Natural Language Processing pour la classification de texte avec NLTK et Scikit-learn

Présenté par yassine NAJMI

Dans le projet, Premiers pas avec le traitement du langage naturel en Python, nous avons appris les bases de la création de jetons, de l'étiquetage d'une partie du discours, de la dérivation, de la segmentation et de la reconnaissance d'entités nommées; De plus, nous avons plongé dans l'apprentissage automatique et la classification de texte à l'aide d'un simple classificateur de vecteur de support et d'un ensemble de données de critiques de films positives et négatives.

Dans ce tutoriel, nous allons développer sur cette base et d'explorer différentes façons d'améliorer nos résultats de la classification texte. Nous couvrirons et utiliserons:

- Expressions régulières
- Ingénierie des fonctionnalités
- Plusieurs classificateurs scikit-learn
- Méthodes d'ensemble

1. Importer les bibliothèques nécessaires

Pour vous assurer que les bibliothèques nécessaires sont correctement installées et à jour, imprimez les numéros de version de chaque bibliothèque. Cela améliorera également la reproductibilité de notre projet.

In [14]:
```
import sys
import nltk
import sklearn
import pandas
import numpy
```

```
print('Python: {}'.format(sys.version))
print('NLTK: {}'.format(nltk.__version__))
print('Scikit-learn: {}'.format(sklearn.__version__))
print('Pandas: {}'.format(pandas.__version__))
print('Numpy: {}'.format(numpy.__version__))
```

```
Python: 2.7.16 |Anaconda, Inc.| (default, Mar 14 2019, 15:42:17) [MSC
v.1500 64 bit (AMD64)]
NLTK: 3.4.5
Scikit-learn: 0.20.3
Pandas: 0.24.2
Numpy: 1.16.5
```

1. Charger le Dataset

Maintenant que nous nous sommes assurés que nos bibliothèques sont installées correctement, nous allons charger l'ensemble de données en tant que Pandas dataframe. De plus, extrayons des informations utiles telles que les informations de colonne et les distributions de classe.

L'ensemble de données que nous utiliserons provient du référentiel UCI Machine Learning. Il contient plus de 5000 messages SMS étiquetés qui ont été collectés pour la recherche de spam sur les téléphones mobiles. Il peut être téléchargé à partir de l'URL suivante:

https://archive.ics.uci.edu/ml/datasets/sms+spam+collection

In [ ]:
```
import pandas as pd
import numpy as np

# charger l'ensemble de données des messages SMS
df = pd.read_table('DataSpamSms', header=None, encoding='utf-8')
```

In [9]:
```
# imprimer des informations utiles sur l'ensemble de données
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
```

```
Data columns (total 2 columns):
0    5572 non-null object
1    5572 non-null object
dtypes: object(2)
memory usage: 87.1+ KB
None
      0                                                  1
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
```

In [10]:
```
# vérifier la distribution des classes
classes = df[0]
print(classes.value_counts())
```

```
ham     4825
spam     747
Name: 0, dtype: int64
```

2. Prétraiter les données

Le prétraitement des données est une étape essentielle du processus de langage naturel. Dans les cellules suivantes, nous convertirons nos étiquettes de classe en valeurs binaires à l'aide de LabelEncoder de sklearn, remplacerons les adresses e-mail, les URL, les numéros de téléphone et d'autres symboles à l'aide d'expressions régulières, supprimerons les mots vides et extraire les racines des mots.

In [11]:
```
from sklearn.preprocessing import LabelEncoder

# convertir les étiquettes de classe en valeurs binaires, 0 = ham and 1
 = spam
encoder = LabelEncoder()
Y = encoder.fit_transform(classes)

print(Y[:10])
```

```
[0 0 1 0 0 1 0 0 1 1]
```

In [18]:
```
# stocker les données du message SMS
text_messages = df[1]
print(text_messages[:10])
```

```
0    Go until jurong point, crazy.. Available only ...
1                        Ok lar... Joking wif u oni...
2    Free entry in 2 a wkly comp to win FA Cup fina...
3    U dun say so early hor... U c already then say...
4    Nah I don't think he goes to usf, he lives aro...
5    FreeMsg Hey there darling it's been 3 week's n...
6    Even my brother is not like to speak with me. ...
7    As per your request 'Melle Melle (Oru Minnamin...
8    WINNER!! As a valued network customer you have...
9    Had your mobile 11 months or more? U R entitle...
Name: 1, dtype: object
```

2.1 Expressions régulières

Some common regular expression metacharacters - copied from wikipedia

^ Matches the starting position within the string. In line-based tools, it matches the starting position of any line.

