Projet FMIN311

Extraction de Connaissances à partir de Données

Classification de documents

Geoffrey Dumas
Olivier Saint-Paul
Quentin Philbert
Thibaut Castanié

Mars 2015

Table des matières

1 Constitution du corpus **3**

2 Transformation des données en format .arff **4**
2.1 Présentation du format .arff 4
2.2 Conversion des fichiers texte en fichier .arff unique 5

3 Mise en œuvre des algorithmes de classification **6**
3.1 Définitions 6
3.2 Pré-traitement weka 7

4 Résultats obtenus avec la prise en compte d'informations linguistiques **8**
4.1 Textes bruts 8
4.1.1 Résultats 8
4.2 Textes lemmatisés 9
4.2.1 Résultats 9
4.3 Textes lemmatisés avec analyse morpho-syntaxique (TreeTagger) 10
4.3.1 Résultats en gardant les noms et verbes 10
4.3.2 Résultats en gardant les noms seulement 11
4.3.3 Résultats en gardant les adjectifs seulement 12

5 Analyse approfondie des résultats **14**
5.1 Comparaison des algorithmes de classification 14
5.2 Comparaison des résultats obtenus sur le texte brut, le texte lemmatisé et le texte lemmatisé en ne gardant que les noms et les verbes en se basant sur la Précision, le Rappel et la F-Mesure 15

Introduction

Le but de ce projet consiste à mettre en œuvre et évaluer une méthode de classification de documents par thème ou opinion. Le programme permettant la transformation des documents texte en document de format .arff que nous avons développé est en Java. Les documents sont au format texte.

Nous avons donc cherché un sujet pertinent pour cet exercice qui nous a été proposé : celui du sport. Nous avons constitué un corpus de textes contenant 62 textes : 20 concernant le basketball, 20 concernant le tennis et 22 concernant le rugby. Ce choix de sports est pertinent car chaque sport a ses particularités tout en ayant des points communs, comme un ballon pour le rugby et basketball, des points pour le tennis et le basketball, etc ...

1 Constitution du corpus

Nous avons récupéré la totalité des textes de nos corpus sur le site l'Équipe.fr, en ne récupérant que les articles parlant des résultats et des résumés des rencontres. Dans chaque article, nous avons récupéré le contenu que nous avons collé dans un fichier texte. Puis nous avons supprimé les informations parasites, telles les légendes des images et les textes pour partager sur les réseaux sociaux. Nous avons ainsi récupéré ainsi 20 textes de résultats de tennis, 20 de basketball et 22 de rugby, ce qui représente 62 fichiers textes nommés tennis1, tennis2, tennis3...

2 Transformation des données en format .arff

2.1 Présentation du format .arff

Un fichier .arff (Attribute-Relation File Format) est un fichier texte qui représente une liste d'instances qui partagent un ensemble d'attributs. Il est composé de deux parties principales : une en-tête et un corps contenant les données. L'en-tête contient le nom du fichier précédé de la mention @relation, suivi de la liste des attributs et de leur type, précédés chacun de la mention @attribute.

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}
```

Exemple d'en-tête

La première ligne du corps de données commence par la mention @data. Ensuite, chaque ligne contient le contenu de chaque attribut, séparé par une virgule.

```
@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

Exemple d'un corps de données

2.2 Conversion des fichiers texte en fichier .arff unique

Afin de pouvoir travailler sur le corpus sportif que nous avons créé, il nous faut créer un programme permettant de fusionner et mettre en forme les textes pour respecter la norme d'un fichier.arff.

Pour cela nous avons codé un petit analyseur en Java qui créé l'en-tête du fichier .arff, puis qui récupère le contenu de chaque fichier texte et le met sur une ligne, suivie de sa classe.
Le code du programme est donné en annexe.

