Introduction à la fouille de textes

Rapport : Une tâche de classification des textes de paroles par auteur dans Weka.

Luísa Nascimento Batista, Sandra Jagodzińska

M1 TAL

2022-2023

Introduction

Avec le développement de la technologie, nous sommes en mesure de livrer et de stocker de plus en plus de masses de données. Par conséquent, l'intérêt d'un domaine tel que l'exploration de données connu sous le nom anglais “data mining” a émergé. L’une des parties de *data mining* est le “text mining” qui en français est appelé “la fouille de texte”. Autrement appelé “l’extraction de connaissances” est une partie du domaine de l’intelligence artificielle.[1] La fouille de texte exploite des algorithmes afin d'extraire des connaissances selon des critères donnés du texte produit par un être humain.

Ce domaine qui trouve ces plusieurs amateurs en TAL s’occupe de quatre tâches principales : Recherche d'information, Extraction d'information, Annotation et Classification. C’est ce dernier qui est très crucial pour nous. Nous allons l’utiliser, tester et découvrir ses usages, aussi ses possibilités au sein de ce projet.

Notre approche est basée sur l’apprentissage automatique. Alors, nous allons fournir les données d’apprentissage à un programme, c'est-à-dire un échantillon déjà classé manuellement, sur lequel le programme va s’améliorer, afin d’apprendre à classifier les données de façon automatisée.

Le logiciel d’apprentissage automatique exploité dans ce projet s’appelle Weka. C’est un logiciel libre développé à l'université de Waikato en Nouvelle-Zélande. Il contient une collection d'algorithmes d’analyse de données et de classification. Notre but est de tester ces algorithmes proposés pour une tâche de classification des paroles de chansons par auteur. Nous avons recueilli des paroles de deux autrices : Taylor Swift et Adele, dont chacune constitue une classe. La quantité de paroles par chaque classe était au début un peu déséquilibrée, ce que nous allons aborder dans le chapitre “Constitution du Corpus”. Alors, une fois que nous avons établi notre corpus, nous avons lancé le script de vectorisation fourni par notre professeur afin de transformer nos données en fichiers exploitables par Weka. Ensuite, nous avons expérimenté différents algorithmes et paramètres afin de comparer les performances des algorithmes sur notre tâche et d'obtenir la meilleure classification de nos données.

[1] https://fr.wikipedia.org/wiki/Fouille_de_textes

Constitution du corpus

Comme déjà rapidement mentionné, notre corpus est constitué de paroles de chansons de deux chanteuses anglophones, l'anglaise Adele et l'américaine Taylor Swift. Notamment, chacune des chanteuses compose une classe différente pour le critère de classification de nos textes. Extraits dans des sites[2] de paroles, ces textes ont été enregistrés en format txt. Nous avons choisi de constituer un corpus en anglais car il est plus aisé de retrouver des contenus dans cette langue, particulièrement dans le sujet sur lequel nous souhaitions travailler, les paroles de chansons.

Nous avons fait ce choix afin de tester l'efficacité de l'outil pour classifier les chansons selon leurs chanteurs – une tâche que nous pensions être assez défiante. Mais, en même temps, il fallait que ces chanteurs soient aussi les auteurs de leurs chansons. Cela nous a donc mené à un défi lors de la constitution de notre corpus, puisque la plupart des chanteurs ne sont pas forcément auteurs, surtout de toutes leurs propres chansons. Ainsi, dans ce contexte, nous avons trouvé ces deux chanteuses, presque les seules qui écrivent leurs propres paroles dans l'actualité. Cette exigence s'est imposée parce que la plupart des chanteurs utilisent des paroles écrites par les mêmes groupes d'auteurs, donc il ne serait pas possible de classifier les chansons écrites par la même personne.

