IFT3335 : Rapport du TP2

Par : Mike Useni et Soumaila Keita

Introduction

Ce travail est une étude de performance des algorithmes de classification appliquée au problème

Désambiguïsation des sens des mots. L'algorithme étudie sont : le Naïve Bayes, Arbre de décision et MultiLayerPerceptron. Les algorithmes sont implémentés grace à la librairie de python SKLearn.

1.Préparation des données

Les donnes sont à récupérer dans le fichier « interest.acl94.txt » qui nous ai fournie et qui contient un corpus de [mot/catégorie]. Il nous est fourni aussi une liste de stop Word de la langue anglaise dans le fichier « stoplist-english.txt ».

1.1. Traitement et extraction des données

Une foi les données chargés, ils sont traités comme suite :

- On retire du corpus les stop Word de la liste « stoplist-english.txt » préalablement convertie en tableau, grace à la fonction « deleteStop () », afin de ne conserver que les mots utiles.
- On a ensuite **tokenize** les mots grace à la fonction « tokenisation () » afin d'obtenir un tableau 2D des mots plus faciles à utilise.
- On effectue en suite une lemmatisation grace à la fonction « lemmatisation () » qui utilise la fonction « lancasterStemmer » de la librairie sklearn. La lemmatisation permet d'obtenir la racine des mots afin de n'ai pas tenir compte des genres, des nombres et de la conjugaison.

Pour l'extraction des données, on va récupérer uniquement les deux mots avant et les deux après le mot « interest » dons on veut connaître le sens, ainsi que leurs catégories.

1.2. Transformation des données et Split

On va séparer les colonnes X des Y puis transformer les données obtenues en données numérique pour les rendre utilisable. Pour les Y on va attribuer une valeur différente à chaque sens différent. Pour se faire on va utiliser la fonction « LabelEncoder » de la librairie « sklearn ». Pour les X ont va faire de chaque mot et chaque catégorie conserver, un attribué, à l'aide de la transformation « OneHotEncoder » prévu dans la librairie « sklearn ».

On va diviser les X et Y en set d'entraînement et set de test (X_train, X_test, Y_train et Y_test) avec une partition de 70% pour train et 30% pour le test. Le split se fait à l'aide de la transformation « train_test_split » de sklearn. Les données sont désormais prêtes à l'usager pour la classification.

2. Analyse

Les fonctions de sklearn utilisé sont les suivants :

Pour le bayes naive

```
from sklearn.naive_bayes import MultinomialNB
```

Pour les arbre de décision

```
from sklearn.tree import DecisionTreeClassifier
```

pour le réseau de neurones

```
from sklearn.neural_network import MLPClassifier
```

pour le réseau de neurones ,on va analyser 3 configuration différentes :

```
random_state=1, max_iter=300
random_state=1, max_iter=300, hidden_layer_sizes=50
random_state=1, max_iter=300, hidden_layer_sizes=200
```

cad avec 100, 50 et 200 neurones et 300 itérations.

2.1. Comparaison

On passe nos données aux différentes fonctions de classification et on obtient les résultats suivants sur 4 exécutions :

Algorithme	Itération 1 Taux de réussite	Itération 2 Taux de réussite	Itération 3 Taux de réussite	Itération 4 Taux de réussite
bayes naive	73.6%	76.4%	77.0%	77.3%
arbre de décision	77.9%	82.0%	80.1%	78.9%
réseau de neurones (100)	83.5%	85.6%	86.8%	85.5%
réseau de neurones (50)	82.9%	85.9%	85.9%	85.9%
réseau de neurones (200)	82.6%	85.6%	86.5%	86.1%

On peut déduire de ce tableau d'exécution le tableau résumé suivant :

Algorithme	Max	Moy	Variation
bayes naive	77.3%	76.0%	3.7%
arbre de décision	82.0%	79.7%	4.1%
réseau de neurones (100)	86.8%	85.35%	3.3%
réseau de neurones (50)	85.9%	85.15%	3%
réseau de neurones (200)	86.5%	85.2%	3.9%

On remarque que les algorithmes les plus performant sont les **réseaux de neurones** avec des taux de réussite maximum de 86.8%, 85.9% et 86.5%. la meilleure performance étant pour le **réseau de neurones (100)** qui a 100 neurones mais avec des moyennes toute au tour de 85%**,** on ne peut pas conclure que le nombre de neurone affecte la performance ici mais avec des variations qui passe de 3% pour 50 neurones a 3.3% pour 100 neurones et 3.9% pour 200 neurones, on peut remarquer une augmentation de l'instabilité de l'algorithme plus le nombre de neurones augmente.

2.2. Améliorations possibles

Il possible d'améliorer les résultats des algorithmes en amélioration le traitement des données avec des méthodes de comme la suppression des données aberrante ou meilleure gestion des données manquent. Une autre manière serait de quantités des données plus grandes pour la partie apprentissage.

Pour améliorer l'étude, on pourrait effectuer plus d itérations afin d avec des moyens plus représentatifs du taux maximum.

2.3. Conclusion

En conclusion, même avec les biais exprimés plus haut et les améliorations proposées, on peut conclure que le réseau de neurones est la meilleure méthode pour résoudre le problème de désambiguïsation des sens des mots.

3.Notice du code

Le code qui a servi pour réaliser ce travail est contenus dans le fichier « desambiguïsation.py » en python, il utilise les librairie et importations suivantes :

```
import re
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

from nltk.stem import LancasterStemmer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.utils import column_or_1d
```

On passe à l au code dans l entêté de la formation « mains » un corpus et une liste de stop Word en format txt.

```
##############charge les donnes ##########################
fichier = open("interest.acl94.txt")
line = fichier.readlines()
fichier.close()

fichier = open("stoplist-english.txt")
tabStop = fichier.readlines()
fichier.close()

phraseV = phraseTab(line)
```

L’exécution est ensuite automatique une fois le code Runner et print les résultats sur console sous format : nombre de teste et taux de réussite.

```
"C:\Users\mike useni\Anaconda3\python.exe" "C:/Users/mike useni/Desktop/IA/TP2 IA/desambiguisation.py"
-------naive bayes-----------
nb tests:1070
resuletat:0.7572540693559802
------- tree -----------
nb tests:1058
resuletat:0.748761500353857
-------neural_network (100)-----------
nb tests:1189
resuletat:0.8414720452937013
-------neural_network (50)-----------
nb tests:1196
resuletat:0.8464260438782731
-------neural_network (200)-----------
nb tests:1196
resuletat:0.8464260438782731
```