LETTRES
SORBONNE
UNIVERSITÉ

28/12/2018

Projet Méthodologie

Méthodologie :

Modélisation informatique et interaction sociale

GAO SHUAI
28505363

Sommaire

Table des matières

1. Introduction 2
2. Jeu de données 2
   2.1. Données, outils et objectif 2
   2.2. Hypothèse 3
3. Analyse statistique du corpus d'apprentissage 3
   3.1. Créations des features 3
   3.2. Exécution du code Python et création des fichiers .arff 4
4. Tests classification Weka 6
   4.1. J48 8
   4.2. NaiveBayes 9
   4.3. RandomForest 10
5. Récapitulatif 11
   5.1. all_features 11
   5.2. verbs_past_tense 11
   5.3. verbs_present_tense 12
6. Conclusion et perspectives 13

1. Introduction

De nos jours, le développement de la technologie et le foisonnement des outils informatiques nous permettent grandement une vie « artificielle ». Entourés d'ordinateurs, de tablettes et surtout de nos smartphones, partenaires omniprésents semble-t-il, nous sommes en permanence connectés à un cyberespace, lequel, d'une façon ou une autre, interagit avec nous.

Il est à noter que cette interaction tend à se caractériser par l'apprentissage automatique, lequel rend l'intelligence artificielle de plus en plus intelligente. Ce processus consiste à utiliser un simple algorithme dit apprentissage pour « enseigner » à la machine comment traiter les données, sans avoir besoin de tout programmer. Cependant, l'efficacité de cet algorithme est toujours à évaluer.

Dans ce rapport du projet, nous mettrons tout d'abord en exergue le jeu de données. Nous nous pencherons ensuite sur l'analyse statistique du corpus d'apprentissage ainsi que sur les tests de classifications à l'aide de Weka. Nous arriverons finalement à une conclusion, tout en élargissant nos pistes de réflexion.

2. Jeu de données

2.1. Données, outils et objectif

Dans ce projet, nous travaillons sur le corpus Brown du NLTK (Natural Language Toolkit). Ce corpus est en anglais et il possède trois genres de textes :

- Journalistiques (news) ;
- Scientifiques (sciences) ;
- Littéraires (literature).

Notre projet consiste à définir d'abord nos propres features dans un fichier .json et à les exécuter en Python (les codes à nous ont été fournis) afin de créer des fichiers .arff. Nous chargeons finalement ces fichiers .arff dans Weka (Waikato Environment for Knowledge Analysis), logiciel d'apprentissage automatique de la

machine, en vue d'évaluer l'efficacité de nos features pour différencier les trois types de textes corpus Brown.

2.2. Hypothèse

Étant donné que le corpus Brown a trois genres de textes, nous nous demandons s'il y a un phénomène morphologique qui pourrait les distinguer. Nous supposons donc que les verbes les plus utilisés en anglais, tels que « be », « have » et « do » etc. soient très présents au passé dans les textes journalistiques, moyennement présents dans les textes littéraires et peu présents dans les textes scientifiques, et que ces verbes précités soient très présents au présent dans les textes scientifiques, moyennement présents dans les textes littéraires et peu présents dans les textes journalistiques.

3. Analyse statistique du corpus d'apprentissage

Comme mentionnés ci-dessus, les codes en Python nous ont été fournis pour procéder à ce projet. Nous avons quatre fichiers .py nommés respectivement « my_toolsv2 », « phase1 », « phase2 » et « phase3 ». Nous faisons quelques modifications des fonctions selon nos besoins dans le premier, et nous combinons les trois derniers dans une seule page de Jupyter Notebook.