À propos des propriétés de notre corpus : tout d'abord, nous avions récupéré 105 paroles de Taylor Swift et 56 paroles d'Adele. Comme nous avions ajouté toutes les 105 chansons de la première chanteuse dans un seul fichier ("TaylorSwift.txt"), on a dû créer un script simple ("TS-separation.ipynb") pour les séparer en fichiers de la même quantité de titres différents. De cette manière, nous avons eu l'impression que notre corpus n'était pas équilibré du tout : il faisait presque le double de fichiers pour la première classe par rapport à la deuxième. Après avoir testé quelques outils sur Weka, nous étions sûres qu'il fallait équilibrer notre corpus : tout de suite, nous avons dû supprimer 30 chansons dans la première classe, en restant 85 par conséquent. Une autre modification effectuée aussi sur ce corpus : l'enlèvement du nom de la chanteuse qui apparaissait au-dessous des titres lors de la récupération des paroles. Finalement, nous avons constitué un corpus avec deux classes différentes contenant 141 textes / paroles au total, qui font la taille moyenne de 1823 caractères (calcul fait par notre script "longueurMoyenne.py").

[2] https://www.letras.com/ et https://www.lyricfind.com/

Taylor Swift	Adele	TOTAL
85	56	141

Méthodologie

Pour rendre ce projet le plus collaboratif possible, nous avons créé un répertoire sur GitHub[3] pour échanger des fichiers et des scripts entre nous, ainsi que des outils de collaboration Google pour rédiger le rapport. Chacune a récolté des paroles de chansons d'une chanteuse, et puis les a placées sur GitHub. Pour résoudre des problèmes rencontrés (séparation de paroles d'un fichier, calcul de longueur moyenne), chacune a écrit un script Python. Vu que notre corpus a évolué car nous l'avons un peu réduit et équilibré après les premières tentatives, chacune de nous a eu la chance d'utiliser le script de vectorisation. De plus, nous n'avons pas eu besoin de créer un fichier avec les mots vides parce qu'ils n'apparaissent pas dans les critères de classification.

Comme abordé précédemment, le corpus avec des chansons de Taylor Swift était deux fois plus grand que celui d'Adele. Nous nous en sommes rendu compte très vite en raison des facteurs suivants :

- Le corpus de Taylor constitue une classe majoritaire, à tel point que le corpus est vraiment mal réparti ;
- Les deux chanteurs créent des chansons d'amour, souvent sur des ruptures ou des pertes, des chagrins et des sentiments, de sorte que les champs lexicaux se chevauchent souvent.

que les textes d'Adèle étaient très mal classés. Les divers algorithmes exploités n'arrivaient pas à les classer correctement. Presque tous ses textes étaient classés comme ceux de Taylor. Alors, nous avons convenu qu'il fallait réduire le corpus de chansons de Taylor dans l'espoir que des mots distinctifs émergeraient, que la nouvelle répartition changerait les choix faits par les algorithmes et que la classification serait meilleure. De cette manière, nous avons supprimé les textes au hasard afin qu'il n'y ait pas d'interférence dans le texte ou de choix artificiel.

[3] https://github.com/SandraJagodzinska/projetWeka

Une fois obtenu un nouveau corpus, nous avons procédé avec des expériences. Après avoir testé différentes mesures de division du corpus en partie train et test, nous avons décidé qu'avec notre corpus relativement petit et pas également réparti, la validation croisée marche le mieux. C'est une méthode qui divise un corpus en *n* parties égales et prend comme *train* *n*-1 parties et le reste comme *test*. L'opération est répétée *n* fois pour avoir toutes les combinaisons de *train* et *test* possibles. Cela permet d'entraîner mieux sur les textes d'Adele dont il y a nettement moins, qu'ils ne le seraient si la division en *train* et *test* était constante. Enfin, notre partie test (en Weka "Folds") est égale à 10% dans chaque expérience que nous allons présenter ensuite.

Expérimentations

Naive Bayes

Cet algorithme utilise la méthode fondée sur les calculs probabilistes, surtout comme son nom indique sur le "Théorème de Bayes". Certaines variantes du modèle de ce type sont plus efficaces lorsque les données d'entrée sont au format booléen, et d'autres donnent les meilleurs résultats lorsqu'elles sont sous forme de calculs. Alors, nous avons décidé de charger les deux formats dans le Weka et d'explorer différentes variantes de cet algorithme, afin de comparer les sorties et les changements.