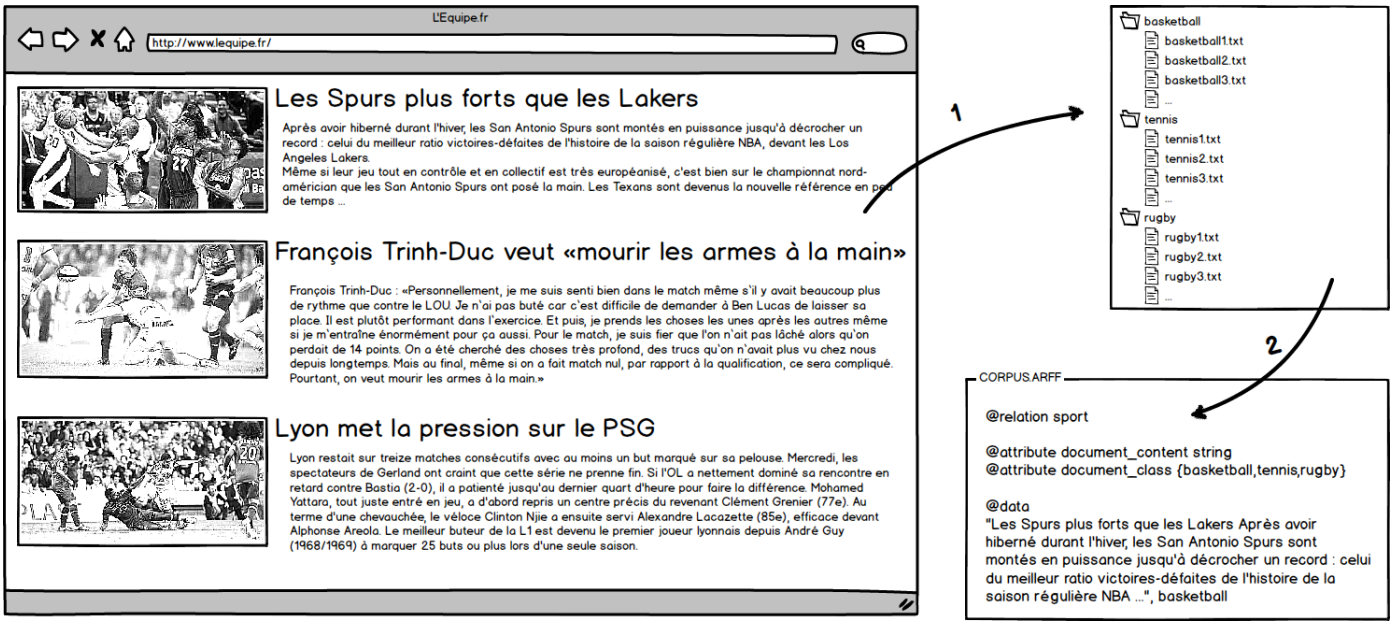


Schéma des actions effectuées pour obtenir un fichier .arff

3 Mise en œuvre des algorithmes de classification

3.1 Définitions

Algorithme NaïveBayes La classification naïve bayésienne est un type de classification probabiliste simple basée sur le théorème de Bayes avec une forte indépendance des hypothèses.

Machines à support de vecteurs (SVM) Les SVM sont des classificateurs qui reposent sur deux idées clés :

- La notion de marge maximale. La marge est la distance entre la frontière de séparation et les échantillons les plus proches. Ces derniers sont appelés vecteurs supports. Dans les SVM, la frontière de séparation est choisie comme celle qui maximise la marge. Le problème est de trouver la frontière séparatrice optimale, à partir d'un ensemble d'apprentissage. Cependant il existe déjà des algorithmes pour résoudre ce problème Transformer l'espace de représentation des données d'entrées en un espace de plus grande dimension, dans lequel il est probable qu'il existe une séparatrice linéaire.

Sous weka, l'algorithme implémentant cette méthode est nommé SMO.

K plus proche voisin (KNN) L'algorithme KNN figure parmi les plus simples algorithmes d'apprentissage artificiel. Dans un contexte de classification d'une nouvelle observation x, l'idée fondatrice simple est de faire voter les plus proches voisins de cette observation. La classe de x est déterminée en fonction de la classe majoritaire parmi les k plus proches voisins de l'observation x. La méthode KNN est donc une méthode à base de voisinage, non paramétrique.
Dans weka, l'algorithme implémentant cette méthode est nommé IBk.