. Matches any single character (many applications exclude newlines, and exactly which characters are considered newlines is flavor-, character-encoding-, and platform-specific, but it is safe to assume that the line feed character is included). Within POSIX bracket expressions, the dot character matches a literal dot. For example, a.c matches "abc", etc., but [a.c] matches only "a", ".", or "c".

[ ] A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] specifies a range which matches any lowercase letter from "a" to "z". These forms can be mixed: [abcx-z] matches "a", "b", "c", "x", "y", or "z", as does [a-cx-z]. The - character is treated as a literal character if it is the last or the first (after the ^,

if present) character within the brackets: [abc-], [-abc]. Note that backslash escapes are not allowed. The] character can be included in a bracket expression if it is the first (after the ^) character: []abc].

[^] Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z] matches any single character that is not a lowercase letter from "a" to "z". Likewise, literal characters and ranges can be mixed.

$ Matches the ending position of the string or the position just before a string-ending newline. In line-based tools, it matches the ending position of any line.

( ) Defines a marked subexpression. The string matched within the parentheses can be recalled later (see the next entry, \n). A marked subexpression is also called a block or capturing group. BRE mode requires ( ).

\n Matches what the nth marked subexpression matched, where n is a digit from 1 to 9. This construct is vaguely defined in the POSIX.2 standard. Some tools allow referencing more than nine capturing groups.

* Matches the preceding element zero or more times. For example, ab*c matches "ac", "abc", "abbbc", etc. [xyz]* matches "", "x", "y", "z", "zx", "zyx", "xyzzy", and so on. (ab)* matches "", "ab", "abab", "ababab", and so on.

{m,n} Matches the preceding element at least m and not more than n times. For example, a{3,5} matches only "aaa", "aaaa", and "aaaaa". This is not found in a few older instances of regexes. BRE mode requires {m,n}.

In [50]:
```
# utiliser des expressions régulières pour remplacer les adresses e-mai
l, les URL, les numéros de téléphone et d'autres numéros

# Remplacez les adresses e-mail par 'email'
processed = text_messages.str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                      'emailaddress')

# Remplacer les URL par 'webaddress'
processed = processed.str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]
```

```
{2,3}(/\S*)?$',
                                    'webaddress')

# Remplacez les symboles d'argent par 'moneysymb'
processed = processed.str.replace(r'£|\$', 'moneysymb')

# Remplacer les numéros de téléphone à 10 chiffres (les formats incluen
t les parenthèses, les espaces, aucun espace, les tirets) par 'phonenum
ber'
processed = processed.str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?
[\d]{4}$',
                                    'phonenumbr')

# Remplacer les nombres par 'numbr'
processed = processed.str.replace(r'\d+(\.\d+)?', 'numbr')
```

In [49]:
```
# Supprimer la ponctuation
processed = processed.str.replace(r'[^\w\d\s]', ' ')

# Remplacer les espaces entre les termes par un seul espace
processed = processed.str.replace(r'\s+', ' ')

# Supprimer les espaces de début et de fin
processed = processed.str.replace(r'^\s+|\s+?$', '')
```

