TP_Opinions

June 13, 2018

```
In [316]: import pandas as pd
          import nltk
          import numpy as np
          import re
```

1 Analyse des opinions sous twitter

```
In [317]: df = pd.read_csv('testdata.manual.2009.06.14.csv',header=None)
          df.columns = ['polarité','identifiant','date','requête','utilisateur','texte']
```

1.1 Prétraitements

Récupérer le texte associé

```
In [318]: text = df.iloc[:,-1]
```

Segmenter en tokens

```
In [319]: tokens = [t.split(' ') for t in text ]
```

Supprimer les urls

```
In [320]: regex = re.compile('http://.*')
          Tokens=[]
          for t in tokens:
              a = ''
              for w in t:
                  if not regex.match(w):
                      a = a + ' ' +  w
              Tokens.append(a)
```

```
In [321]: from nltk.tokenize.casual import TweetTokenizer
          tknz = TweetTokenizer(preserve_case=False, reduce_len=True, strip_handles=True)

          Tokens = [tknz.tokenize(mystring) for mystring in Tokens]
```

Nettoyer les caractères inhérents à la structure d'un tweet

```
In [322]: regex = re.compile('#|@')
          Tokens_2=[]
          for t in Tokens:
              a = []
              for w in t:
                  if not regex.match(w):
                      a.append(w)
              Tokens_2.append(a)
```

Corriger les abréviations et les spécificités langagières des tweets

```
In [323]: lexique = pd.read_csv('Lexiques/SlangLookupTable.txt', header = None,encoding='latin1'
          Lex = lexique.set_index(0)
```

```
In [324]: Tokens_3=[]
          for t in Tokens_2:
              a = []
              for w in t:
                  if w not in Lex.index:
                      a.append(w)
                  else:
                      a.append(Lex.loc[w].values[0])
              Tokens_3.append(a)
```

Etiquetage grammatical

```
In [325]: Tokens_4 = [nltk.pos_tag(Token) for Token in Tokens_3]
```

```
In [326]: verb=0
          for T in Tokens_4:
              for t in T:
                  if re.match('VB.',t[1]):
                      verb+=1
          print('Le nombre de mots étiquetés verbes dans le corpus est :', verb)
```

```
Le nombre de mots étiquetés verbes dans le corpus est : 836
```

1.2 Algorithme de détection v1 : appel au dictionnaire Sentiwordnet

```
In [327]: from sentiwordnet import SentiWordNetCorpusReader, SentiSynset
          from nltk.corpus import wordnet as wn
```

Récupération des mots correspondant à des adjectifs, noms, adverbes et verbes

```
In [328]: Tokens_5=[]
          for T in Tokens_4:
              tt=[]
              for t in T:
                  if re.match('VB.?|NN.?|JJ.?|VB.?|RB.?',t[1]):
                      tt.append(t[0])
              Tokens_5.append(tt)
```

Algorithme de détection v1

```
In [329]: posneg=[]
          for T in Tokens_5:
              a=0
              b=0
              for t in T:
                  syn = wn.synsets(t)
                  if syn:
                      senti = swn.senti_synset(syn[0].name())
                      try:
                          a+= senti.pos_score
                          b+= senti.neg_score
                      except AttributeError:
                          a+=0
                          b+=0
                  else:
                      a+=0
                      b+=0
              if a<b:
                  c=0
              elif a>b:
                  c=4
              else:
                  c=2
              posneg.append([a,b,c])
```

Affichage du compte rendu de classification. le nombre de tweets positifs correctement détectés avec cette version de l'algorithme est à lire dans le Recall_4

```
In [330]: from sklearn.metrics import classification_report
          print( classification_report(df.iloc[:,0].values,np.array(posneg)[:,-1]))
```

```
             precision    recall  f1-score   support

          0       0.61      0.39      0.47       177
          2       0.47      0.44      0.45       139
          4       0.46      0.64      0.54       182

avg / total       0.51      0.50      0.49       498
```

1.3 Algorithme de détection v2 : gestion de la négation et des modifieurs

```
In [331]: lexique = pd.read_csv('Lexiques/BoosterWordList.txt', header = None,encoding='latin1',
          negatif =  pd.read_csv('Lexiques/NegatingWordList.txt', header = None,encoding='latin1
```

```
In [332]: Tokens_6=[]
          for T in Tokens_4:
              tt=[]
              for t in T:
                  if re.match('VB.?|NN.?|JJ.?|VB.?|RB.?',t[1]):
                      tt.append(t[0])
              Tokens_6.append(tt)
```

```
In [333]: posneg=[]
          neginpos = 0
          for T in Tokens_6:
              a=0
              b=0
              tavt=''
              neginpos1 = 0
              for t in T:

                  syn = wn.synsets(t)
                  if syn:
                      senti = swn.senti_synset(syn[0].name())
                      try:
                          # Ajout des mots boosts
                          if tavt in list(lexique.iloc[:,0]):
                              a+= 2*senti.pos_score
                              b+= 2*senti.neg_score
                          # Ajout des négations
                          elif tavt in list(negatif.iloc[:,0]):
                              a+= senti.neg_score
                              b+= senti.pos_score
                              neginpos1 += 1
                          else:
                              b+= senti.neg_score
                              a+= senti.pos_score
                      except AttributeError:
                          a+=0
                          b+=0
                  else:
                      a+=0
                      b+=0
                  tavt=t
```

```
            if a<b:
                c=0
            elif a>b:
                c=4
                neginpos+=neginpos1
            else:
                c=2
            posneg.append([a,b,c])
```

```
In [334]: print( classification_report(df.iloc[:,0].values,np.array(posneg)[:,-1]))
          print('Nombre nombre de termes négatifs contenus dans les tweets positifs :',neginpos)
```

```
             precision    recall  f1-score   support

          0       0.62      0.42      0.51       177
          2       0.48      0.44      0.46       139
          4       0.47      0.64      0.54       182

avg / total       0.53      0.51      0.51       498

Nombre nombre de termes négatifs contenus dans les tweets positifs : 9
```