Arbre de décision Cet algorithme utilise une structure d'arbre. L'extrémité de chaque branche représente les différents résultats possibles en fonction des décisions prises à chaque étape. Cet algorithme répartit une population d'individus en groupes homogènes, selon un ensemble de variables discriminantes en fonction d'un objectif fixé et connu.
Dans weka, l'algorithme implémentant cette méthode est nommé j48.

3.2 Pré-traitement weka

Dans le but de "nettoyer" les textes nous avons utilisé une liste de stopwords afin de supprimer les mots français qui sont très présents.

```
alors
au
aucuns
aussi
autre
avant
avec
avoir
a
A
aux
Aux
au
aux
bon
car
ce
Ce
cela
```

Extrait de la liste des stopwords *utilisés*

En utilisant le filtre *StringToWordVector*, nous utilisons la méthode de pondération TF-IDF (Term Frequency - Inverse Document Frequency) afin de donner un poids à chaque mot en fonction de sa fréquence d'apparition dans le texte et dans le corpus. Si un mot est présent de nombreuses fois dans un texte, son TF sera élevé, s'il est présent de nombreuses fois dans le corpus entier, son IDF sera plus faible. Ainsi, on peut évaluer l'importance d'un terme contenu dans un texte, relativement au corpus entier.

4 Résultats obtenus avec la prise en compte d'informations linguistiques

4.1 Textes bruts

4.1.1 Résultats

Pour analyser les textes, nous avons utilisé le filtre StringToWordVector, pour convertir le texte des articles en une liste d'attributs qui représente l'occurrence de chaque mot contenu dans les textes.

Bayes Instances correctement classées : **100%**

a	b	c	classé dans
20	0	0	a = basketball
0	20	0	b = tennis
0	0	22	c = rugby

SMO Instances correctement classées : **98.3871%**

a	b	c	classé dans
19	0	1	a = basketball
0	20	0	b = tennis
0	0	22	c = rugby

IBK, 1 voisin Instances correctement classées : **61.2903%**

a	b	c	classé dans
19	1	0	a = basketball
2	18	0	b = tennis
20	1	1	c = rugby

IBK, 4 voisins Instances correctement classées : **64.5161%**

a	b	c	classé dans
16	4	0	a = basketball
0	20	0	b = tennis
0	18	4	c = rugby

J48 Instances correctement classées : **90.3226%**

a	b	c	classé dans
19	0	1	a = basketball
0	16	4	b = tennis
0	1	21	c = rugby

4.2 Textes lemmatisés

4.2.1 Résultats

Bayes Instances correctement classées : **100%**

SMO Instances correctement classées : **98.3871%**

a	b	c	classé dans
19	0	1	a = basketball
0	20	0	b = tennis
0	0	22	c = rugby

IBK, 1 voisin Instances correctement classées : **59.6774%**

a	b	c	classé dans
19	1	0	a = basketball
2	18	0	b = tennis
20	2	0	c = rugby

IBK, 4 voisins Instances correctement classées : **59.6774%**

a	b	c	classé dans
15	5	0	a = basketball
0	20	0	b = tennis
0	20	2	c = rugby

J48 Instances correctement classées : **91.9355%**

a	b	c	classé dans
19	0	1	a = basketball
0	16	4	b = tennis
0	0	22	c = rugby

4.3 Textes lemmatisés avec analyse morpho-syntaxique (Tree-Tagger)

Afin de savoir quelle catégorie de mot est la plus pertinente, nous avons effectué des tests avec Weka en ne gardant que certaines catégories de mots (cf. plus bas).

a	VER :pres	avoir
reconnu	VER :pper	reconnaître
l'	DET :ART	le
élève	NOM	élève
de	PRP	de
Michael	NAM	Michael
Chang	NAM	Chang
au	PRP :det	au
micro	NOM	micro
du	PRP :det	du
stade	NOM	stade
.	SENT	.
Je	PRO :PER	je
le	PRO :PER	le
félicite	VER :pres	féliciter
.	SENT	.