3.1. Créations des features

On établit une liste de 25 verbes anglais les plus utilisés et la transforme en un fichier .json. Puisque notre hypothèse consiste à distinguer des textes de types différents, nous gardons le corpus comme tel qu'il est, ayant « news », « literature » et « sciences ».

```json
{
    "verbs_present_tense":
        [
            "be", "am", "are", "is", "have", "has",
            "do", "does", "say", "says", "go", "goes",
            "get", "gets", "make", "makes", "know", "knows",
            "think", "thinks", "take", "takes",
            "see", "sees", "come", "comes", "want", "wants",
            "look", "looks", "use", "uses", "find", "finds",
            "give", "gives", "tell", "tells", "work", "works",
            "call", "calls", "try", "tries", "ask", "asks",
            "need", "needs", "feel", "feels", "become", "becomes"
        ],

    "verbs_past_tense":
        [
            "was", "were", "had",
            "did", "said", "went",
            "got", "made", "knew",
            "thought", "took",
            "saw", "came", "wanted",
            "looked", "used", "found",
            "gave", "told", "worked",
            "called", "tried", "asked",
            "needed", "felt", "became"
        ]
}
```

3.2. Exécution du code Python et création des fichiers .arff

En exécutant la première phase, nous obtenons un fichier « trains_test.json » avec une description en sortie comme suit :

```
news :
   88 documents
literature :
   88 documents
sciences :
   80 documents
NB instances : 256
On prend 17 éléments sur 88 pour le test
On prend 17 éléments sur 88 pour le test
On prend 16 éléments sur 80 pour le test
Dataset stocké dans train_test.json
```

Pour entrainer les algorithmes, il faut leur donner des textes sur lesquels ils peuvent apprendre. Pour cela, nous on utilise 80% de notre corpus entier. Ensuite pour tester les algorithmes, il faut les tester sur des textes qu'il ne connaît pas, d'où les 20% restants.

Avant d'adapter le code dans la deuxième phase, on est obligé de modifier le code dans « my_toolsv2.py » pour qu'il puisse compter les occurrences des verbes à temps différents dans notre fichier de features.

```
81  def get_stats_occurrence(liste, json):
82      occu_present_tense = 0
83      occu_past_tense = 0
84      usual_verbs_set = lire_json(json)
85      for verb in liste:
86          if verb in usual_verbs_set["verbs_present_tense"]:
87              occu_present_tense += 1
88          if verb in usual_verbs_set["verbs_past_tense"]:
89              occu_past_tense += 1
90      return {"verbs_present_tense": occu_present_tense, "verbs_past_tense": occu_past_tense}
```

On adapte ensuite le code dans la deuxième phase comme suit :

```
def get_features_light(liste_fichier):
  features_file = {}
  for fileid in liste_fichier:
    features_file[fileid] = {}
    stats_mots = mt.get_stats_occurrence(brown.words(fileid), "usual_verbs.json") # my_toolsv2 ; liste des mots dans chaq
    for feature, valeur in stats_mots.items():
      features_file[fileid][feature] = valeur # couple clé-valeur; utiliser "1/0" ou "input" pour arrêter le programme
  print("->Features extraites:", list(features_file[fileid].keys())[:20],"...")
  return features_file
```

Ainsi, nous l'exécutons et nous obtenons un fichier .json qui stocke les features dans tous les textes du corpus.

```
Récupération de la liste des fichiers
-> 256 fichiers

Extraction des features
->Features extraites: ['verbs_present_tense', 'verbs_past_tense'] ...

Ecriture de la sortie JSON
-> features_by_file.json
```

On adapte également le code de la troisième phase comme suit :

```
###Définition des combinaisons de features
liste_verbes = mt.lire_json("usual_verbs.json") # jeu de features
feature_sets = {"all_features" : feature_names,
                "verbs_present_tense" : ["verbs_present_tense"],
                 "verbs_past_tense": ["verbs_past_tense"]}
```

