

IFT-3295 - TP2

Rapport

Charlotte de Lanauze (20078840)

Maya Moussaoui (20157653)

Jian-Yun Nie

Université de Montréal

23 Décembre 2022

Fonctionnement du programme

Tout le code de notre travail est contenu dans le fichier `main.py`. Pour faire fonctionner le programme correctement, il suffit de décommenter les sections de code appropriées :

- Pour Naive Bayes, décommenter la section pour la caractéristique(sac de mot ou catégorie) et son appel(plus de détail dans les commentaires dans le code).
- Pour l'arbre de décision, décommenter la caractéristique voulus et son appel.
- Pour la forêt aléatoire, décommenter la caractéristique voulus et son appel.
- Pour SVM, décommenter la caractéristique voulus et son appel.
- Pour MultiLayerPerceptron, la caractéristique voulus et son appel.

Chacune de ces fonctions affiche les mesures de performance de l'algorithme en question, mais il faut s'assurer que les autres fonctions soient commentées pour que cela fonctionne.

Prétraitement

Pour le prétraitement, nous avons tout d'abord séparé le corpus en phrases (lignes 29-32). Ensuite, pour chaque phrase du corpus, nous avons séparé les mots et les avons stocké dans `corpus_sliced` (lignes 45-51) pour ensuite se débarrasser des "mots" ou plutôt des caractères inutiles tels que '[', ']', etc. Après avoir effectué les tâches mentionnées en haut, on fait appel à la fonction "`preparationData(nbElement)`", où l'argument `nbElement` représente le nombre de mots à chercher avant et après le mot d'intérêt, soit 2 fois `nbElement` mots au totale (le nombre en défaut est de 2). Cette fonction consiste à parcourir le `corpus_sliced` pour trouver les `nbElement` avant et les `nbElement` après le mot et sa catégorie d'intérêt (nous avons utilisé les expressions régulières pour les trouver car certains sont au pluriel "`interests_`" et d'autres non "`ineterst_`"). Tous les détails de cette fonction se trouvent dans le code lui-même sous forme de commentaire. Finalement, il ne restait qu'à mettre dans le bon ordre les catégories donc : de `C-nbElement` à `C+nbElement`.

Résultats de classification

Ci-dessous sont présentés les résultats des différents algorithmes implémentés avec une fenêtre de contexte de 2 mots avant et 2 mots après.

Naive Bayes

Pour la caractéristique sac de mot

```
Accuracy: 0.869198312236287
Confusion matrix: [[ 49  0  0  2  9  9]
 [ 1  0  0  0  0  2]
 [ 4  0  7  1  1  1]
 [ 0  0  0 16  2  7]
 [ 3  0  0  5 86  7]
 [ 3  0  0  2  3 254]]
```

Classification report:		precision		recall	f1-score
support					
1	0.82	0.71	0.76	69	
2	0.00	0.00	0.00	3	
3	1.00	0.50	0.67	14	
4	0.62	0.64	0.63	25	
5	0.85	0.85	0.85	101	
6	0.91	0.97	0.94	262	
accuracy			0.87	474	
macro avg		0.70	0.61	0.64	474
weighted avg		0.86	0.87	0.86	474

Pour la caractéristique categorie

```
Accuracy: 0.7616033755274262
Confusion matrix: [[ 37  0  6  4 11 11]
 [ 1  0  0  0  1  1]
 [ 0  0  6  2  4  2]
 [ 4  0  0 12  4  5]
 [ 5  0  4  7 73 12]
 [14  0  4  2  9 233]]
```

Classification report:		precision		recall	f1-score
support					
1	0.61	0.54	0.57	69	
2	0.00	0.00	0.00	3	
3	0.30	0.43	0.35	14	
4	0.44	0.48	0.46	25	
5	0.72	0.72	0.72	101	
6	0.88	0.89	0.89	262	
accuracy			0.76	474	
macro avg		0.49	0.51	0.50	474
weighted avg		0.76	0.76	0.76	474

Arbre de décision

Pour la caractéristique sac de mot

```
Accuracy: 0.8016877637130801
Confusion matrix: [[ 42  0  0  5  4 18]
 [ 0  0  0  0  0  3]
 [ 1  0  9  1  1  2]
 [ 1  0  0 17  3  4]
 [10  0  2  6 65 18]
 [ 8  0  1  3  3 247]]
```

Classification report:		precision		recall	f1-score
support					
1	0.68	0.61	0.64	69	
2	0.00	0.00	0.00	3	
3	0.75	0.64	0.69	14	
4	0.53	0.68	0.60	25	
5	0.86	0.64	0.73	101	
6	0.85	0.94	0.89	262	
accuracy			0.80	474	
macro avg		0.61	0.59	0.59	474
weighted avg		0.80	0.80	0.79	474