Extrait du résultat du Tree-Tagger sur le corpus

4.3.1 Résultats en gardant les noms et verbes

Bayes Instances correctement classées : **100%**

SMO Instances correctement classées : **98.3871%**

a	b	c	classé dans
19	0	1	a = basketball
0	20	0	b = tennis
0	0	22	c = rugby

IBK, 1 voisin Instances correctement classées : **61.2903%**

a	b	c	classé dans
18	2	0	a = basketball
0	20	0	b = tennis
20	2	0	c = rugby

IBK, 4 voisins Instances correctement classées : **56.4516%**

a	b	c	classé dans
14	6	0	a = basketball
0	20	0	b = tennis
0	21	1	c = rugby

J48 Instances correctement classées : **91.9355%**

a	b	c	classé dans
19	0	1	a = basketball
0	16	4	b = tennis
0	0	22	c = rugby

4.3.2 Résultats en gardant les noms seulement

Bayes Instances correctement classées : **100%**

SMO Instances correctement classées : **96.7742%**

a	b	c	classé dans
19	0	1	a = basketball
0	19	1	b = tennis
0	0	22	c = rugby

IBK, 1 voisin Instances correctement classées : **54.8387%**

a	b	c	classé dans
14	6	0	a = basketball
0	20	0	b = tennis
2	20	0	c = rugby

IBK, 4 voisins Instances correctement classées : **53.2258%**

a	b	c	classé dans
13	7	0	a = basketball
0	20	0	b = tennis
0	22	0	c = rugby

J48 Instances correctement classées : **88.7097%**

a	b	c	classé dans
19	0	1	a = basketball
0	15	5	b = tennis
1	0	21	c = rugby

4.3.3 Résultats en gardant les adjectifs seulement

Bayes Instances correctement classées : **87.0968%**

a	b	c	classé dans
18	1	1	a = basketball
0	18	2	b = tennis
1	3	18	c = rugby

SMO Instances correctement classées : **75.8065%**

a	b	c	classé dans
17	2	1	a = basketball
0	15	5	b = tennis
4	3	15	c = rugby

IBK, 1 voisin Instances correctement classées : **56.4516%**

a	b	c	classé dans
20	0	0	a = basketball
6	14	0	b = tennis
16	5	1	c = rugby

IBK, 4 voisins Instances correctement classées : **45.1613%**

a	b	c	classé dans
20	0	0	a = basketball
12	8	0	b = tennis
22	0	0	c = rugby

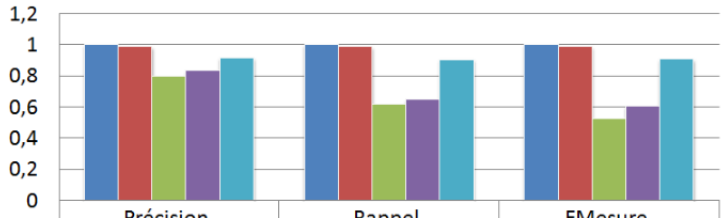
J48 Instances correctement classées : **56.4516%**

a	b	c	classé dans
11	3	6	a = basketball
6	12	2	b = tennis
8	2	12	c = rugby

5 Analyse approfondie des résultats

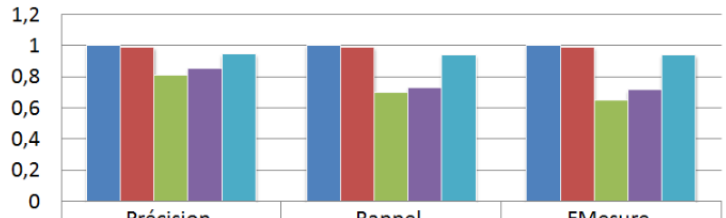
5.1 Comparaison des algorithmes de classification