Nous avons utilisé le Naive Bayes Multinomial avec le fichier contenant des *counts* (donc non-boolean) et nous avons obtenu la classification et les mesures suivantes :

Nombre de textes bien classés	108 (76.5957%)
Nombre de textes mal classés	33 (23.4043%)
Précision	0.771 (77,1%)
Rappel	0.766 (76,6%)
F-mesure	0.767 (76,7%)

MATRICE DE CONFUSION		
Adele	**Taylor**	**← classified as**
42	14	**Adele**
19	66	**Taylor**

L'expérience avec un fichier booléen avait les meilleurs résultats de la classification avec un algorithme Naive Bayes. Nous avons obtenu une amélioration des résultats par rapport à l'expérience précédente, et les mesures ont augmenté d'environ 10 %. À partir de 33 textes mal classés, nous avons réduit les résultats à 19 textes, cela montre que près de la moitié des erreurs ont été éliminées. Certainement, dans cette expérience les textes de Taylor étaient mieux classés, et seulement 4 d'entre eux ont été mis dans la mauvaise classe. Malheureusement, quant aux paroles d'Adele, l'algorithme n'a pas fait beaucoup de progrès.

Nombre de textes bien classés	122 (86.5248%)
Nombre de textes mal classés	19 (13.4752%)
Précision	0.871 (87,1%)
Rappel	0.865 (86,5%)
F-mesure	0.862 (86,2%)

MATRICE DE CONFUSION		
Adele	**Taylor**	**← classified as**
41	15	**Adele**
4	81	**Taylor**

Remarques : La plupart des textes de paroles étaient bien classifiés. Seulement 14 textes d'Adele parmi 56 étaient mal classés pendant la première expérience, et 15 pendant la deuxième. Un peu plus car 19 paroles de Taylor parmi 85 étaient classifiées comme celles d'Adele pendant la première expérience, avec une grosse amélioration pendant la deuxième où seulement 4 paroles de Taylor étaient mal classées.

Les résultats sont satisfaisants vu que les deux chanteuses écrivent leurs paroles sur des sujets similaires, et des mots redondants sont très souvent les mêmes. Les résultats de ces deux expériences, selon nous, sont surprenants car nous ne nous attendions pas à ce que l'algorithme se révèle aussi efficace avec des textes aussi spécifiques dans lesquels il y a peu de différences notables.

Nous avons essayé d'analyser quelques mots qui ont plus d'importance pour une classe donnée, et quels mots prédominent dans chaque catégorie, pour voir comment l'algorithme fonctionne et comment il décide. En se basant sur des résultats affichés[4] dans l'expérience avec l'algorithme Naive Bayes qui utilise un fichier booléen, nous avons remarqué que :

- Il est plus probable que la chanson avec des mots "boys" ou "boyfriend" appartient à la classe Taylor qu'Adele, par contre le mot "boy" est aussi bien probable pour les deux chanteuses.

```
boy
  mean                                0.1071 0.0706
  std. dev.                           0.3093 0.2561
  weight sum                              56     85
  precision                                1      1

boyfriend
  mean                                       0 0.0235
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

boys
  mean                                       0 0.0588
  std. dev.                           0.1667 0.2353
  weight sum                              56     85
  precision                                1      1
```

[4] Dans tous les captures d'écran la première colonne appartient à la classe Adele et la deuxièeme à la classe Taylor

- Les deux chanteuses utilisent le vocabulaire liée à l'amour (“love”, “lover”, “lovely”...), comme nous avons mentionné avant, les champs lexicaux se chevauchent.

```
loved
  mean                                0.0893 0.1529
  std. dev.                           0.2852 0.3599
  weight sum                              56     85
  precision                                1      1

lovely
  mean                                     0 0.0118
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

lover
  mean                                0.0357 0.0353
  std. dev.                           0.1856 0.1845
  weight sum                              56     85
  precision                                1      1

lover'
  mean                                     0 0.0118
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

lovers
  mean                                0.0357 0.0118
  std. dev.                           0.1856 0.1667
  weight sum                              56     85
  precision                                1      1

loves
  mean                                0.0357 0.0588
  std. dev.                           0.1856 0.2353
  weight sum                              56     85
  precision                                1      1
```

