TER : Intégration et optimisation d'algorithmes de classifications supervisées pour Weka

ALIJATE Mehdi - NEGROS Hadrien- TURKI Batoul

Université Montpellier 2 - LIRMM

23 février 2014

Abstract

Résumé

Ce sujet vise à intégrer et à optimiser des algorithmes de classifications supervisées de documents dans la suite logiciel WEKA. Ces algorithmes sont issus de travaux de recherche menés récemment au sein du LIRMM.

Sommaire

Sommaire

Introduction

- Ce TER vise à intégrer des algorithmes de classifications supervisées de documents dans la suite logiciel WEKA
- Intégrant de nouvelles solutions adaptées aux faibles quantités de données textuelles.
- Se basant sur de nouvelles pondérations.

La classification

La classification est le processus visant à associer un document à la classe le représentant le mieux.

Exemple

Classer des résumés de films selon leur genre. (Policier, Comédie, etc...)

Vocabulaire

Attribut	Ou **terme unique** est un élément d'un document.
Document	Un ensemble d'attributs (ex : un résumé)
Classe	Un ensemble de documents liés entre eux. (ex : genre cinématographique)

Sommaire

Besoins

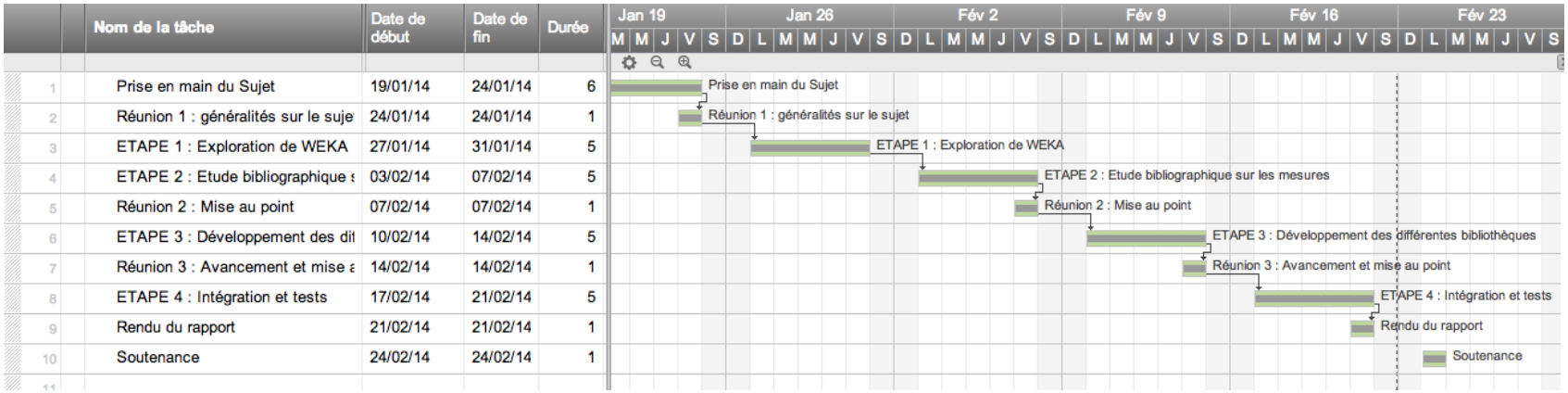
Besoins

- Prise en main de Weka
- Développement des différentes bibliothèques en java
- L'intégration dans l'écosystème Weka

Organisation

- Plusieurs réunions
- Un outil collaboratif pour la gestion du projet : Github
- Mises au point régulières

Diagramme de Gantt :



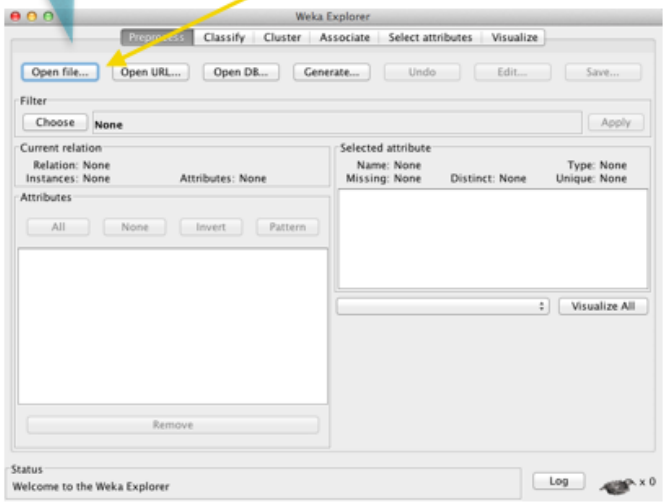
Sommaire

Qu'est-ce que Weka ?