Textes bruts



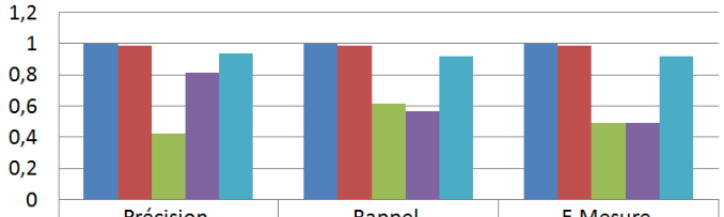
	Précision	Rappel	FMesure
Bayes	1	1	1
SMO	0,985	0,984	0,984
IBK KNN=1	0,795	0,613	0,522
IBK KNN=4	0,831	0,645	0,604
J48	0,913	0,903	0,904

Textes Lemmatisés



	Précision	Rappel	FMesure
Bayes	1	1	1
SMO	0,985	0,984	0,984
IBK KNN=1	0,806	0,694	0,647
IBK KNN=4	0,852	0,726	0,712
J48	0,941	0,935	0,935

Textes Lemmatisés avec analyse morpho-syntaxique



	Précision	Rappel	F-Mesure
Bayes	1	1	1
SMO	0,985	0,984	0,984
IBK KNN=1	0,422	0,613	0,493
IBK KNN=4	0,815	0,565	0,489
J48	0,934	0,919	0,92

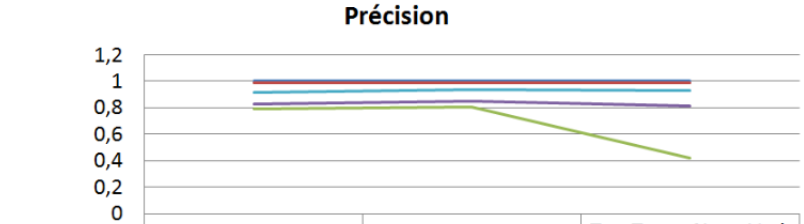
Sur les 3 graphiques précédents, on peut voir que l'algorithme le plus efficace est incontestablement Bayes avec 100% de textes classifiés correctement. Il est talonné par SMO. Le moins efficace étant L'algorithme IBK avec un nombre de voisins réduits(1 ici). Si l'on augmente le nombre de voisins (à 4), on augmente son efficacité.

5.2 Comparaison des résultats obtenus sur le texte brut, le texte lemmatisé et le texte lemmatisé en ne gardant que les noms et les verbes en se basant sur la Précision, le Rappel et la F-Mesure

Précision $P = \frac{nbDocPertinentsTrouvs}{nbTotalTrouvs}$
Plus la précision est élevée et plus ce qui est trouvé est pertinent.

Rappel $R = \frac{nbDocPertinentsTrouvs}{nbTotalDePertinents}$
Plus le rappel est élevé et plus on trouve de documents pertinents.

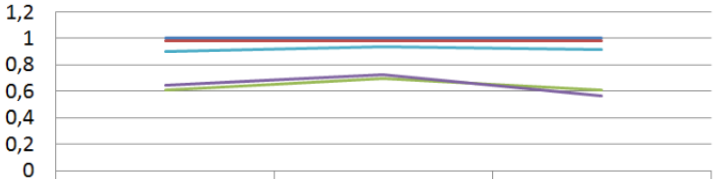
F-Mesure = Mesure combinant le Rappel et la Précision afin d'obtenir des valeurs prenant en compte ces deux mesures. Elle permet donc de nous fournir une mesure d'efficacité pour les différents algorithmes de classification testés.



	Brut	Lemmatisé	TreeTaggerNomsVerbes
Bayes	1	1	1
SMO	0,985	0,985	0,985
IBK KNN=1	0,795	0,806	0,422
IBK KNN=4	0,831	0,852	0,815
J48	0,913	0,941	0,934

Sur le graphique ci-dessus, on peut voir que la précision est moins bonne pour les textes lemmatisés qui ne gardent seulement les noms et les verbes (fait à partir de l'outil TreeTagger) que pour les textes bruts. Cependant, la précision en ce qui concerne les textes lemmatisés est légèrement meilleure que pour les textes bruts. La lemmatisation diminuerait donc le bruit.

