Opinion

June 17, 2019

1 TP : Analyse des opinions sous twitter

Maël Fabien

```
In [451]: ### General import
          import pandas as pd
          import numpy as np
          import os
          import re
          from sklearn.metrics import accuracy_score

          ### NLTK
          from nltk.corpus import sentiwordnet as swn
          from nltk.corpus import wordnet as wn
          import nltk as nltk
          from nltk.tokenize.casual import TweetTokenizer

          nltk.download('sentiwordnet')
```

```
[nltk_data] Downloading package sentiwordnet to
[nltk_data]     /Users/maelfabien/nltk_data...
[nltk_data]   Package sentiwordnet is already up-to-date!
```

```
Out[451]: True
```

1.1 I. Importer les fichiers

Les tweets a analyser sont disponibles a l'adresse suivante : https://clavel.wp.imt.fr/files/2018/05/testdata.manual.2009.06.14.csv_.zip. Cette base (Sentiment140) a ete obtenue sur le site de l'universite de Stanford http://help.sentiment140.com/for-students. Un extrait en est donne dans le tableau 1. La base contient 498 tweets annotes manuellement. La base propose 6 champs corres- pondant aux informations suivantes : 1. la polarite du tweet : Chaque tweet est accompagne d'un score pouvant etre egal a 0 (negatif), 2 (neutre) ou 4 (positif). 2. l'identifiant du tweet (2087) 3. la date du tweet (Sat May 16 23 :58 :44 UTC 2009) 4. la requete associee (lyx). Si pas de requete la valeur est NO_ QUERY. 5. l'utilisateur qui a tweete (robotickilldozr) 6. le texte du tweet(Lyx is cool)

```
In [452]: tokenizer = TweetTokenizer(strip_handles=True, reduce_len=True)
```

```
In [453]: # Import data
          df = pd.read_csv('testdata.manual.2009.06.14.csv', header=None)
          df = df.drop([1], axis=1)
          df.head(15)
```

```
Out[453]:     0                             2        3            4  \
          0   4  Mon May 11 03:17:40 UTC 2009  kindle2       tpryan
          1   4  Mon May 11 03:18:03 UTC 2009  kindle2       vcu451
          2   4  Mon May 11 03:18:54 UTC 2009  kindle2       chadfu
          3   4  Mon May 11 03:19:04 UTC 2009  kindle2        SIX15
          4   4  Mon May 11 03:21:41 UTC 2009  kindle2     yamarama
          5   4  Mon May 11 03:22:00 UTC 2009  kindle2  GeorgeVHulme
          6   0  Mon May 11 03:22:30 UTC 2009      aig      Seth937
          7   4  Mon May 11 03:26:10 UTC 2009   jquery    dcostalis
          8   4  Mon May 11 03:27:15 UTC 2009  twitter      PJ_King
          9   4  Mon May 11 03:29:20 UTC 2009    obama  mandanicole
          10  2  Mon May 11 03:32:42 UTC 2009    obama         jpeb
          11  0  Mon May 11 03:32:48 UTC 2009    obama  kylesellers
          12  4  Mon May 11 03:33:38 UTC 2009    obama  theviewfans
          13  4  Mon May 11 05:05:58 UTC 2009     nike       MumsFP
          14  0  Mon May 11 05:06:22 UTC 2009     nike   vincentx24x

                                                               5
          0   @stellargirl I loooooooovvvvvveee my Kindle2. ...
          1   Reading my kindle2...  Love it... Lee childs i...
          2   Ok, first assesment of the #kindle2 ...it fuck...
          3   @kenburbary You'll love your Kindle2. I've had...
          4   @mikefish  Fair enough. But i have the Kindle2...
          5   @richardebaker no. it is too big. I'm quite ha...
          6   Fuck this economy. I hate aig and their non lo...
          7                        Jquery is my new best friend.
          8                                        Loves twitter
          9   how can you not love Obama? he makes jokes abo...
          10  Check this video out -- President Obama at the...
          11  @Karoli I firmly believe that Obama/Pelosi hav...
          12  House Correspondents dinner was last night who...
          13  Watchin Espn..Jus seen this new Nike Commerica...
          14  dear nike, stop with the flywire. that shit is...
```

```
In [454]: # Compute sentiment
          score = {}
          score[0] = 'Negatif'
          score[2] = 'Neutre'
          score[4] = 'Positif'

          df['sent'] = df[0].apply(lambda x : score[x])
```

1.2 II. Pretraitements

Les tweets contiennent des caracteres speciaux susceptibles de nuire a la mise en place des methodes d'analyse d'opinions. Ecrire un programme permettant pour chaque tweet de :