Pour la caractéristique categorie

```

Accuracy: 0.7784810126582279
Confusion matrix: [[ 37  0  1  5 12 14]
 [ 1  0  0  0  1  1]
 [ 0  0  8  1  3  2]
 [ 3  0  1 14  2  5]
 [10  0  0  9 70 12]
 [ 5  2  3  2 10 240]]
Classification report:
support
      1      0.66      0.54      0.59      69
      2      0.00      0.00      0.00       3
      3      0.62      0.57      0.59      14
      4      0.45      0.56      0.50      25
      5      0.71      0.69      0.70     101
      6      0.88      0.92      0.90     262

accuracy
macro avg      0.55      0.55      0.55     474
weighted avg    0.77      0.78      0.77     474

```

Forêt aléatoire

Pour la caractéristique sac de mot

```

Accuracy: 0.8248945147679325
Confusion matrix: [[ 35  0  0  2  7 25]
 [ 0  0  0  0  0  3]
 [ 1  0  8  1  0  4]
 [ 1  0  0 15  1  8]
 [ 3  0  1  3 75 19]
 [ 2  0  0  1  1 258]]

```

```

Classification report:
support
      1      0.83      0.51      0.63      69
      2      0.00      0.00      0.00       3
      3      0.89      0.57      0.70      14
      4      0.68      0.60      0.64      25
      5      0.89      0.74      0.81     101
      6      0.81      0.98      0.89     262

accuracy
macro avg      0.69      0.57      0.61     474
weighted avg    0.82      0.82      0.81     474

```

Pour la caractéristique categorie

```

Accuracy: 0.8164556962025317
Confusion matrix: [[ 41  0  0  4 13 11]
 [ 0  0  0  0  1  2]
 [ 0  0  6  3  3  2]
 [ 5  0  0 11  2  7]
 [ 6  0  0  3 79 13]
 [ 8  0  0  2  2 250]]

```

```

Classification report:
support
      1      0.68      0.59      0.64      69
      2      0.00      0.00      0.00       3
      3      1.00      0.43      0.60      14
      4      0.48      0.44      0.46      25
      5      0.79      0.78      0.79     101
      6      0.88      0.95      0.91     262

accuracy
macro avg      0.64      0.53      0.57     474
weighted avg    0.81      0.82      0.81     474

```

SVM

Pour la caractéristique sac de mot

```
Accuracy: 0.820675105485232
Confusion matrix: [[ 34  0  0  2  4 29]
 [ 0  0  0  0  0  3]
 [ 1  0  8  1  0  4]
 [ 0  0  0 12  0 13]
 [ 3  0  0  2 76 20]
 [ 2  0  0  1  0 259]]
```

Classification report:		precision		recall	f1-score
support					
1	0.85	0.49	0.62	69	
2	0.00	0.00	0.00	3	
3	1.00	0.57	0.73	14	
4	0.67	0.48	0.56	25	
5	0.95	0.75	0.84	101	
6	0.79	0.99	0.88	262	
accuracy			0.82	474	
macro avg		0.71	0.55	0.60	474
weighted avg		0.83	0.82	0.81	474

Pour la caractéristique categorie

```
Accuracy: 0.8164556962025317
Confusion matrix: [[ 40  0  0  2 13 14]
 [ 0  0  0  0  1  2]
 [ 0  0  6  2  4  2]
 [ 6  0  0 10  2  7]
 [ 2  0  0  3 86 10]
 [14  0  0  1  2 245]]
```

Classification report:		precision		recall	f1-score
support					
1	0.65	0.58	0.61	69	
2	0.00	0.00	0.00	3	
3	1.00	0.43	0.60	14	
4	0.56	0.40	0.47	25	
5	0.80	0.85	0.82	101	
6	0.88	0.94	0.90	262	
accuracy			0.82	474	
macro avg		0.65	0.53	0.57	474
weighted avg		0.81	0.82	0.81	474

MultiLayerPerceptron

Pour la caractéristique sac de mot

```
Accuracy: 0.7130801687763713
Confusion matrix: [[ 0  0  0  4 47 18]
 [ 0  0  0  1  1  1]
 [ 0  0  0  2 10  2]
 [ 0  0  0  2 18  5]
 [ 0  0  0  7 82 12]
 [ 0  0  0  2  6 254]]
```

Classification report:		precision		recall	f1-score
support					
1	0.00	0.00	0.00	69	
2	0.00	0.00	0.00	3	
3	0.00	0.00	0.00	14	
4	0.11	0.08	0.09	25	
5	0.50	0.81	0.62	101	
6	0.87	0.97	0.92	262	
accuracy			0.71	474	
macro avg		0.25	0.31	0.27	474
weighted avg		0.59	0.71	0.64	474

Pour la caractéristique categorie

```
Accuracy: 0.6940928270042194
Confusion matrix: [[ 29  0  0  8 19 13]
 [ 1  0  0  0  1  1]
 [ 1  0  7  1  3  2]
 [ 9  0  2  7  3  4]
 [12  0  1  9 58 21]
 [10  0  2  4 18 228]]
```

Classification report:		precision		recall	f1-score
support					
1	0.47	0.42	0.44	69	
2	0.00	0.00	0.00	3	
3	0.58	0.50	0.54	14	
4	0.24	0.28	0.26	25	
5	0.57	0.57	0.57	101	
6	0.85	0.87	0.86	262	
accuracy			0.69	474	
macro avg		0.45	0.44	0.45	474
weighted avg		0.69	0.69	0.69	474

Comparaison

On peut utiliser n'importe laquelle des mesures présentées ci-haut pour comparer nos résultats.