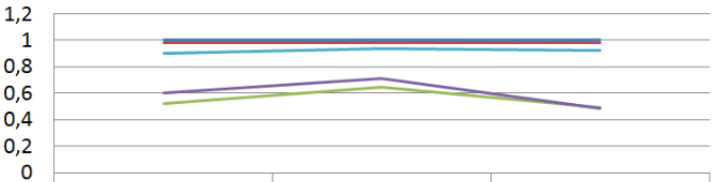
Rappel



	Brut	Lemmatisé	TreeTaggerNomsVerbes
Bayes	1	1	1
SMO	0,984	0,984	0,984
IBK KNN=1	0,613	0,694	0,613
IBK KNN=4	0,645	0,726	0,565
J48	0,903	0,935	0,919

Pour ce qui est du rappel, les textes lemmatisés sont encore une fois ceux qui se comportent le mieux. Il y a donc moins de silence avec les textes lemmatisés.

F-Mesure



	Brut	Lemmatisé	TreeTaggerNomsVerbes
Bayes	1	1	1
SMO	0,984	0,984	0,984
IBK KNN=1	0,522	0,647	0,493
IBK KNN=4	0,604	0,712	0,489
J48	0,904	0,935	0,92

La F-mesure confirme bien les résultats précédents, à savoir que la lemmatisation apporte un plus quand à la bonne classification des textes. Néanmoins, la catégorie grammaticale n'apporte rien.

Globalement, la précision est moins bonne mais le rappel est légèrement meilleur lorsque le paramètre K de l'algorithme IBK (nombre de voisins) est faible.

	Brut	Lemmatisé	TreeTaggerNomsVerbes
Bayes	100	100	100
SMO	98,3871	98,3871	98,3871
IBK KNN=1	61,2903	59,6774	61,2903
IBK KNN=4	64,5161	59,6774	56,4516
J48	90,3226	91,9355	91,9355

Annexes

Code java permettant de convertir les fichiers textes en un fichier .arff

```
package ArffCreatorFromTextFiles;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintStream;

public class Main {
  public static void charger(String path) {
    // on set le system.out en mettant le fichier arff
    // que l'on souhaite en sortie
    try {
      System.setOut(new PrintStream(
        new FileOutputStream(path+"/../fichier.arff"))
      );
    }
    catch (FileNotFoundException e) {
      e.printStackTrace();
    }

    // on lui ajoute l'entete
    System.out.println("@relation sport");
    System.out.println("");
    System.out.println("@attribute document_content string");
    System.out.println(
      "@attribute document_class {basketball,tennis,rugby}"
    );
    System.out.println("");
    System.out.println("@data");

    File corpus = new File(path);
    String [] listeRep;
    listeRep = corpus.list();

    // pour chaque repertoire...
    for (int i = 0; i < listeRep.length; i++) {
```

```
        File sport = new File(path+"/"+listeRep[i]);
        String[] listeTextes = sport.list();
        String line;
        BufferedReader buff = null;
        // ... on prends chaque texte ...
        for (int j = 0; j < listeTextes.length; j++) {
          String content = "";
          String titre = listeTextes[j];
          try {
            //...on recupere la ligne du fichier texte
            //et on la "nettoie" un peu...
            buff = new BufferedReader(new FileReader(sport+"/"+titre));
            while((line = buff.readLine()) != null) {
              line += "␣";
              line = line.replace("\"", "");
              content += line;
            }
          }
          catch (FileNotFoundException e) {
            e.printStackTrace();
          }
          catch (IOException e) {
            e.printStackTrace();
          }
          //...et on y ajoute le contenu du texte ainsi
          //que sa categorie dans le fichier arff
          System.out.println('"'+content+'"'+","+sport.getName());
        }

      }
    }

    public static void main(String[] args){
      charger("/auto_home/qphilbert/EC/corpus");
    }
}
```