Weka

- Suite de logiciel d'apprentissage automatique
- Développé en Java, à l'université de Waikato en Nouvelle-Zélande
- Librement disponible sous la licence publique générale GNU
- Très portable car il est entièrement implémenté en Java
- Contient une collection complète de préprocesseurs de données et de techniques de modélisation
- Facile à utiliser (interface graphique)

Weka




```
@attribute truand numeric
@attribute force|forces numeric
@attribute phénomène numeric
@attribute combat numeric
@attribute personne numeric
@attribute enquête numeric
@attribute braquage numeric
@attribute classeFilm {Policier,Fantastique,Comedie}

@data
0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,Policier
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Policier
0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Policier
0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Policier
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,Policier
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,Policier
0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Policier
0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Policier
0,0,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Policier
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Fantastique
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Fantastique
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Fantastique
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,Fantastique
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,Fantastique
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,Fantastique
0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Fantastique
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,Fantastique
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,Fantastique
```

Qu'est-ce qu'un fichier arff ?

Format arff

- Attribute-Relation File Format (créé également par l'universirté Waikato)
- Weka utilise (entre autres) le format arff comme fichier de données
- Il s'agit d'une liste d'exemple auxquels sont associées des valeurs d'attributs

```
@relation Rel
@attribute "a" numeric
@attribute "b" numeric
@attribute "c" numeric
@attribute "d" numeric
@attribute "e" numeric
@attribute "f" numeric
@attribute "classe" {A,B,C}
@data
2,3,2,1,0,1,A
1,2,3,2,3,2,A
3,1,1,2,1,0,A
3,3,2,2,3,1,B
2,1,0,0,2,0,B
1,2,1,0,1,3,B
0,0,0,3,3,2,C
```

Exploration de Weka

- L'API Weka
- L'utilisation des classes
- Ajout d'un algorithme dans Weka

A la fin de cette étape

- Méthodes et classes ciblées
- Le Package *weka.classifiers*
- Le classifieur *NaiveBayesMultinomial*
- Pour l'ajout d'algorithme : Le package *weka.gui*

Sommaire

Nouvelles méthodes de classifications

- Différentes pondérations pour la construction des nouveaux classifieurs
  1. Des mesures intra-classe.
  2. Des mesures inter-classe.

Ces mesures sont inspirées du TF-IDF et ont étés développées au LIRMM.
Elles sont définies dans l'article :

De nouvelles pondérations adaptées à la classification de petits volumes de données textuelles.
De *M. Roche, F. Bouillot, P. Poncelet*
EGC 2014

Pondérations intra-classe

- Les pondérations que nous définissons ci-après sont dites **intra-classe**
- Les différentes valeurs que nous utilisons pour les calculer sont dépendantes d'une classe.

Intra-classe document

Cette mesure dépend du nombre de documents contenant le terme dans la classe.

$$inner\text{-}weight_{ij}^{Df} = \frac{DF_{ti}^{j}}{|d_j|}$$

Avec :

- DF_{ti}^{j} : Nombre de documents contenant le terme t_i dans la classe C_j
- $|d_j|$: Nombre de documents dans C_j

Intra-classe terme

Cette mesure dépend du nombre d'occurrences du terme dans la classe.

$$inner\text{-}weight_{ij}^{Tf} = \frac{TF_{ti}^{j}}{|n_j|}$$

Avec :

- TF_{ti}^{j} : Nombre d'occurrences du terme t_i dans la classe C_j
- $|n_j|$: Nombre de termes total dans la classe C_j

Pondérations inter-classe

- Les pondérations inter-classes utilisent des valeurs calculées à partir de l'ensemble du corpus.
- Depuis les classes extérieures à celle qui nous intéresse.