- Adele utilise le mot “hope”, “espoir”, mais une fois le mot “hopeless”, “désespéré”, apparaît il est complètement distinctif pour Taylor.

```
hope
  mean                                 0.125 0.0471
  std. dev.                           0.3307 0.2118
  weight sum                              56     85
  precision                                1      1

hoped
  mean                                0.0357      0
  std. dev.                           0.1856 0.1667
  weight sum                              56     85
  precision                                1      1

hopeless
  mean                                     0 0.0118
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

hopes
  mean                                0.0357      0
  std. dev.                           0.1856 0.1667
  weight sum                              56     85
  precision                                1      1

hoping
  mean                                0.0714 0.0235
  std. dev.                           0.2575 0.1667
  weight sum                              56     85
  precision                                1      1
```

- Adele est plus mélancolique dans ses paroles que Taylor. Les mots distinctifs pour elle sont : “begged”, “defeat”, “defeated”, “distance”, “bitterness”, “bittersweet” (ces deux peuvent être le résultat de ses “espoir”, “déçu”) et des diverses dérivés de “deep”.

```
deep
  mean                                0.0893 0.0588
  std. dev.                           0.2852 0.2353
  weight sum                              56     85
  precision                                1      1

deeper
  mean                                0.0179      0
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

deepest
  mean                                0.0179 0.0118
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

deeply
  mean                                0.0179      0
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

defeat
  mean                                0.0179      0
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

defeated
  mean                                0.0179      0
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1
```

```
bitterness
  mean                                0.0179      0
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

bittersweet
  mean                                0.0179      0
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1
```

```
distance
  mean                                0.0179      0
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1
```

- Adele n’utilise pas du tout le mot “evil”, “diable”, par contre le mot “god”, “dieu”, a beaucoup plus d’importance pour elle que pour Taylor.

```
god
  mean                                0.1429 0.0588
  std. dev.                           0.3499 0.2353
  weight sum                              56     85
  precision                                1      1
```

```
devil
  mean                                     0 0.0235
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

devil'
  mean                                     0 0.0118
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1

devils
  mean                                     0 0.0118
  std. dev.                           0.1667 0.1667
  weight sum                              56     85
  precision                                1      1
```

J48

Il existe plusieurs algorithmes qui utilisent la méthode des arbres de décision et, dans Weka, on utilise le plus connu, le C4.5, mais appelé J48 dans cet outil. Cette méthode nous affiche un ensemble d'arbres de recherche selon les attributs choisis, dans notre cas, il faut classifier les textes de chansons pour nos deux classes en affichant un arbre qui démontre la décision prise par l'algorithme.

L'avantage de cette méthode est sa simplicité : sa procédure d'apprentissage est simple, et sa visualisation est facile à interpréter. Cependant son défaut est son instabilité. Pour avoir un arbre bon, c'est-à-dire pas trop profond et qui donne de bons résultats, il faut choisir les attributs les plus discriminants[5]. Étant donné le petit déséquilibre de notre corpus et également la proximité entre les sujets des contenus des deux classes, les résultats renvoyés par cet outil n'ont pas été avantageux.

Pour le premier test que nous avons fait, nous avons lancé le fichier arff booléen et nous avons choisi de maintenir les paramètres utilisés dans les autres outils : la validation croisée à 10 "folds". L'outil a mis 0.68 secondes pour construire le modèle de l'arbre ci-dessous.

```
car <= 0
|   ah <= 0
|   |   phone <= 0
|   |   |   family <= 0
|   |   |   |   cried <= 0
|   |   |   |   |   screaming <= 0
|   |   |   |   |   |   high <= 0
|   |   |   |   |   |   |   hey <= 0
|   |   |   |   |   |   |   |   shit <= 0
|   |   |   |   |   |   |   |   |   along <= 0
|   |   |   |   |   |   |   |   |   |   white <= 0
|   |   |   |   |   |   |   |   |   |   |   alright <= 0
|   |   |   |   |   |   |   |   |   |   |   |   does <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   smile <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   spinning <= 0: adele (41.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   spinning > 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   am <= 0: taylor (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   am > 0: adele (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   smile > 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   behind <= 0: taylor (3.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   behind > 0: adele (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   does > 0: taylor (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   alright > 0: taylor (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   white > 0: taylor (4.0/1.0)
|   |   |   |   |   |   |   |   |   along > 0: taylor (4.0)
|   |   |   |   |   |   |   |   shit > 0: taylor (8.0)
|   |   |   |   |   |   |   hey > 0: taylor (10.0/1.0)
|   |   |   |   |   |   high > 0: taylor (5.0)
|   |   |   |   |   screaming > 0: taylor (7.0)
|   |   |   |   cried > 0: adele (7.0)
|   |   |   family > 0: taylor (6.0)
|   |   phone > 0: taylor (9.0)
|   ah > 0: taylor (12.0)
car > 0: taylor (13.0)
```

Cet arbre mesure 35 et contient 18 feuilles.

[5] L'analyse discriminante est utilisée pour déterminer les variables qui permettent de discriminer deux ou plusieurs groupes se produisant naturellement.

Les résultats obtenus ne sont pas satisfaisants : 55 paroles ont été mal classifiées. Par conséquent, on a obtenu les valeurs de précision, rappel et F-mesure très faibles, ayant environ 60% de réussite.

Nombre de textes bien classés	86 (60,9929%)
Nombre de textes mal classés	55 (39,0071%)
Précision (moyenne)	0.619 (61,9%)
Rappel (moyenne)	0.610 (61,0%)
F-mesure (moyenne)	0.613 (61,3%)

MATRICE DE CONFUSION		
Adele	**Taylor**	**← classified as**
32	24	**Adele**
31	54	**Taylor**

Puis nous avons testé le fichier arff numérique avec les mêmes paramètres de validation croisée. Tout de suite, une partie des résultats nous a attiré l'attention : il a mis 0.18 secondes pour exécuter. Ce temps fait moins d'un tiers du temps mis par le premier test. En raison de cela, nous avons lancé encore une fois le fichier booléen et maintenant nous avons obtenu un temps de 0.21 secondes, un intervalle plus similaire à celui mis par le fichier numérique. Après avoir testé d'autres fois les deux fichiers, en utilisant aussi d'autres méthodes différentes de classification de textes, nous avons remarqué que Weka met toujours plus de temps pour effectuer la première classification réalisée lorsqu'on ouvre le programme. Ainsi, nous nous demandions s'il faut considérer comme le bon temps celui, le tout premier, ou si on doit considérer plutôt le temps que Weka met lorsqu'il est déjà en utilisation. Nous avons donc décidé de ne plus considérer le temps de la première utilisation pour les raisons suivantes : d'abord, probablement c'est le temps que l'outil met pour démarrer ses fonctionnalités, après, tous les résultats des autres tests ont été obtenus en utilisant le logiciel directement, sans l'ouvrir et le fermer à chaque nouveau test.

Le deuxième test a construit l'arbre suivant :

```
car <= 0
|   took <= 0
|   |   some <= 1
|   |   |   hard <= 0
|   |   |   |   fall <= 1
|   |   |   |   |   let <= 5
|   |   |   |   |   |   whenever <= 0
|   |   |   |   |   |   |   meant <= 0
|   |   |   |   |   |   |   |   heart <= 0
|   |   |   |   |   |   |   |   |   glory <= 0
|   |   |   |   |   |   |   |   |   |   tears <= 0: taylor (40.0)
|   |   |   |   |   |   |   |   |   |   tears > 0
|   |   |   |   |   |   |   |   |   |   |   ' <= 2: adele (3.0)
|   |   |   |   |   |   |   |   |   |   |   ' > 2: taylor (2.0)
|   |   |   |   |   |   |   |   |   glory > 0: adele (3.0/1.0)
|   |   |   |   |   |   |   |   heart > 0
|   |   |   |   |   |   |   |   |   got <= 0
|   |   |   |   |   |   |   |   |   |   one' <= 0: adele (16.0)
|   |   |   |   |   |   |   |   |   |   one' > 0: taylor (3.0)
|   |   |   |   |   |   |   |   |   got > 0: taylor (5.0)
|   |   |   |   |   |   |   |   meant > 0: adele (5.0/1.0)
|   |   |   |   |   |   |   whenever > 0: adele (4.0)
|   |   |   |   |   |   let > 5: adele (5.0)
|   |   |   |   |   fall > 1: adele (5.0)
|   |   |   |   hard > 0: adele (16.0)
|   |   |   some > 1: taylor (10.0)
|   |   took > 0: taylor (11.0/1.0)
car > 0: taylor (13.0)
```

Cet arbre mesure 29 et contient 15 feuilles.

Les résultats obtenus sont encore moins satisfaisants : il a mal classifié 58 chansons. Forcément, les calculs de précision, rappel et F-mesure ont aussi été affectés, ayant un résultat catastrophique.

Nombre de textes bien classés	83 (58.8652 %)
Nombre de textes mal classés	58 (41.1348 %)
Précision	0.589 (58,9%)
Rappel	0.589 (58,9%)
F-mesure	0.589 (58,9%)

MATRICE DE CONFUSION		
Adele	**Taylor**	**← classified as**
27	29	**Adele**
29	56	**Taylor**

SMO

Le SMO (Sequential minimal optimization) est une variante d'un algorithme de la famille SVM (Support Vector Machine). Le SMO est largement utilisé pour l'entraînement des machines à vecteurs de support. Il s'agit d'une classification binaire (avec des étiquettes entre -1 et 1) des vecteurs d'entrée. Le SMO utilise également le **Kernel trick**. "[...] C'est une astuce du noyau qui consiste à transformer un problème non linéairement séparable en un problème linéairement séparable dans un espace de dimension supérieure [...]"[6].

Au début nous avons testé cet algorithme avec le fichier "singers.arff", alors celui avec des valeurs numériques (non booléen). Nous avons aussi testé différents Kernels, mais soit cela résultait en mauvaise classification, soit certains Kernels n'ont pas supporté des valeurs numériques. Les meilleurs résultats sur ce fichier ont été obtenus avec un "Poly Kernel".

Nombre de textes bien classés	114 (80.8511%)
Nombre de textes mal classés	27 (19.1489%)
Précision	0.811 (81,1%)
Rappel	0.809 (80,9%)
F-mesure	0.809 (80,9%)

MATRICE DE CONFUSION		
Adele	**Taylor**	**← classified as**
44	12	**Adele**
15	70	**Taylor**

Remarques : Les résultats ne sont pas les meilleurs, mais ils ne sont pas les pires non plus. C'est intéressant que la quantité des textes mal classés pour chaque classe est similaire. De plus, les résultats sous la représentation binaire des vecteurs est difficilement compréhensible et inanalysable pour un humain.

[6] Isabelle Tellier, Loïc Grobol, Yoann Dupont, *Introduction à la fouille de textes Cours 10 SVM*

Dans notre deuxième test sur SVM, nous avons utilisé le fichier "singers-booleen.arff". De la même manière que pour le premier fichier, nous avons testé plusieurs paramètres, dont Kernel, et cette fois-ci nous en avons deux qui ont marché de façon presque similaire : PolyKernel et RBFKernel. Malgré la similarité, Poly a mieux marché encore une fois, parce qu'il a classé correctement une chanson de plus et il a mis un peu moins de temps (0.07 contre le 0.1 seconde de RBF). C'est une petite différence mais il faut le considérer.

De cette manière, on a obtenu les résultats suivants :

Nombre de textes bien classés	123 (87.234%)
Nombre de textes mal classés	18 (12.766%)
Précision (moyenne)	0.872 (87,2%)
Rappel (moyenne)	0.872 (87,2%)
F-mesure (moyenne)	0.872 (87,2%)

MATRICE DE CONFUSION		
Adele	**Taylor**	**← classified as**
47	9	**Adele**
9	76	**Taylor**

Dans la matrice de confusion, il est intéressant d'observer que l'outil a mal classé exactement la même quantité de chansons pour les deux classes. Cela veut dire qu'il a été plus facile de classifier les fichiers de Taylor, en pourcentage, ils ont été mieux classés que ceux d'Adele. Probablement, ce résultat a eu lieu en raison de l'existence d'un corpus plus grand pour une classe au détriment de l'autre, par conséquent il serait plus facile pour l'algorithme de bien identifier la classe plus représentative.

La prochaine démarche peut confirmer cette hypothèse de plus grande représentativité, donc meilleure classification. Nous continuons dans le même Kernel, sauf

que nous avons changé le paramètre “filterType”. Le paramètre par défaut est “Normalize training data”, pourtant, pour ce nouveau lancement, nous avons choisi le filtre “Standardize training data”. Observons les résultats suivants :

Nombre de textes bien classés	128 (90.7801%)
Nombre de textes mal classés	13 (9.2199%)
Précision (moyenne)	0.914 (84,1%)
Rappel (moyenne)	0.882 (88,2%)
F-mesure (moyenne)	0.908 (90,8%)

MATRICE DE CONFUSION		
Adele	**Taylor**	**← classified as**
53	3	**Adele**
10	75	**Taylor**

Les résultats de la matrice de confusion vérifient l’analyse faite précédemment : comme Taylor est la classe la plus représentative, SMO classe ses textes plus correctement avec moins d’échecs, dans ce cas particulièrement, avec très peu de faux positifs (seulement 3). On réalise aussi la tendance de l’algorithme d’équilibrer le corpus : il classe comme Adele plus de chansons qu’il y a vraiment dans cette classe, de façon à essayer de partager plus égalitairement la quantité de fichiers par classe.

Conclusion

Les critères d'évaluation des classifieurs que nous avons utilisés sont la précision, le rappel, la F-mesure et aussi le nombre de textes bien / mal classés génériquement. D'une certaine manière, les résultats que nous avons obtenus pendant tous les tests nous ont surpris positivement, étant donné que nous avons choisi un sujet assez particulier et difficile à classer, et que nous n'avons pas très bien choisi le corpus. Malgré le problème d'avoir récolté deux classes très similaires, nous avons eu de la chance d'un autre côté : notre corpus n'a pas présenté beaucoup de mots vides (sauf les "yeah-yeah-yeah-yeah"), alors nous avons pu faire des analyses intéressantes du lexique lorsque l'algorithme finissait son travail.

Il était déjà connu que les modèles SVM, le dernier de nos expériences, sont les plus efficaces. Nous ne nous attendions pas aux bons résultats qu'il nous a renvoyés en raison de la difficulté des autres algorithmes pour classer notre corpus. À travers nos tests, nous avons pu constater que le modèle des vecteurs s'en est sorti vraiment performant quant aux critères d'évaluation que nous avons utilisés. Néanmoins, il n'est pas un algorithme très facile à interpréter et à régler, par conséquent nos choix de paramétrage se faisaient un peu au hasard afin de vérifier quelles combinaisons de paramètres nous donneraient les meilleures évaluations dans notre cas.

Bibliographie

https://www.letras.com/

https://www.lyricfind.com/

Isabelle Tellier, Loïc Grobol, Yoann Dupont, *Introduction à la fouille de textes Cours 8 Arbres de décision*, (2022-2023)

Isabelle Tellier, Loïc Grobol, Yoann Dupont, *Introduction à la fouille de textes Cours 9 Naïve Bayes*, (2022-2023)

Isabelle Tellier, Loïc Grobol, Yoann Dupont, *Introduction à la fouille de textes Cours 10 SVM*, (2022-2023)