In [48]:
```
# changer les mots en minuscules - Bonjour, BONJOUR, bonjour sont tous
 le même mot
processed = processed.str.lower()
print(processed)
```

```
0       go jurong point crazi avail bugi n great world...
1                                   ok lar joke wif u oni
2       free entri numbr wkli comp win fa cup final tk...
3                    u dun say earli hor u c alreadi say
4                   nah think goe usf live around though
5       freemsg hey darl numbr week word back like fun...
6           even brother like speak treat like aid patent
7       per request mell mell oru minnaminungint nurun...
8       winner valu network custom select receivea num...
```

```
9       mobil numbr month u r entitl updat latest colo...
10      gonna home soon want talk stuff anymor tonight...
11      six chanc win cash numbr numbr numbr pound txt...
12      urgent numbr week free membership numbr numbr ...
13      search right word thank breather promi wont ta...
14                                              date sunday
15      xxxmobilemovieclub use credit click wap link n...
16                                               oh k watch
17      eh u rememb numbr spell name ye v naughti make...
18                            fine way u feel way gota b
19      england v macedonia dont miss goal team news t...
20                                    seriou spell name
21                         go tri numbr month ha ha joke
22                          pay first lar da stock comin
23      aft finish lunch go str lor ard numbr smth lor...
24                    fffffffff alright way meet sooner
25      forc eat slice realli hungri tho suck mark get...
26                                    lol alway convinc
27      catch bu fri egg make tea eat mom left dinner ...
28                        back amp pack car let know room
29                   ahhh work vagu rememb feel like lol
                              ...
5542                          armand say get ass epsilon
5543                  u still havent got urself jacket ah
5544    take derek amp taylor walmart back time done l...
5545                               hi durban still number
5546                               ic lotta childporn car
5547    contract mobil numbr mnth latest motorola noki...
5548                                          tri weekend v
5549    know wot peopl wear shirt jumper hat belt know...
5550                                    cool time think get
5551                           wen get spiritu deep great
5552    safe trip nigeria wish happi soon compani shar...
5553                                 hahaha use brain dear
5554    well keep mind got enough ga one round trip ba...
5555    yeh indian nice tho kane bit shud go numbr dri...
5556                             ye u text pshew miss much
5557    meant calcul lt gt unit lt gt school realli ex...
5558                                       sorri call later
5559                        next lt gt hour imma flip shit
```

```
5560                                anyth lor juz us lor
5561                  get dump heap mom decid come low bore
5562    ok lor soni ericsson salesman ask shuhui say q...
5563                                ard numbr like dat lor
5564                     wait til least wednesday see get
5565                                                huh lei
5566    remind onumbr get numbr pound free call credit...
5567    numbrnd time tri numbr contact u u numbr pound...
5568                                b go esplanad fr home
5569                                    piti mood suggest
5570    guy bitch act like interest buy someth el next...
5571                                         rofl true name
Name: 1, Length: 5572, dtype: object
```

In [ ]:
```
import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords

# supprimer les mots vides des messages texte

stop_words = set(stopwords.words('english'))

processed = processed.apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))
```

In [45]:
```
# Retirer les tiges de mots à l'aide de Porter Stemmer
ps = nltk.PorterStemmer()

processed = processed.apply(lambda x: ' '.join(
    ps.stem(term) for term in x.split()))
```

3.Générer des fonctionnalités

L'ingénierie des fonctionnalités consiste à utiliser la connaissance du domaine des données pour créer des fonctionnalités pour les algorithmes d'apprentissage automatique. Dans ce projet, les

mots de chaque message texte seront nos fonctionnalités. Pour cela, il sera nécessaire de tokeniser chaque mot. Nous utiliserons les 1500 mots les plus courants comme fonctionnalités.

In [ ]:
```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize

# create bag-of-words
all_words = []

for message in processed:
    words = word_tokenize(message)
    for w in words:
        all_words.append(w)

all_words = nltk.FreqDist(all_words)
```

In [43]:
```
# imprimer le nombre total de mots et les 15 mots les plus communs
print('Number of words: {}'.format(len(all_words)))
print('Most common words: {}'.format(all_words.most_common(15)))
```

```
Number of words: 6562
Most common words: [(u'numbr', 2961), (u'u', 1207), (u'call', 679),
(u'go', 456), (u'get', 451), (u'ur', 391), (u'gt', 318), (u'lt', 316),
(u'come', 304), (u'ok', 293), (u'free', 284), (u'day', 276), (u'know',
275), (u'love', 266), (u'like', 261)]
```

In [42]:
```
# utiliser les 1500 mots les plus communs comme caractéristiques
word_features = list(all_words.keys())[:1500]
```

In [41]:
```
# La fonction find_features déterminera les caractéristiques de 1500 mo
ts contenues dans la revue
def find_features(message):
    words = word_tokenize(message)
    features = {}
    for word in word_features:
        features[word] = (word in words)
```

```
    return features

# Voyons un exemple!
features = find_features(processed[0])
for key, value in features.items():
    if value == True:
        print key
```

```
avail
buffet
world
great
```

In [39]:
```
# Maintenant, nous allons le faire pour tous les messages
messages = zip(processed, Y)

# définir une graine pour la reproductibilité
seed = 1
np.random.seed = seed
np.random.shuffle(messages)

# appeler la fonction find_features pour chaque message SMS
featuresets = [(find_features(text), label) for (text, label) in messag
es]
```

