータをmongodbから出力する。ツイートの確率だけが入ってるデータなど。例えば京都から発したであろうユーザのツイートの確率。ただuser\_id\_to\_data\_for\_eval\_prepared\_geo2\_short.jsonはどこからやってきたのだろう？（他のコードで探してみたけどoutput fileとしては無かった）

Output files:

Cnt\_geotag\_short.json

7- experiment.pyでバンディットアルゴリズムを実行する。

Get\_result\_of\_experiments で報酬の計算をする。

まず、paramsに全てのパラメータを入れる（どのアルゴリズムを使うか、各アルゴリズムのパラメーターepsilon greedy, epsilon alpha greedy dynamic）

そしてget\_follow\_users\_by\_epsilon\_greedyとか（dependiendo que algoritmo usar）でタイムウィンドウごとに千人のユーザーを取ってくる。

報酬の計算とはあるユーザーをフォローしたところそのユーザーが発したツイートの位置推定を行う。以下中川さんのスライドから：

提案手法はフォロー候補ユーザ集合とそれらのフォロー関係を入力とし

処理の流れは順に

フォローユーザ選択

フォローユーザからツイートを収集

フォローユーザの評価

収集手法のパラメタ更新

を繰り返します

1サイクルごとに、つまり1つのタイムウィンドウごとにフォローしたユーザ集合と収集したツイート集合を出力します

この実験ではtest\_periodを短い期間と長い期間のテストデータを対象にしています。Short /long

Short だとtotal\_W 全タイムウィンドウ数が５２、longだと２６７タイムウィンドウ

ファイルはユーザリスト　user\_list.json (short/long 別ファイル)

例えばget\_follow\_users\_by\_epsilon\_greedyメソッドではuser\_list の中から一人ずつ取ってきて

Summarize\_evaluated\_n\_tweets.py

**Mongoimport mongoexportの使い方**

Es un ejecutable, por tanto no se ejecuta en la consola de mongo corriendo.

Abres otra consola y vas a la carpeta bin/

Ejecutar el comando: mongoexport –db Tweets –collection tweets\_from\_users –file /home/miwayoshi/

mongoimport --db dbName --collection collectionName --file fileName.json --jsonArray