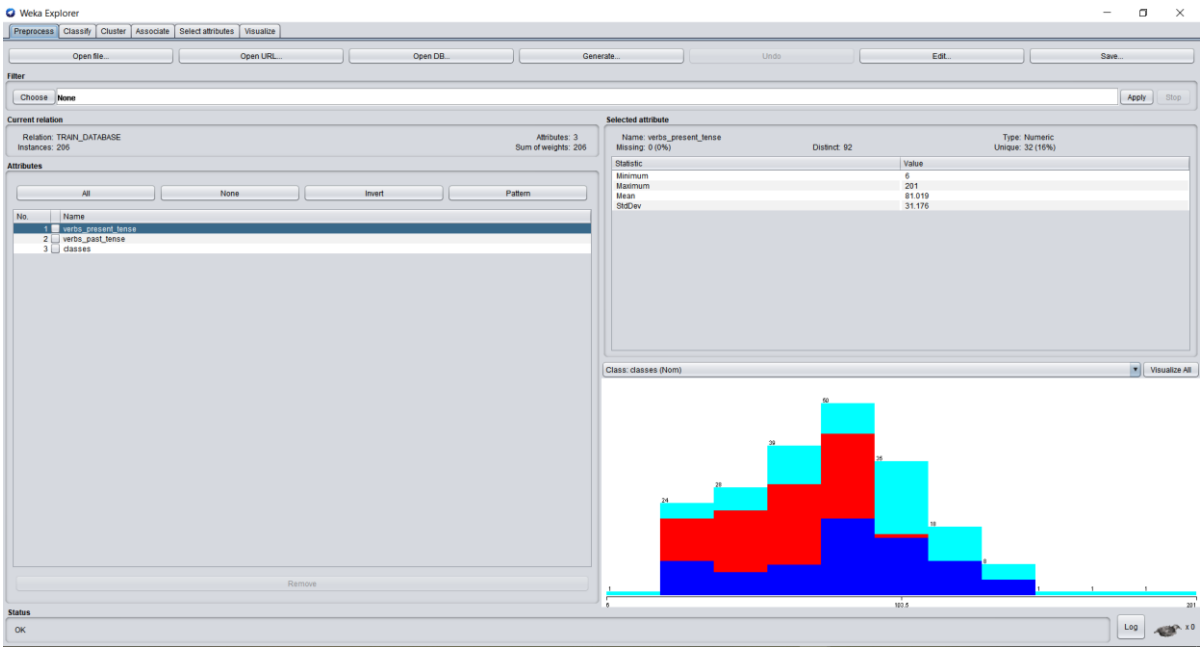
On l'exécute et les fichiers .arff sont générés.

```
Dossier 'arff_files' créé

Feature set : all_features
  Processing train set
    ->sortie = arff_files/all_features__train.arff
  Processing test set
    ->sortie = arff_files/all_features__test.arff

Feature set : verbs_present_tense
  Processing train set
    ->sortie = arff_files/verbs_present_tense__train.arff
  Processing test set
    ->sortie = arff_files/verbs_present_tense__test.arff

Feature set : verbs_past_tense
  Processing train set
    ->sortie = arff_files/verbs_past_tense__train.arff
  Processing test set
    ->sortie = arff_files/verbs_past_tense__test.arff
```

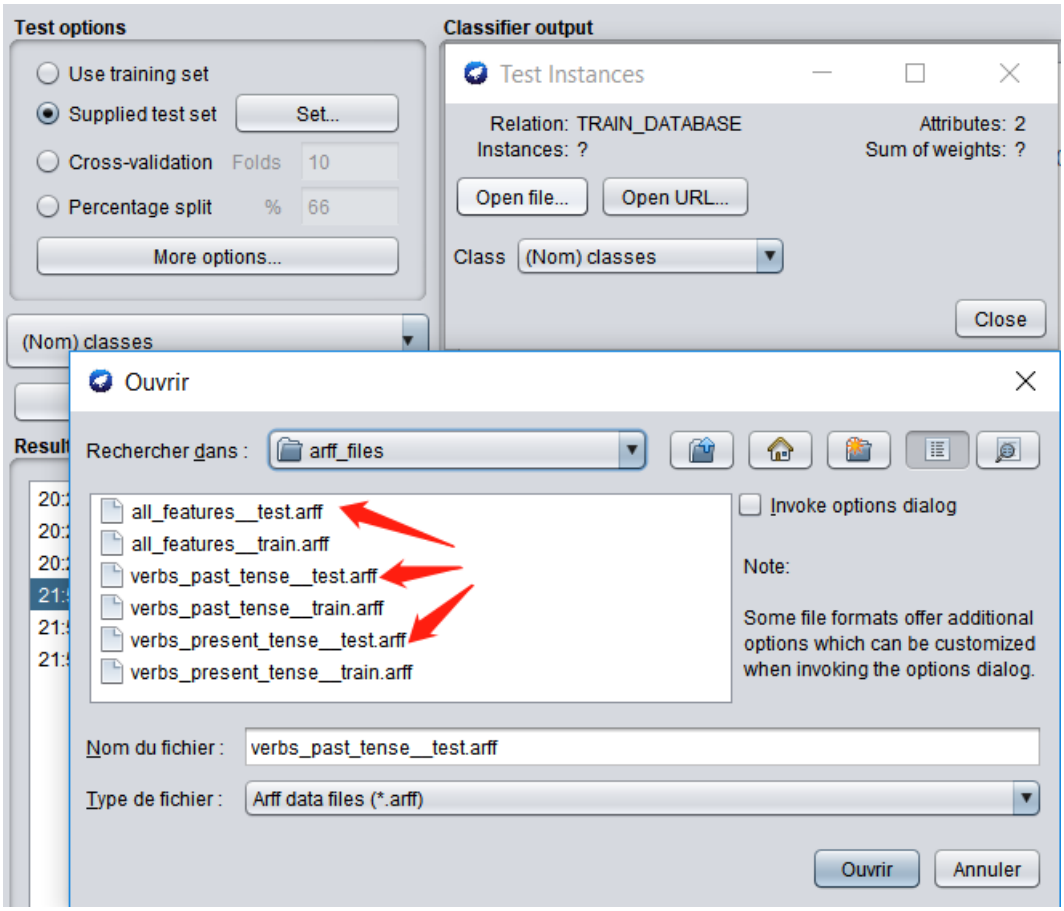
4. Tests classification Weka

Après avoir créé les fichiers .arff, nous faisons des tests de classification à l'aide de Weka.

En entrant dans Weka, on charge d'abord le fichier « all_features_train.arff ».



Nous allons ensuite dans l'onglet « Classify ». Pour configurer « Test options », on croche « Supplied test set », clique sur « set… » et charge le fichier « all_features_test.arff ».



Nous procédons ensuite des tests avec différents classifieurs. Une fois trois tests à J48, à NaiveBayes et à RandomForest faits, nous changeons dès l'entrée le fichier .arff successivement en « verbs_past_tense_train.arff » et en « verbs_present_tense_train.arff », et nous refaisons deux fois les trois tests.

4.1. J48

```
=== Summary ===

Correctly Classified Instances          36               72      %
Incorrectly Classified Instances        14               28      %
Kappa statistic                          0.5775
Mean absolute error                      0.2566
Root mean squared error                  0.3897
Relative absolute error                 57.7774 %
Root relative squared error             82.6692 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,824    0,152    0,737      0,824    0,778      0,656    0,873     0,699     news
                 1,000    0,212    0,708      1,000    0,829      0,747    0,894     0,708     literature
                 0,313    0,059    0,714      0,313    0,435      0,341    0,632     0,463     sciences
Weighted Avg.    0,720    0,142    0,720      0,720    0,686      0,586    0,803     0,627

=== Confusion Matrix ===

  a  b  c   <-- classified as
 14  1  2 |  a = news
  0 17  0 |  b = literature
  5  6  5 |  c = sciences
```

```
=== Summary ===

Correctly Classified Instances          36               72      %
Incorrectly Classified Instances        14               28      %
Kappa statistic                          0.5775
Mean absolute error                      0.2566
Root mean squared error                  0.3897
Relative absolute error                 57.7774 %
Root relative squared error             82.6692 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,824    0,152    0,737      0,824    0,778      0,656    0,873     0,699     news
                 1,000    0,212    0,708      1,000    0,829      0,747    0,894     0,708     literature
                 0,313    0,059    0,714      0,313    0,435      0,341    0,632     0,463     sciences
Weighted Avg.    0,720    0,142    0,720      0,720    0,686      0,586    0,803     0,627
```

```
=== Summary ===

Correctly Classified Instances          23               46      %
Incorrectly Classified Instances        27               54      %
Kappa statistic                          0.1892
Mean absolute error                      0.4045
Root mean squared error                  0.4562
Relative absolute error                 91.0817 %
Root relative squared error             96.7821 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,000    0,000    ?          0,000    ?          ?        0,585     0,392     news
                 0,941    0,576    0,457      0,941    0,615      0,378    0,683     0,450     literature
                 0,438    0,235    0,467      0,438    0,452      0,206    0,601     0,384     sciences
Weighted Avg.    0,460    0,271    ?          0,460    ?          ?        0,623     0,409
```

4.2. NaiveBayes

```
=== Summary ===

Correctly Classified Instances          31               62      %
Incorrectly Classified Instances        19               38      %
Kappa statistic                          0.4284
Mean absolute error                      0.2835
Root mean squared error                  0.4144
Relative absolute error                 63.8352 %
Root relative squared error             87.9109 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,529    0,121    0,692      0,529    0,600      0,441    0,856     0,712     news
                 0,941    0,273    0,640      0,941    0,762      0,633    0,918     0,831     literature
                 0,375    0,176    0,500      0,375    0,429      0,217    0,642     0,527     sciences
Weighted Avg.    0,620    0,190    0,613      0,620    0,600      0,435    0,808     0,693

=== Confusion Matrix ===

  a  b  c   <-- classified as
  9  3  5 |  a = news
  0 16  1 |  b = literature
  4  6  6 |  c = sciences
```

```
=== Summary ===

Correctly Classified Instances          32               64      %
Incorrectly Classified Instances        18               36      %
Kappa statistic                          0.4578
Mean absolute error                      0.2835
Root mean squared error                  0.4031
Relative absolute error                 63.8392 %
Root relative squared error             85.5193 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,588    0,152    0,667      0,588    0,625      0,451    0,854     0,679     news
                 1,000    0,242    0,680      1,000    0,810      0,718    0,931     0,834     literature
                 0,313    0,147    0,500      0,313    0,385      0,193    0,630     0,475     sciences
Weighted Avg.    0,640    0,181    0,618      0,640    0,611      0,459    0,808     0,666
```

```
=== Summary ===

Correctly Classified Instances          24               48      %
Incorrectly Classified Instances        26               52      %
Kappa statistic                          0.2159
Mean absolute error                      0.4043
Root mean squared error                  0.4501
Relative absolute error                 91.038  %
Root relative squared error             95.4966 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,353    0,212    0,462      0,353    0,400      0,152    0,639     0,488     news
                 0,824    0,455    0,483      0,824    0,609      0,354    0,702     0,513     literature
                 0,250    0,118    0,500      0,250    0,333      0,168    0,668     0,486     sciences
Weighted Avg.    0,480    0,264    0,481      0,480    0,450      0,226    0,670     0,496
```

4.3. RandomForest

```
=== Summary ===

Correctly Classified Instances          28               56      %
Incorrectly Classified Instances        22               44      %
Kappa statistic                          0.3389
Mean absolute error                      0.274
Root mean squared error                  0.4411
Relative absolute error                 61.6933 %
Root relative squared error             93.5758 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,412    0,182    0,538      0,412    0,467      0,248    0,766     0,656     news
               0,941    0,212    0,696      0,941    0,800      0,693    0,943     0,836     literature
               0,313    0,265    0,357      0,313    0,333      0,050    0,649     0,433     sciences
Weighted Avg.  0,560    0,219    0,534      0,560    0,537      0,336    0,789     0,646

=== Confusion Matrix ===

  a  b  c   <-- classified as
  7  2  8 |  a = news
  0 16  1 |  b = literature
  6  5  5 |  c = sciences


=== Summary ===

Correctly Classified Instances          28               56      %
Incorrectly Classified Instances        22               44      %
Kappa statistic                          0.3389
Mean absolute error                      0.2754
Root mean squared error                  0.4566
Relative absolute error                 62.0165 %
Root relative squared error             96.8608 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,529    0,182    0,600      0,529    0,563      0,359    0,780     0,604     news
               0,824    0,212    0,667      0,824    0,737      0,587    0,898     0,728     literature
               0,313    0,265    0,357      0,313    0,333      0,050    0,568     0,414     sciences
Weighted Avg.  0,560    0,219    0,545      0,560    0,548      0,338    0,752     0,585


=== Summary ===

Correctly Classified Instances          20               40      %
Incorrectly Classified Instances        30               60      %
Kappa statistic                          0.1039
Mean absolute error                      0.3707
Root mean squared error                  0.4856
Relative absolute error                 83.4622 %
Root relative squared error            103.0169 %
Total Number of Instances               50

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,294    0,152    0,500      0,294    0,370      0,169    0,647     0,516     news
               0,471    0,242    0,500      0,471    0,485      0,232    0,770     0,622     literature
               0,438    0,500    0,292      0,438    0,350      -0,058   0,589     0,506     sciences
Weighted Avg.  0,400    0,294    0,433      0,400    0,403      0,118    0,670     0,549
```