- recuperer le texte associe
- segmenter en tokens
- supprimer les urls
- nettoyer les caracteres inherents a la structure d'un tweet
- corriger les abreviations et les specificites langagieres des tweets a l'aide du dictionnaire DicoS- lang (fichier SlangLookupTable.txt disponible ici : https://clavel.wp.imt.fr/files/2018/06/ Lexiques.zip), encodage du fichier : latin1

```
In [455]: # Slang lookup table
          slang = pd.read_csv('Lexiques/SlangLookupTable.txt', encoding='latin1', sep='\t', hea
          slang.head()
```

```
Out[455]:        0                    1
          0    121           one to one
          1  a/s/l  age, sex, location
          2    adn          any day now
          3  afaik    as far as I know
          4    afk  away from keyboard
```

Vous preciserez dans le CR le nombre d'occurrences des caracteres inherents a la struc- ture du tweet et le nombre d'occurrences des 'hash-tags' dans le corpus.

```
In [456]: # Count characters
          def count_char(txt) :

              count_hastag = 0
              count_struc = 0

              for string in txt :
                  count_hastag += string.count("#")
                  count_struc += string.count("@")

              return "There are %d hashtags and %d tweet inherecent characters." % (count_hasta
```

```
In [457]: count_char(list(df[5]))
```

```
Out[457]: 'There are 52 hashtags and 128 tweet inherecent characters.'
```

```
In [458]: # Clean the text

          def treat_text(txt) :

              # Remove URLs
```

```
        txt = re.sub(r'^https?:\/\/.*[\r\n]*', '', txt, flags=re.MULTILINE)

        # Replace characters
        txt = txt.replace("@", "")
        txt = txt.replace("#", "")

        # Tokenize
        tokens = tokenizer.tokenize(txt)

        # Replace slang language
        for i in range(len(tokens)) :
            if tokens[i] in slang[0] :
                tokens[i] = slang[slang[0] == tokens[i]][1]

        return tokens
```

```
In [459]: tokens = df[5].apply(lambda x : treat_text(x))
```

```
In [460]: tokens[:15]
```

```
Out[460]: 0     [stellargirl, I, looovvveee, my, Kindle, 2, .,...
          1     [Reading, my, kindle, 2, ..., Love, it, ..., L...
          2     [Ok, ,, first, assesment, of, the, kindle, 2, ...
          3     [kenburbary, You'll, love, your, Kindle, 2, .,...
          4     [mikefish, Fair, enough, ., But, i, have, the,...
          5     [richardebaker, no, ., it, is, too, big, ., I'...
          6     [Fuck, this, economy, ., I, hate, aig, and, th...
          7                  [Jquery, is, my, new, best, friend, .]
          8                                      [Loves, twitter]
          9     [how, can, you, not, love, Obama, ?, he, makes...
          10    [Check, this, video, out, -, -, President, Oba...
          11    [Karoli, I, firmly, believe, that, Obama, /, P...
          12    [House, Correspondents, dinner, was, last, nig...
          13    [Watchin, Espn, .., Jus, seen, this, new, Nike...
          14    [dear, nike, ,, stop, with, the, flywire, ., t...
          Name: 5, dtype: object
```

1.3 III. Etiquetage Grammatical

Developper une fonction capable de determiner la categorie grammaticale (POS : Part Of Speech) de chaque mot du tweet en utilisant la commande suivante de la libraire nltk :

```
In [461]: def POS(tokens) :

              list_pos = []

              # For each token
              for token in tokens:
```

```
                # Append the POS tag
                list_pos.append(nltk.pos_tag(token))

            return list_pos
```

```
In [462]: taggedData = POS(tokens)
```

1.4 IV. Algorithme de détection - V1

NLTK dispose entre autre d'une interface pour manipuler la base de donnees WordNet. Ainsi, apres installation de NLTK et du package WordNet, un utilisateur peut acceder a l'ensemble des synsets qui sont lies a un mot donne a l'aide d'une commande simple sous Python. Observez son fonctionnement a l'aide des lignes de code suivantes :

```
In [464]: from nltk.corpus import wordnet as wn
          wn.synsets('dog')
```

```
Out[464]: [Synset('dog.n.01'),
           Synset('frump.n.01'),
           Synset('dog.n.03'),
           Synset('cad.n.01'),
           Synset('frank.n.02'),
           Synset('pawl.n.01'),
           Synset('andiron.n.01'),
           Synset('chase.v.01')]
```