Inter-classe terme

Cette mesure dépend du nombre de classes contenant le terme.

$$inter\text{-}weight_{ij}^{class} = log_2 \frac{|C|}{C_{ti}}$$

Avec :

- $|C|$: Nombre de classes
- C_{ti} : Nombre de classes contenant le terme t_i

Formule inter-classe document

Cette mesure dépend du nombre de documents extérieurs à la classe contenant le terme.

$$inter\text{-}weight_{ij}^{doc} = log_2 \frac{|d \notin C_j| + 1}{|d : t_i \notin C_j| + 1}$$

Avec :

- $|d \notin C_j|$: Nombre de documents n'appartenant pas à la classe C_j
- $|d : t_i \notin C_j|$: Nombre de documents n'appartenant pas à la classe C_j qui contient t_i
- En ajoutant 1, permet de prévenir le cas où t_i est uniquement utilisé dans C_j (quand $|d : t_i \notin C_j| = |d : t_i| - |d : t_i \in C_j| = 0$)

Algorithmes de classifications

- Nous avons implémenté un classifieur *Naive Bayes* et *Class-Feature-Centroid*.
- Pour calculer la probabilité w_{ij} d'un terme i dans une classe j, nous avons combiné les différentes pondérations de 4 façons :

Les quatres pondérations

- $w_{ij}^{Tf-Class} = inner\text{-}weight_{ij}^{Tf} \times inter\text{-}weight_{ij}^{class}$
- $w_{ij}^{Df-Class} = inner\text{-}weight_{ij}^{Df} \times inter\text{-}weight_{ij}^{class}$
- $w_{ij}^{Tf-Doc} = inner\text{-}weight_{ij}^{Tf} \times inter\text{-}weight_{ij}^{doc}$
- $w_{ij}^{Df-Doc} = inner\text{-}weight_{ij}^{Df} \times inter\text{-}weight_{ij}^{doc}$

Paramètres : α, β

Nous avons aussi mis en place une combinaison de ces mesures dépendantes de deux paramètres $\alpha, \beta \in [0, 1]$:

$$w_{ij}^{\alpha\beta} = (\alpha \times inner\text{-}weight_{ij}^{Tf} + (1-\alpha) \times inner\text{-}weight_{ij}^{Df}) \times (\beta \times inter\text{-}weight_{ij}^{class} + (1-\beta) \times inter\text{-}weight_{ij}^{doc})$$

Sommaire

Développement

buildClassifier() C'est la méthode dans laquelle est calculé le tableau des w_{ij} (La probabilité d'un mot par rapport à une classe).

distributionForInstance(Instance) Renvoie les probabilités du document (Instance) en entrée pour chacune des classes du corpus.

NaiveBayesMultinomialTER

Version 1 : implémentant les quatre pondérations.