In [38]:
```
# nous pouvons diviser les ensembles de fonctionnalités en ensembles de
 données d'entraînement et de test à l'aide de sklearn
from sklearn import model_selection

# diviser les données en ensembles de données d'entraînement et de test
training, testing = model_selection.train_test_split(featuresets, test_
size = 0.25, random_state=seed)
```

In [37]:
```
print(len(training))
print(len(testing))
```

```
4179
```

1393

4. Classificateurs Scikit-Learn avec NLTK

Maintenant que nous avons notre jeu de données, nous pouvons commencer à créer des algorithmes! Commençons par un simple classificateur de vecteur de support linéaire, puis développons d'autres algorithmes. Nous devrons importer chaque algorithme que nous prévoyons d'utiliser depuis sklearn. Nous devons également importer certaines mesures de performances, telles que precision_score et classification_report.

In [51]:
```
# Nous pouvons utiliser les algorithmes sklearn dans NLTK
from nltk.classify.scikitlearn import SklearnClassifier
from sklearn.svm import SVC

model = SklearnClassifier(SVC(kernel = 'linear'))

# faire Entrainer le modèle sur les données d'entraînement
model.train(training)

# et testez sur l'ensemble de données de test!
accuracy = nltk.classify.accuracy(model, testing)*100
print("SVC Accuracy: {}".format(accuracy))
```

```
SVC Accuracy: 95.6927494616
```

In [52]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, conf
usion_matrix

# Définir des modèles à former
names = ["K Nearest Neighbors", "Decision Tree", "Random Forest", "Logi
stic Regression", "SGD Classifier",
```

```
         "Naive Bayes", "SVM Linear"]

classifiers = [
    KNeighborsClassifier(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    LogisticRegression(),
    SGDClassifier(max_iter = 100),
    MultinomialNB(),
    SVC(kernel = 'linear')
]

models = zip(names, classifiers)

for name, model in models:
    nltk_model = SklearnClassifier(model)
    nltk_model.train(training)
    accuracy = nltk.classify.accuracy(nltk_model, testing)*100
    print("{} Accuracy: {}".format(name, accuracy))
```

```
K Nearest Neighbors Accuracy: 92.9648241206
Decision Tree Accuracy: 94.7595118449
Random Forest Accuracy: 95.1184493898
Logistic Regression Accuracy: 95.4773869347
SGD Classifier Accuracy: 95.8363244795
Naive Bayes Accuracy: 95.2620244078
SVM Linear Accuracy: 95.6927494616
```

In [30]:
```
# Ensemble des méthodes - classificateur de vote
from sklearn.ensemble import VotingClassifier

names = ["K Nearest Neighbors", "Decision Tree", "Random Forest", "Logi
stic Regression", "SGD Classifier",
         "Naive Bayes", "SVM Linear"]

classifiers = [
    KNeighborsClassifier(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
```

```
    LogisticRegression(),
    SGDClassifier(max_iter = 100),
    MultinomialNB(),
    SVC(kernel = 'linear')
]

models = zip(names, classifiers)

nltk_ensemble = SklearnClassifier(VotingClassifier(estimators = models,
 voting = 'hard', n_jobs = -1))
nltk_ensemble.train(training)
accuracy = nltk.classify.accuracy(nltk_model, testing)*100
print("Voting Classifier: Accuracy: {}".format(accuracy))
```

```
Voting Classifier: Accuracy: 94.4005743001
```

In [27]:
```
# faire une prédiction d'étiquette de classe pour l'ensemble de test
txt_features, labels = zip(*testing)

prediction = nltk_ensemble.classify_many(txt_features)
```

In [28]:
```
# imprimer une matrice de confusion et un rapport de classification
print(classification_report(labels, prediction))

pd.DataFrame(
    confusion_matrix(labels, prediction),
    index = [['actual', 'actual'], ['ham', 'spam']],
    columns = [['predicted', 'predicted'], ['ham', 'spam']])
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      1199
           1       0.90      0.73      0.81       194

   micro avg       0.95      0.95      0.95      1393
   macro avg       0.93      0.86      0.89      1393
weighted avg       0.95      0.95      0.95      1393
```

Out[28]:

		predicted	
		ham	spam
actual	ham	1184	15
	spam	53	141

Merci Mon Prof Mr. Abdelhak Mahmoudi !

In [ ]: