WILLARD

Zeste de savoir

Deviner qui est l'auteur d'un message grâce au Machine Learning

16 mai 2019

Table des matières

Je sais qui je suis, mais quelques uns m'ont déjà posé cette question : "Qui es-tu ?" et ben, c'est dans ce billet que je souhaite répondre.

Je me doute bien que si je me contente de dire, je suis XXX, on risque de me traiter de menteur, alors j'aimerai apporter la preuve que je suis qui je suis avec l'aide d'algorithmes de *machine learning*.

Sans plus attendre, raisonnons ensemble !

1. Allons à la recherche d'informations sur moi

Tout ce que l'on sait sur moi se trouve sur Zeste de Savoir. Et, croyez le ou pas, tout ce que l'on dit sur internet, même derrière un pseudonyme, permet de nous identifier. Sur internet on peu changer notre adresse email, notre pseudonyme, notre adresse ip, notre FAI, mais quelque chose de très compliqué à changer, c'est notre façon de parler/d'écrire. Et c'est comme ça que je vais vous prouver que je suis ce que je suis.

Nous allons nous concentrer sur tout ce que j'ai pu dire sur les forums de Zeste de Savoir. Pour cela, j'ai écrit un script qui me permet de collecter l'ensemble de mes messages sur le site.

Nous allons partir de l'hypothèse suivante : **le jour ou j'ai écris cet article , j'étais infiltré dans le staff**. Ce qui signifie donc que la liste des suspects est réduite aux membres du staff ce jour là.

C'est la raison pour laquelle on va aussi télécharger l'ensemble des messages des membres du staff à cette époque ainsi que des messages de ceux dont on est certain que ce n'est pas moi (des contres exemples quoi).

En résumé, voici les données que je vous propose de télécharger pour tester le code donné par la suite.

Lien de téléchargement

Le contenu de l'archive est le suivant :

fichier/dossier	Description
staff.csv	Fichier csv contenant la liste des membres du staff à l'époque ou le fameux article a été publié.
data.csv	Liste des phrases que l'on trouve sur le forum. La colonne target est affectée à 1 lorsque c'est une phrase écrite par willard et à 0 si la phrase n'a pas été écrite par lui.
sentences	Répertoire avec une liste de fichiers csv contenant des phrases de plusieurs membres, téléchargées sur le forum Zeste de Savoir
build_model.py	Script de construction du modèle
guess.py	Script qui permet de deviner qui est willard.

Une fois les fichiers téléchargés, vous devez obtenir l'arborescence suivante :

Contenu masqué n°1

2. Construisons notre modèle de prédiction

2.1. Installation des libs externes

Pour nos travaux nous travaillons avec python 3.6. Nous aurons besoin d'installer les libs python suivantes :

```
pip install pandas
pip install scikit-learn
```

2.2. Séparer nos données (train et test)

Pour construire notre modèle, nous allons commencer par séparer nos données en deux catégories. Les données d'entrainement (celles qui vont servir à entraîner notre modèle) et les données de validation (celles qui vont permettre de valider que notre modèle est juste).

L'opération se fait grâce au code suivant :

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split

```

```
# Import des données
data = pd.read_csv('data.csv')
CURRENT_DIR = os.path.dirname(os.path.realpath(__file__))

texts = data.sentence
target = data.target

# definir la colonne target comme étant de type booléen
target = target.astype('bool')

# split des données en deux catégories
sentences_train, sentences_test, y_train, y_test =
    train_test_split(texts.values, target.values, test_size=0.3,
    random_state=1000)
```

Nous venons d'effectuer une séparation dont 30% de nos données initiales serviront pour l'apprentissage.

2.3. La vectorisation du texte

Notre variable principale (la colonne X) contient surtout du texte. Le problème c'est que le texte ne se donne pas comme cela aux algorithmes, il faut au préalable le vectoriser. C'est à dire, transformer une phrase en une liste de nombre. On peut le voir comme la version mathématique de la phrase.

Il existe plusieurs types de vectorisation, et chacun s'utilise en fonction des cas que l'on veut traiter. Pour notre cas, nous utiliserons la vectorisation par comptage de mots.

En python ça donnera ceci :

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
vectorizer.fit(sentences_train)

# vectorise les phrases d'entrainement et de test avec
    CountVectorizer
X_train = vectorizer.transform(sentences_train)
X_test  = vectorizer.transform(sentences_test)
```

Maintenant que l'on dispose de la forme vectorisée des phrases, on enregistre le modèle de vectorisation dans un fichier (on le réutilisera quand il faudra effectuer les prédictions).

```
import pickle
```

```
2 pickle.dump(vectorizer, open(os.path.join(CURRENT_DIR,
      "vectorize.data"), 'wb'))
```

2.4. L'apprentissage du modèle

Nos phrases sont vectorisées, cela signifie que l'on peut les passer à un algorithme. Mais la question principale qui se pose est :

? Quel est l'algorithme dois-je utiliser ?

Hey oui, il y en a tellement des algorithmes, mais on sait néanmoins choisir les algorithmes en fonction du type de nos variables. Par exemple observons ce que nous avons :

— Nos variables sont des vecteurs (oui, on vient de les calculer)
— On essaye de prédire une cible de type booléenne (willard or not willard ?)

Vous l'avez deviné, on est tout a fait dans le cas à traiter par une régression logistique .

— **Régression** parce qu'on essaye de prédire quelque chose, on parle aussi d'expliquer un résultat en fonction d'une ou plusieurs variables.
— **Logistique** parce que la variable que l'on essaye de prédire n'est pas quantitative, mais qualitative (booléenne dans notre cas).

C'est parti pour le code d'apprentissage.

```
1 from sklearn.linear_model import LogisticRegression
2
3 model = LogisticRegression()
4 # l'apprentissage se fait bien sur la variable vectorisée de
      l'entrainement
5 model.fit(X_train, y_train)
6 # on calcule la précision du modèle sur les données de validation
7 score = model.score(X_test, y_test)
8
9 print("Precision:", score)
```

La précision renvoyée est :

```
1 Precision: 0.9800575263662512
```

On a donc ici une précision de 98% avec notre modèle d'apprentissage. Autant dire que c'est formidable. On saura, avec ce modèle prédire avec 98% de chances qu'une phrase a été écrite par moi. Fantastique non ?

Vite, vite ! Enregistrons notre modèle.

```
import pickle

pickle.dump(model, open(os.path.join(CURRENT_DIR,"model.data"),
    'wb'))
```

Vous pouvez tout simplement lancer la commande `python build_model.py` qui résume ce que l'on a vu dans cette section.
Deux fichiers seront crées, `model.data` et `vectorize.data`.

3. A la recherche de l'infiltré

L'infiltré est un membre de l'ancien staff qui écrit de la même manière que moi. On va donc chercher dans la liste des messages des membres de l'ancien staff, lesquels correspondent à des phrases de Willard.

Le code qui permet de le calculer est le suivant :

```
# guess.py
import os
import csv
import pickle
import pandas as pd

CURRENT_DIR = os.path.dirname(os.path.realpath(__file__))
# on charge le modèle de vectorisation
vectorizer =
    pickle.load(open(os.path.join(CURRENT_DIR,"vectorize.data"),
    'rb'))
# on charge le modèle d'apprentissage
model = pickle.load(open(os.path.join(CURRENT_DIR,"model.data"),
    'rb'))
res = []

def get_ids_from_file(filepath):
    """
    Cette fonction renvoi la liste des id dans un fichier à deux colonnes "id;u
    """
    ids = []
    with open(filepath, 'r') as csvfile:
        readercsv = csv.reader(csvfile, delimiter=',',
            quotechar='"')
```

```
21          next(readercsv)
22          for row in readercsv:
23              ids.append(row[0])
24
25      return ids
26
27  def get_username_from_id(filepath, id):
28      """
29      Cette fonction renvoi le nom d'utilisateur correspondant à un id
30      """
31      with open(filepath, 'r') as csvfile:
32          readercsv = csv.reader(csvfile, delimiter=',',
                quotechar='"')
33          next(readercsv)
34          for row in readercsv:
35              if row[0] == id:
36                return row[1]
37
38      return None
39
40  def predict_from_sent(x):
41      """
42      Cette fonction renvoi True si la phrase est de willard.
43      """
44      p_value = model.predict(vectorizer.transform([x]))
45
46      return p_value[0]
47
48  # on recupère la liste des ids des staff de l'époque
49  ids = get_ids_from_file("staff.csv")
50  for id in ids:
51      data = pd.read_csv('sentences/sentences_{}.csv'.format(id),
            header=None)
52      data["is_willard"] = data[1].map(predict_from_sent)
53      guilty = data.loc[data['is_willard'] == True]
54      res.append({"id": get_username_from_id("staff.csv", id),
            "guilty": len(guilty.index)})
55
56  print(sorted(res, key = lambda i: i['guilty'], reverse=True))
```

i Vous pouvez tout simplement lancer la commande `python guess.py`.

Si vous lancer le code chez vous (après avoir construit le modèle), vous pouvez retrouver qui sur ce forum, s'amuse à parler comme moi.

Pour les plus paresseux (mais je n'en dirais pas plus), sachez que le coupable se trouve parmi les cinq membres suivants (il s'agit des cinq membres dont les messages ressemblent furieusement à ma façon d'écrire) :

Contenu masqué

— Andr0
— Arius
— firm1
— Kje
— ShigeruM

Bon courage.

L'objectif de ce billet était de montrer que, plus on poste sur les forums, les réseaux sociaux, plus les algorithmes arriveront à nous identifier.

Maintenant que vous savez qui je suis, une question éthique se pose

? Est-ce que je peux être sanctionné uniquement parce qu'un algorithme l'a décidé ? Souhaitons nous faire confiance aux machines pour prendre des décisions importantes ?

Je vous laissez réfléchir !

Contenu masqué

Contenu masqué n°1

```
.
├── build_model.py
├── data.csv
├── guess.py
├── sentences
│   ├── sentences_10199.csv
│   ├── sentences_101.csv
│   ├── sentences_110.csv
│   ├── sentences_1309.csv
│   ├── sentences_138.csv
│   ├── sentences_141.csv
│   ├── sentences_143.csv
│   ├── sentences_155.csv
│   ├── sentences_2315.csv
│   ├── sentences_241.csv
│   ├── sentences_243.csv
│   ├── sentences_244.csv
│   ├── sentences_267.csv
│   ├── sentences_276.csv
│   ├── sentences_288.csv
```

Contenu masqué

```
21 │   │   ├── sentences_2.csv
22 │   │   ├── sentences_326.csv
23 │   │   ├── sentences_3349.csv
24 │   │   ├── sentences_379.csv
25 │   │   ├── sentences_542.csv
26 │   │   ├── sentences_5500.csv
27 │   │   ├── sentences_588.csv
28 │   │   ├── sentences_615.csv
29 │   │   ├── sentences_65.csv
30 │   │   ├── sentences_66.csv
31 │   │   ├── sentences_67.csv
32 │   │   ├── sentences_68.csv
33 │   │   ├── sentences_69.csv
34 │   │   ├── sentences_72.csv
35 │   │   ├── sentences_73.csv
36 │   │   ├── sentences_747.csv
37 │   │   ├── sentences_74.csv
38 │   │   ├── sentences_80.csv
39 │   │   ├── sentences_819.csv
40 │   │   ├── sentences_82.csv
41 │   │   ├── sentences_90.csv
42 │   │   ├── sentences_92.csv
43 │   │   └── sentences_94.csv
44 │   └── staff.csv
```

Retourner au texte.