Prenons par exemple la mesure d'Accuracy, qui correspond à la précision en français. C'est une mesure pertinente dans notre cas, car on s'intéresse au taux de réussite de notre algorithme, c'est-à-dire le taux de classements qui ont été bien classés. Comme l'utilisation de la caractéristique de sacs de mots performe mieux que celle des catégories et ce pour tous les algorithmes, nous allons concentrer notre analyse sur les algorithmes qui utilisent la caractéristique de sacs de mots. On se retrouve avec les précisions suivantes arrondies à 5 décimales :

Algorithme	Naive Bayes	Arbre de décision	Forêt aléatoire	SVM	MultiLayerPerceptron
Précision	0.8692	0.80169	0.82489	0.82068	0.71308

L'algorithme ayant la meilleure précision est Naive Bayes, avec un score de précision de 86.920%. Les algorithmes Forêt aléatoire et SVM ne sont pas bien loin de ce score et valent certainement la peine d'être considérés.

Analyse de la performance des algorithmes

Naive Bayes

On observe que l'algorithme Naive Bayes est notre algorithme le plus performant. On s'attendait plutôt à ce que sa performance soit en dessous de celle d'autres algorithmes, puisqu'il fait l'hypothèse que toutes les caractéristiques en entrée sont indépendantes les unes des autres. Or, on sait que les différents mots et leurs catégories entourant le mot qui nous intéresse ne sont pas indépendants les uns des autres. En effet, certains mots ont plus de chance de se retrouver ensemble et certaines catégories ont également tendance à être regroupées ensemble. Un autre problème de Naive Bayes est qu'il considère que tous les mots du contexte ont la même importance pour prédire le résultat. Cependant, on sait que certains mots comme des déterminants ou la ponctuation n'ont pas beaucoup d'influence sur le sens du mot interest (ou tout autre mot). Malgré tout, il performe très bien pour notre tâche. L'avantage avec Naive Bayes est qu'il est très facile à implémenter et ne requière pas beaucoup de données pour bien performer. C'est peut-être ce qui explique sa bonne performance dans notre cas.

Arbre de décision et forêt aléatoire

L'arbre de décision est un algorithme qui est facile à interpréter, mais qui a une grande variance, c'est-à-dire que des changements minimes dans les données d'apprentissage vont entraîner des changements importants dans le résultat final. Comme pour la forêt aléatoire, l'arbre de décision est un algorithme non déterministe. Il performe en général moins bien que la forêt aléatoire, ce qui est ce à quoi on s'attendait puisque la forêt aléatoire a une plus faible variance que l'arbre de décision. Cette variance plus faible, qu'on pourrait qualifier de robustesse, aide à obtenir des résultats plus exacts dans la tâche de classification. La forêt aléatoire est cependant plus difficile à interpréter que l'arbre de décision et le processus est plus lent, mais cela reste l'algorithme le plus approprié pour notre tâche parmi les deux.

SVM

Un autre algorithme qui performe plutôt bien est SVM. C'est un algorithme pertinent, puisqu'il n'est pas biaisé par les valeurs aberrantes et il n'est pas sensible au surentraînement. Son seul désavantage est qu'il ne s'applique qu'aux problèmes qui sont linéaires, mais il fonctionne bien dans ce cas-ci.

MultiLayer Perceptron

L'algorithme le moins précis dans notre cas est celui de MultiLayerPerceptron, qui avec un score de 71.308%, performe significativement moins bien que les autres algorithmes. Son principal avantage est qu'il peut être appliqué à des problèmes non linéaires et est efficace avec de très grandes quantités de données. Une fois que les données sont entraînées, il est très rapide à donner des prédictions. Cependant, ces avantages sont peu significatifs dans notre cas, puisque tous les algorithmes sont plutôt rapides. Il se pourrait qu'il n'y ait pas assez de données pour que le MultiLayer Perceptron ait un réel avantage, ce qui expliquerait sa performance moindre.

Effet des caractéristiques sur la performance

Les résultats des algorithmes ci-hauts ont été obtenus avec une fenêtre de contexte de quatre mots avant et quatre mots après. Pour tester l'effet que cet hyperparamètre a sur la performance, nous allons présenter les résultats de différentes tailles de fenêtre de contexte sur l'algorithme Naive Bayes et assumer que cela produirait le même effet sur les autres algorithmes.

Taille de la fenêtre de contexte	2	3	4	5	6	7
Précision	0.87764	0.85865	0.8692	0.85021	0.84388	0.83544

On observe que la précision de l'algorithme est optimal avec une fenêtre de contexte de 2 (soit 2 mots avant et 2 mots après) et tend à plutôt diminuer quand on augmente sa taille. Cela est dû au surentraînement, c'est à dire qu'avec plus de mots dans la fenêtre de contexte, l'algorithme devient trop spécifique lors de l'entraînement et son apprentissage se généralise moins bien à l'ensemble des données.