3D YEAR PROJECT
SENTIMENT ANALYSIS OF ADVERTISEMENT TEXTS BASED ON TWITTER MESSAGES

15 janvier 2017

Irina Stolbova

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

INTRODUCTION

This project is being done in collaboration with the start-up company Nelper who is currently developing the software being able to automatically generate concise advertisement messages for their customers. Creating an attractive informative message out of a product description text, Nelper software will help the customers to lift their ranking at the search system (on google page for example). As a part of the team, we are working on the sentiment analyses of the generated advertisements. Based on the Twitter messages sample, we are performing an algorithm being able to predict whether the sentence is positive, negative or neutral in terms of sentiment analysis. Designed mostly for the French clients, the algorithm should be adapted for the French language.
We mainly use the paper of Alexander Pak and Patrick Paroubek "Twitter as a Corpus for Sentiment Analysis and Opinion Mining" to guide us at our steps.

TWITTER PARSER

The initial step for the algorithm development is to collect the Twitter messages. Using the python *tweepy* library we collected a corpus of text posts and formed a data set of two classes : positive sentiments, negative sentiments. To collect negative and positive sentiments
We have created two files : the one with the positive twitters and the one with negatives ones. Each category has been collected by specifying the language of the retrieved posts (French) and filtering for two types of emoticons :

— Happy emoticons : [' :)', ' :-)', '=)', ' :D', ' ;-)', ' ;)', '(- ;', '(- :', '(:', '(;', '(=', ' :d', ' :p']
— Sad emoticons : [' :(', ' :-(', '=(', ')- :', ') :', ') ;', ')=', " :'(", " :'("]

TEXT PREPROCESSING

As soon as we have collected the posts, they should be preprocessed. At this part we create the preprocessing function. In order to better understand its steps, we'll take one of the positive twitts as an example.

— Initial twitt :
"Il neige @neige encore ce matin ! :) c'est très beau ! On va avoir une belle hivers on dirait ! Nous n'allons pas nous ennuier ! ! ! hivers neige"
— removing of HTML tags, @-mentions, -words, URLs, numbers and emoticons from the text :
"Il neige encore ce matin ! On va avoir une très belle hivers on dirait ! On va pas s'ennuier ! ! !"
— tokinization (devision into separate words), bringing the words to the lower case and stemming (reducing to the same root eliminating the endings) :
['il','neig','encor','ce','matin',' !','c',"'",'est','tres','beau',' !','on','va', 'avoir','une','bel','hiver','on','dir',' !','nous','n',"'",'allon','pas','nous','ennui',' !',' !',' !']
— removing the french stopwords (but leaving the punctuation) :
['neig','encor','matin',' !','tres','beau',' !','va','avoir','bel', 'hiver','dir',' !','allon','pas','ennui',' !',' !',' !']
— bringing the tokenized text back to the string :

"neig encor matin ! tres beau ! va avoir bel hiver dir ! allon pas ennui ! ! !"

— tokenizing the new string created at the previous step taking into account that if "word" is not a punctuation symbol ("word"+"pas"), ("pas"+"word"), ("trop"+"word"), ("tres"+"word"), ("moins"+"word") form the bigramms and removing the punctuation :

['neig','encor','matin','tres beau','va','avoir','bel','hiver','dir','pas ennui','allon pas']

To implement the stages above there have been used such python libraries like *re* for the text patterns recognition, *string* for the punctuation symbols and *nltk.stem.snowball* for the french stemmer function.

FROM RAW TWITS TO THE TF-IDF MATRIX

At the next step each message is tokenized using the function *preprocess()* described above and for every term we calculate the number of its occurrences in the whole text. Afterwards we don't consider the terms appearing less than 5 times in the document.
In information retrieval, tf–idf is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus. Using the *sklearn* python library, we generate the tf-idf matrix which reflects the tf-idf coefficient of the term in the specific twitter message with lines corresponding to messages and columns to terms' names.

MACHINE LEARNING FOR THE SENTIMENT PREDICTION

As a predictor algorithm we are using the random forest regressor which takes the tf-idf matrix as a matrix of features and the vector of ones and zeros as labels with "1" corresponding to a positive twitt and "0" to a negative one.
The next step is to divide the data into two parts for training and testing purpose for the cross validation. The error measuring the quality of the predictor is the ratio between the number of incorrect answers of the predictor to the overall number of testing messages.

RESULTS

At the moment we have collected 24627 twitts and we continue updating our data with new ones. This amount of information contains 3117 words that appear more then 5 times.
Currently with the developed method we have reached the result of 18% of error.

STEPS FORWARD

The list of things that still need to be added and improved :

— The preprocessing function needs to be reviewed and checked for the possible inconsistencies

— Collecting more twitts to give a more precise prediction result.

— Including the messages with neutral sentiments in the dataset.

— Implementing the predictor to the advertisement messages taking from the outside and not from the current sample.