Version 2 : paramètrable avec α et β

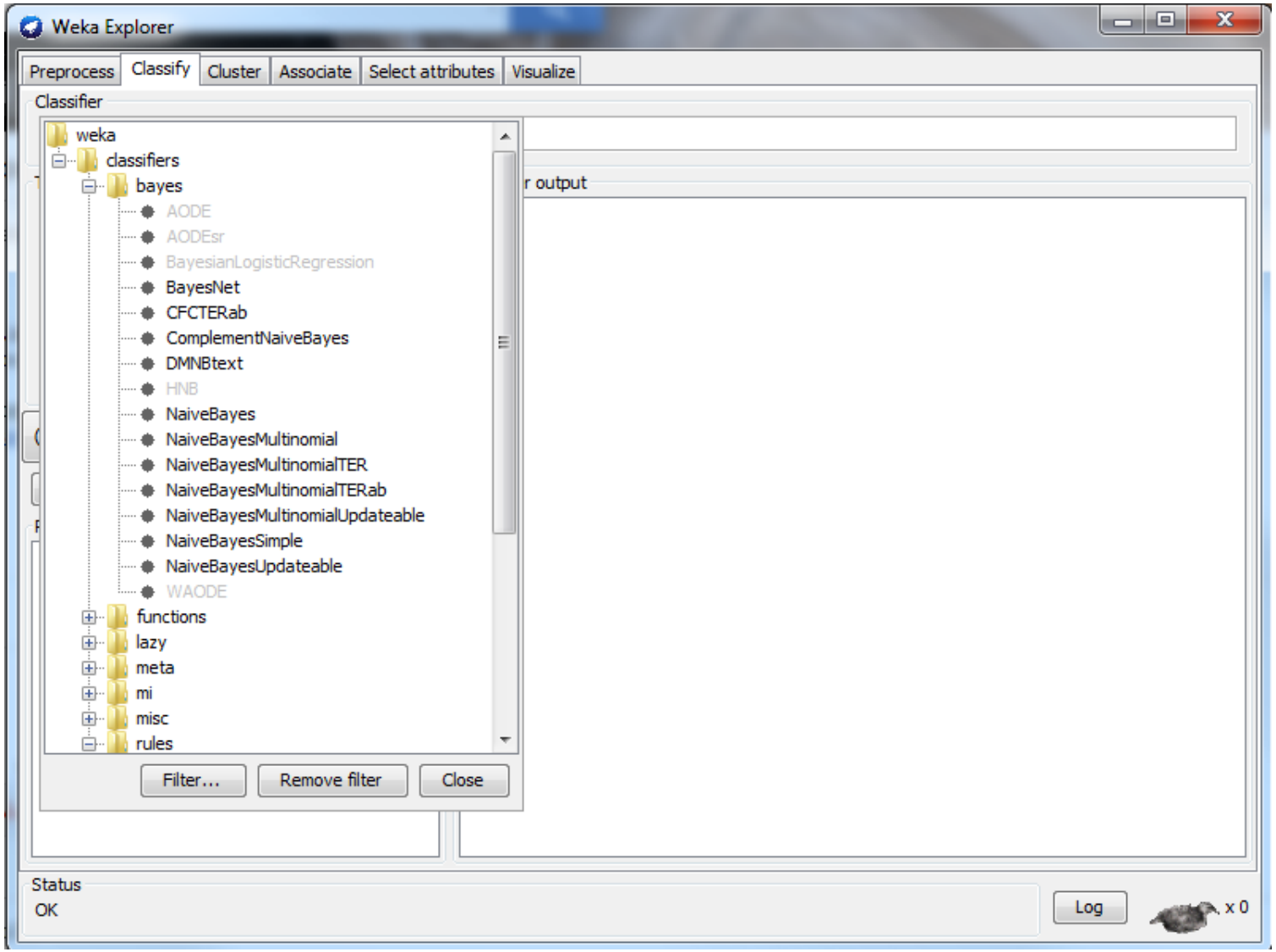
CFCTERab

- Représentation des classes comme des vecteurs (exemple : $\vec{C_j} = (0.1, 0.3, 0.2, 0)$)
- Représentation des documents comme des vecteurs (exemple : $\vec{d} = (0.1, 0, 0.2, 0)$, le terme 2 n'apparait pas dans le document)
- Mesure de la proximité entre les vecteurs en utilisant la proximité cosinus :

$$simcos(\vec{u}, \vec{v}) = arccos(\frac{\vec{u}.\vec{v}}{\|\vec{u}\|.\|\vec{v}\|})$$

Sommaire

Intégration



Tests

Classification de résumés de films.

Jeux de données

Nos jeux de données :

- `test3classes.arff` : **150** instances et **41** attributs (une selection d'attributs **SubsetEval** a été faite dessus), avec **3** classes : Policier, Fantastique, Comédie.
- `test5classes.arff` : **248** instances et **5082** attributs au complet (sans selection d'attributs), avec **5** classes : Thriller, Western, Guerre, Policier, Sciences.

Résultats des tests : NaiveBayesMultinomialTER

NBMultinomialTER/fichierTest	$Nb^{Df-Class}$	NBMultinomial
test3classes.arff	**67%**	66%
test5classes.arff	**68%**	63%

Expérimentations avec $Nb^{Df-Class}$ et comparaison avec NBMultinomial

Résultats des tests : NBTER $\alpha\beta$ et CFCTER$\alpha\beta$

Algo/FichierTest	α	β	NBMTER$\alpha\beta$	CFCTER$\alpha\beta$	NBMulti
test3classes.arff	0.0	1.0	**67%**	68%	66%
	0.6	0.6	66%	**74%**	
	0.7	0.3	66%	73%	
test5classes.arff	0.0	1.0	**67%**	68%	63%
	0.6	0.6	65%	**70%**	
	0.7	0.3	58%	60%	

Expérimentations avec différentes valeurs de α et β pour NBTER$\alpha\beta$ et CFCTER$\alpha\beta$

Sommaire

Conclusion

Ce TER nous a permis de :

- Prendre en main Weka
- Comprendre les nouvelles mesures de classification
- Intégrer les algorithmes dans l'écosystème Weka

Perspective

Implémenter de nouvelles métriques pour CFC (exemple : Distance de Jaccard)

Démonstration

Sommaire