Pour cette etape, vous devez developper un programme permettant : - de recuperer uniquement les mots correspondant a des adjectifs, noms, adverbes et verbes

```
In [465]: def sort_words(taggedData) :
              list_token = []

              # For each tweet
              for tokens in taggedData :
                  intermediate_list_token = []
                  # For each token
                  for token in tokens :
                      # Keep only tokens whose POS starts with the following letters
                      if token[1][:2] == 'JJ' or token[1][:2] == 'NN' or token[1][:2] == 'VB' o
                          intermediate_list_token.append(token[0])
                  list_token.append(intermediate_list_token)
              # Returns a list of lists
              return list_token
```

```
In [466]: filtered_words = sort_words(taggedData)
```

- d'acceder aux scores (positifs et negatifs) des synsets dans la librairie NLTK. Ce script definira dans une classe Python l'objet SentiSynset sur le meme modele que le Synset developpe dans NLTK pour WordNet, et permettra de lire le tableau de SentiWordNet comme suit.

- de calculer pour chaque mot les scores associes a leur premier synset,
- de calculer pour chaque tweet la somme des scores positifs et negatifs des SentiSynsets du tweet,
- de comparez la somme des scores positifs et des scores negatifs de chaque tweet pour decider de la classe a associer au tweet.

```
In [467]: def compute_score_v1(taggedData) :

              # Move the function defined above here
              def sort_words(taggedData) :
                  list_token = []
                  for tokens in taggedData :
                      intermediate_list_token = []
                      for token in tokens :
                          if token[1][:2] == 'JJ' or token[1][:2] == 'NN' or token[1][:2] == ''
                              intermediate_list_token.append(token[0])
                      list_token.append(intermediate_list_token)
                  return list_token

              filtered_words = sort_words(taggedData)
              score_tweet = []

              # For each tweer
              for tweet in filtered_words :

                  # Initalize the scores
                  score = 0
                  score_pos = 0
                  score_neg = 0

                  # For each token within the tweer
                  for token in tweet :

                      try :
                          # Try to compute and add the positive and negative scores linked to
                          score_pos += swn.senti_synset(wn.synsets(token)[0].name()).pos_score
                          score_neg += swn.senti_synset(wn.synsets(token)[0].name()).neg_score

                      except :
                          pass

                  # Format of the output : score pos, score neg, label
                  if score_pos > score_neg :
                      score_tweet.append([score_pos, score_neg, 'Positif'])
                  elif score_pos == score_neg :
                      score_tweet.append([score_pos, score_neg, 'Neutre'])
                  else :
                      score_tweet.append([score_pos, score_neg, 'Negatif'])
```

```
        return np.array(score_tweet)
```

```
In [468]: score_v1 = compute_score_v1(taggedData)
```

```
In [469]: accuracy_score(df['sent'], score_v1[:,2])
```

```
Out[469]: 0.5341365461847389
```

```
In [470]: sum(df['sent'] == score_v1[:,2])
```

```
Out[470]: 266
```

L'accuracy atteint 53.4% et on identifie correctement 266 labels sur les 498.

1.5 Algorithme de détection - V2

Vous aurez besoin de : - la liste des mots en anglais correspondant a des negations (fichier NegatingWordList.txt disponible ici : https://clavel.wp.imt.fr/files/2018/06/Lexiques.zip) - et celle correspondant aux modifieurs (fichier BoosterWordList.txt disponible ici : https://clavel.wp.imt.fr/files/2018/06/Lexiques.zip).

Pour chaque mot, l'algorithme doit effectuer les operations suivantes : - multiplie par 2 le score negatif et le score positif associes au mot si le mot precedent est un modifieur ; - utilise uniquement le score negatif du mot pour le score positif global du tweet et le score positif du mot pour le score negatif global du tweet si le mot precedent est une negation.

```
In [471]: negating = ["aren't","arent", "can't", "cannot", "cant", "don't", "dont", "isn't", "i
```

```
In [472]: booster = pd.read_csv('Lexiques/BoosterWordList.txt', header=None, delimiter='\t')
          booster
```

```
Out[472]:                 0  1
          0       absolutely  1
          1       definitely  1
          2        extremely  2
          3           fuckin  2
          4          fucking  2
          5           hugely  2
          6       incredibly  2
          7             just -1
          8   overwhelmingly  2
          9               so  0
          10            some -1
          11             sum -1
          12            very  1
```

```
In [480]: def compute_score_v2(taggedData) :

              def sort_words(taggedData) :
                  list_token = []
```

```
    for tokens in taggedData :
        intermediate_list_token = []
        for token in tokens :
            if token[1][:2] == 'JJ' or token[1][:2] == 'NN' or token[1][:2] == 'V
                intermediate_list_token.append(token[0])
        list_token.append(intermediate_list_token)
    return list_token

filtered_words = sort_words(taggedData)
score_tweet = []

for tweet in filtered_words :
    modifier = None
    negator = None

    score = 0
    score_pos = 0
    score_neg = 0

    total_negating = 0
    pos_negating = 0

    for token in tweet :

        try :
            # Check if the previous word in negating list (and inverse scores)
            if negator in negating :
                # Total negating words
                total_negating += 1
                # Check if previous word in booster list (and double score)
                if modifier in booster[0] :
                    score_pos += swn.senti_synset(wn.synsets(token)[0].name()).ne
                    score_neg += swn.senti_synset(wn.synsets(token)[0].name()).po
                else :
                    score_pos += swn.senti_synset(wn.synsets(token)[0].name()).ne
                    score_neg += swn.senti_synset(wn.synsets(token)[0].name()).po
            else :
                if modifier in booster[0] :
                    score_pos += swn.senti_synset(wn.synsets(token)[0].name()).po
                    score_neg += swn.senti_synset(wn.synsets(token)[0].name()).ne
                else :
                    score_pos += swn.senti_synset(wn.synsets(token)[0].name()).po
                    score_neg += swn.senti_synset(wn.synsets(token)[0].name()).ne

        except :
            pass

        # Set the current token as the modifier and the negator for the next ite
```

```
                    modifier = token
                    negator = token


                if score_pos > score_neg :
                    # Count number of positive tweets with negating words
                    if total_negating > 0 :
                        pos_negating =+ 1
                    score_tweet.append([score_pos, score_neg, 'Positif'])
                elif score_pos == score_neg :
                    score_tweet.append([score_pos, score_neg, 'Neutre'])
                else :
                    score_tweet.append([score_pos, score_neg, 'Negatif'])
            print("Total number of negating terms in positive tweets : " + str(pos_negating))
            return np.array(score_tweet)
```

```
In [481]: score_v2 = compute_score_v2(taggedData)
```

```
Total number of negating terms in positive tweets : 0
```

```
In [482]: accuracy_score(df['sent'], score_v2[:,2])
```

```
Out[482]: 0.5381526104417671
```

```
In [483]: sum(df['sent'] == score_v2[:,2])
```

```
Out[483]: 268
```

L'accuracy augmente légèrement avec cette nouvelle version, et on classifie correctement 2 exemples de plus.

1.6 Algorithme de détection - V3

Vous avez ici besoin du dictionnaire d'emoticons est disponible (fichier EmoticonLookupTable.txt disponible ici : https://clavel.wp.imt.fr/files/2018/06/Lexiques.zip).

Cet algorithme demande en entree deux listes supplementaires : - une liste d'emoticons positifs - et une liste d'emoticons negatifs

Les emoticons positifs rencontres augmentent de 1 la valeur du score positif du tweet, tandis que les emoticons negatifs augmentent de 1 la valeur du score negatif du tweet.

```
In [484]: emo = pd.read_csv('Lexiques/EmoticonLookupTable.txt', sep='\t', header=None)
```

```
In [485]: emo_pos = list(emo[emo[1] > 0][0])
          emo_neg = list(emo[emo[1] < 0][0])
```

```
In [492]: def compute_score_v3(taggedData) :

              def sort_words(taggedData) :
                  list_token = []
```

```
    for tokens in taggedData :
        intermediate_list_token = []
        for token in tokens :
            if token[1][:2] == 'JJ' or token[1][:2] == 'NN' or token[1][:2] == '
                intermediate_list_token.append(token[0])
        list_token.append(intermediate_list_token)
    return list_token

filtered_words = sort_words(taggedData)
score_tweet = []

# Total number of smileys
nb_smileys = 0

for tweet in filtered_words :

    modifier = None
    negator = None

    score = 0
    score_pos = 0
    score_neg = 0

    for token in tweet :

        try :
            if negator in negating :
                if modifier in booster[0] :
                    score_pos += swn.senti_synset(wn.synsets(token)[0].name()).ne
                    score_neg += swn.senti_synset(wn.synsets(token)[0].name()).po
                else :
                    score_pos += swn.senti_synset(wn.synsets(token)[0].name()).ne
                    score_neg += swn.senti_synset(wn.synsets(token)[0].name()).po
            else :
                if modifier in booster[0] :
                    score_pos += swn.senti_synset(wn.synsets(token)[0].name()).po
                    score_neg += swn.senti_synset(wn.synsets(token)[0].name()).ne
                else :
                    score_pos += swn.senti_synset(wn.synsets(token)[0].name()).po
                    score_neg += swn.senti_synset(wn.synsets(token)[0].name()).ne

        # We now change the except and if the token does not have a synset, we c
        except :

            if token in emo_pos :
                score_pos += 1
                nb_smileys += 1
            elif token in emo_neg :
```

```
                    score_neg += 1
                    nb_smileys += 1

            else :
                pass

            modifier = token
            negator = token


        if score_pos > score_neg :
            score_tweet.append([score_pos, score_neg, 'Positif'])

        elif score_pos == score_neg :
            score_tweet.append([score_pos, score_neg, 'Neutre'])

        else :
            score_tweet.append([score_pos, score_neg, 'Negatif'])

    print("Total number of smileys : " + str(nb_smileys))
    return np.array(score_tweet)
```

```
In [493]: score_v3 = compute_score_v3(taggedData)
```

```
Total number of smileys : 52
```

```
In [494]: accuracy_score(df['sent'], score_v3[:,2])
```

```
Out[494]: 0.5742971887550201
```

```
In [495]: sum(df['sent'] == score_v3[:,2])
```

```
Out[495]: 286
```