5. Récapitulatif

Après avoir fait ces trois tests (résultats capturés ci-dessus), nous récapitulons leurs résultats afin de mieux faire une analyse comparative.

5.1. all_features

	F-Mesure	Rappel	Précision	Instances correctement classifiés
J48	0,686	0,720	0,720	72%
NaiveBayes	0,600	0,620	0,613	62%
RandomForest	0,537	0,560	0,534	56%

	Matrice de confusion
J48	a b c <-- classified as 14 1 2 | a = news 0 17 0 | b = literature 5 6 5 | c = sciences
NaiveBayes	a b c <-- classified as 9 3 5 | a = news 0 16 1 | b = literature 4 6 6 | c = sciences
RandomForest	a b c <-- classified as 7 2 8 | a = news 0 16 1 | b = literature 6 5 5 | c = sciences

5.2. verbs_past_tense

	F-Mesure	Rappel	Précision	Instances correctement classifiés
J48	0,686	0,720	0,720	72%
NaiveBayes	0,611	0,640	0,618	64%
RandomForest	0,548	0,560	0,545	56%

	Matrice de confusion
J48	a b c <-- classified as 14 1 2 | a = news 0 17 0 | b = literature 5 6 5 | c = sciences
NaiveBayes	a b c <-- classified as 10 2 5 | a = news 0 17 0 | b = literature 5 6 5 | c = sciences
RandomForest	a b c <-- classified as 9 1 7 | a = news 1 14 2 | b = literature 5 6 5 | c = sciences

5.3. verbs_present_tense

	F-Mesure	Rappel	Précision	Instances correctement classifiés
J48	?	0,460	?	46%
NaiveBayes	0,450	0,480	0,481	48%
RandomForest	0,403	0,400	0,433	40%

	Matrice de confusion
J48	a b c <-- classified as 0 10 7 | a = news 0 16 1 | b = literature 0 9 7 | c = sciences
NaiveBayes	a b c <-- classified as 6 9 2 | a = news 1 14 2 | b = literature 6 6 4 | c = sciences
RandomForest	a b c <-- classified as 5 3 9 | a = news 1 8 8 | b = literature 4 5 7 | c = sciences

Somme toute, pour les tests d' « all_features » et de « verbs_past_tense », J48 est le meilleur classifieur ayant 72% des résultats correctement classifiés. Pour ce qui est de « verbs_present_tense », NaiveBayes est le meilleur. Cependant, aucun de ces trois classifieurs ne sont pas performants ici.

6. Conclusion et perspectives

Les résultats que nous avons obtenus sont en général peu satisfaisants. Seulement avec le meilleur classifieur J48, le taux de classification correcte atteint 72%, ce qui est, en ce qui me concerne, passable et un peu loin d'être performant. Nos features s'avèrent par conséquent être peu pertinents dans la différenciation des trois types, surtout les verbes au présent.

On s'interroge tout de suite sur la raison de cette inefficacité. On se rend compte donc que pour les verbes en anglais, les participes passés se manifestent très souvent sous la même forme que ses passés, et que la présence des verbes au présent est beaucoup plus générale que l'on imaginait, ce qui ne peut donc pas parler grand-chose sur la distinction des trois types. D'où la déficience de nos features.

Si l'on prenait en compte les participes passés ainsi que des auxiliaires, ce qui demande certainement un algorithme beaucoup plus complexe, et que l'on n'envisage pas le présent, la performance de ces features basés sur les formes des verbes usuels en anglais pourrait être améliorée dans une large mesure.