1.4 Algorithme de détection v3 : gestion des emoticons

```
In [335]: emot =  pd.read_csv('Lexiques/EmoticonLookupTable.txt', header = None,encoding='latin1
          emot = emot.set_index(0)
```

```
In [336]: posneg=[]
          n_emot=0
          for T in Tokens_6:
              a=0
              b=0
              tavt=''
              for t in T:
                  syn = wn.synsets(t)
                  if syn:
                      senti = swn.senti_synset(syn[0].name())
                      try:
                          if tavt in list(lexique.iloc[:,0]):
                              a+= 2*senti.pos_score
                              b+= 2*senti.neg_score
                          elif tavt in list(negatif.iloc[:,0]):
                              a+= senti.neg_score
                              b+= senti.pos_score
                          else:
                              b+= senti.neg_score
                              a+= senti.pos_score
                      except AttributeError:
```

```
                    a+=0
                    b+=0
                # Ajout des émoticones
                elif t in list(emot.index):
                    a+= emot.loc[t].values[0]
                    b = b
                    n_emot+=1
                else:
                    a+=0
                    b+=0
                tavt=t
            if a<b:
                c=0
            elif a>b:
                c=4
            else:
                c=2
            posneg.append([a,b,c])
```

```
In [337]: print(classification_report(df.iloc[:,0].values,np.array(posneg)[:,-1]))
          print('Le nombre d\'emoticones présents dans le corpus est :', n_emot)
```

```
             precision    recall  f1-score   support

          0       0.66      0.48      0.56       177
          2       0.50      0.44      0.47       139
          4       0.50      0.69      0.58       182

avg / total       0.56      0.54      0.54       498

Le nombre d'emoticones présents dans le corpus est : 57
```

1.5 Votre version : v4

Améliorations: * Ajout d'un seuil pour la différence du score positif et du score négatif du tweet en deça duquel le tweet est considéré comme neutre. En effet en regardant les erreurs sur neutres, il est apparu que dés qu'il y a une interrogation le tweet est classifié comme neutre. * Tous les mots homonymes sont pris en comptes pour le calcul du score positif et négatif, en les sommant avec des poids w = 1/ordre d'apparition dans les synonimes. Il apparaît en effet que souvent les mots utilisés sur twitter sont utilisés avec le second sens ou plus.

```
In [338]: seuil=0.2
          posneg=[]
          n_emot=0
          for T in Tokens_6:
              a=0
              b=0
              tavt=''
```

```
            for t in T:
                syn = wn.synsets(t)
                if syn:
                    pos=0
                    neg=0
                    # Calcul des scores avec tous les synonimes (somme pondérées)
                    for i,sy in enumerate(syn, 1):
                        if swn.senti_synset(sy.name()) != None and t in sy.name():
                            pos +=(1/i)*swn.senti_synset(sy.name()).pos_score
                            neg +=(1/i)*swn.senti_synset(sy.name()).neg_score
                        pos = pos
                        neg = neg
                    if tavt in list(lexique.iloc[:,0]):
                        a+= 2*pos
                        b+= 2*neg
                    elif tavt in list(negatif.iloc[:,0]):
                        a+= neg
                        b+= pos
                    else:
                        b+= neg
                        a+= pos

                elif t in list(emot.index):
                        a+= 2*emot.loc[t].values[0]
                        b = b
                        n_emot+=1
                else:
                    a+=0
                    b+=0
                tavt=t

            # Calcul de la nouvelle attribution avec un seuil
            if abs(a-b) > seuil:
                if a<b:
                    c=0
                else:
                    c=4
            else:
                c=2
            posneg.append([a,b,c])
```

In [339]:
```
print(classification_report(df.iloc[:,0].values,np.array(posneg)[:,-1]))
```

```
             precision    recall  f1-score   support

          0       0.73      0.44      0.55       177
          2       0.48      0.62      0.54       139
          4       0.55      0.65      0.60       182
```

```
avg / total       0.60      0.56      0.56       498
```

1.6 Concusion:

F1 score

- Algo v1: 0.49
- Algo v2: 0.51
- Algo v3: 0.54
- ALgo v4: 0.56

On a amélioré l'algotithme au fur et à mesure des briques ajoutées. Les possibles améliorations sont en enrichissants les différentes bases de données, ou alors en regardant le problème avec des n-uplets de mots.