1.7 Algorithme de détection - V4

En analysant les sorties des algorithmes proposes precedemment, proposez votre propre algorithme d'analyse des opinions dans les tweets et les performances que vous obtenez.

```
In [516]: def compute_score_v4(taggedData, factor) :

              def sort_words(taggedData) :
                  list_token = []
                  for tokens in taggedData :
                      intermediate_list_token = []
                      for token in tokens :
                          if token[1][:2] == 'JJ' or token[1][:2] == 'NN' or token[1][:2] == '
                              intermediate_list_token.append(token[0])
```

```
        list_token.append(intermediate_list_token)
    return list_token

filtered_words = sort_words(taggedData)
score_tweet = []
nb_smileys = 0

for tweet in filtered_words :
    modifier = None
    negator = None

    score = 0
    score_pos = 0
    score_neg = 0

    for token in tweet :

        score_pos_int = 0
        score_neg_int = 0
        score_obj_int = 0

        tot = len(wn.synsets(token))

        if tot > 0 :

            # We now iterate through all the wordnets of a given token
            # And give each definition a decaying weight on the overall score (F
            for i in range(tot) :

                if negator in booster[0] :
                    if modifier in booster[0] :
                        score_pos_int += swn.senti_synset(wn.synsets(token)[i].na
                        score_neg_int += swn.senti_synset(wn.synsets(token)[i].na
                        score_obj_int += swn.senti_synset(wn.synsets(token)[i].na

                    else :
                        score_pos_int += swn.senti_synset(wn.synsets(token)[i].na
                        score_neg_int += swn.senti_synset(wn.synsets(token)[i].na
                        score_obj_int += swn.senti_synset(wn.synsets(token)[i].na
                else :
                    if modifier in booster[0] :
                        score_pos_int += swn.senti_synset(wn.synsets(token)[i].na
                        score_neg_int += swn.senti_synset(wn.synsets(token)[i].na
                        score_obj_int += swn.senti_synset(wn.synsets(token)[i].na
                    else :
                        score_pos_int += swn.senti_synset(wn.synsets(token)[i].na
                        score_neg_int += swn.senti_synset(wn.synsets(token)[i].na
                        score_obj_int += swn.senti_synset(wn.synsets(token)[i].na
```

```
                        else :
                            if token in emo_pos :
                                score_pos += 1
                                nb_smileys += 1
                            elif token in emo_neg :
                                score_neg += 1
                                nb_smileys += 1

                        # We then normalize the total score to sum to 1 for each token
                        total = score_pos_int + score_neg_int + score_obj_int

                        if total > 0 :
                            score_pos += score_pos_int / total
                            score_neg += score_neg_int / total

                        modifier = token
                        negator = token

                    # Also change thresholds by allocation more values to Neutral
                    if score_pos > score_neg + 0.1 :
                        score_tweet.append([score_pos, score_neg, 'Positif'])

                    elif score_pos < score_neg - 0.1 :
                        score_tweet.append([score_pos, score_neg, 'Negatif'])

                    else :
                        score_tweet.append([score_pos, score_neg, 'Neutre'])

                print("Number of smileys : " + str(nb_smileys))
                return np.array(score_tweet)
```

```
In [536]: score_v4 = compute_score_v4(taggedData, 0.9)
```

```
Number of smileys : 52
```

```
In [537]: accuracy_score(df['sent'], score_v4[:,2])
```

```
Out[537]: 0.642570281124498
```

```
In [529]: sum(df['sent'] == score_v4[:,2])
```

```
Out[529]: 320
```

Le résultat global est amélioré de quasiment 7% avec le seuil rajouté et le facteur de poids.