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

Introduction

Introduction

- Ce TER vise à intégrer des algorithmes de classifications supervisées de documents dans la suite logiciel WEKA
- Intégrant de nouvelles pondérations adaptées
- Se basant sur un nouveau modèle de classification à partir d'un faible nombre de documents

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

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*

Nouvelles méthodes de classifications

- Différentes pondérations pour la construction des nouveaux classifieurs
  1. Les mesures intra-classe inspirées du TF-IDF
  2. Les mesures inter-classe développées au LIRMM

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

Inter-classe document

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

Formule inter-classe document

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

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
- $|d|$: Nombre de documents dans l'ensemble des classes
- $|d \in C_j|$: Nombre de documents de la classe C_j
- $|d : t_i|$: Nombre de documents dans l'ensemble des classes contenant le terme t_i
- $|d : t_i \in C_j|$: Nombre de documents de la classe C_j qui contiennent 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 $\alpha, \beta \in [0, 1]$:

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

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.

NaiveBayesMulinomialTER

Version 1 : implémentant les quatre pondérations.

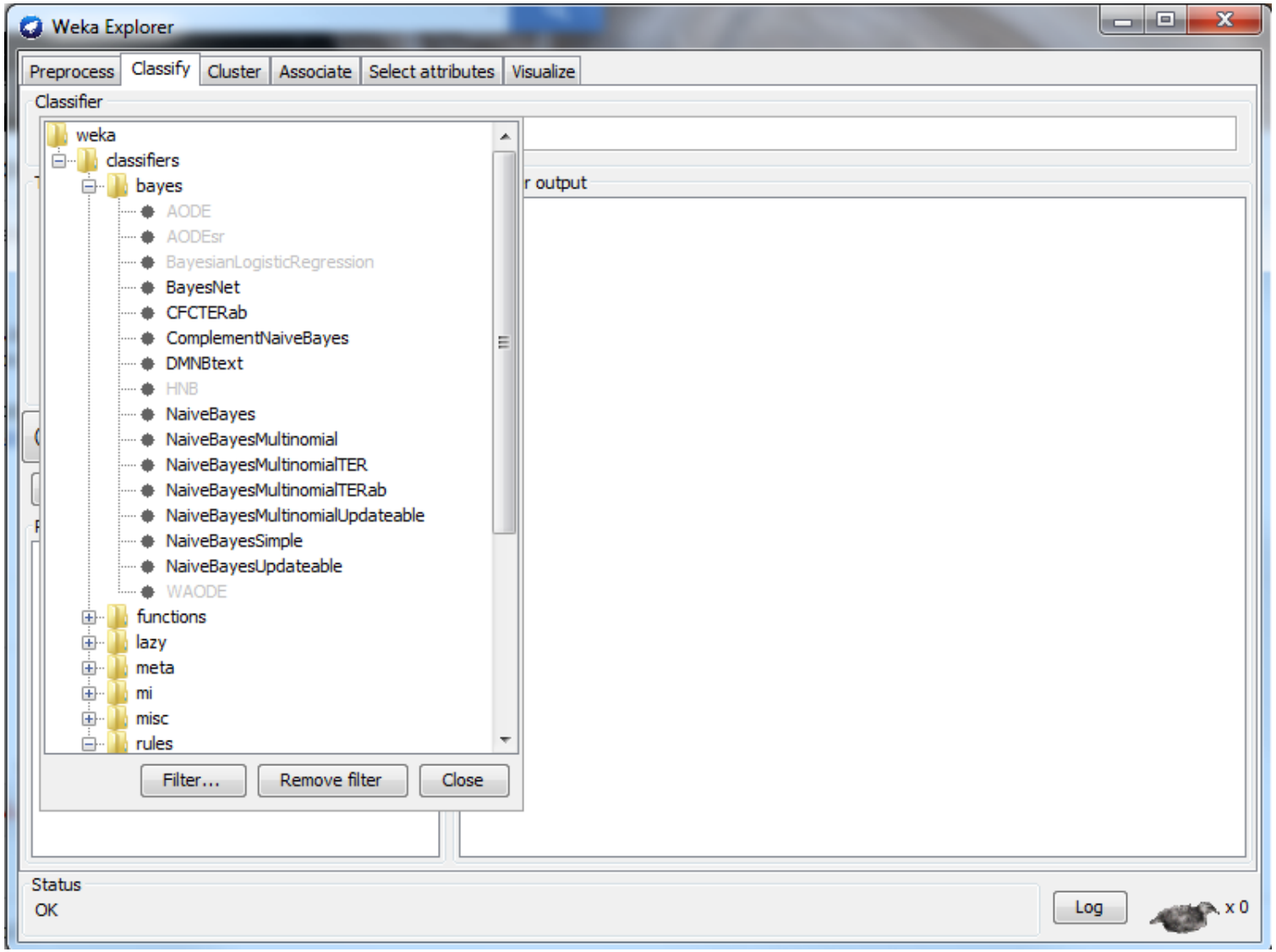
Version 2 : paramètrable avec α et β

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 :

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

Intégration



Jeu de données

Nos jeu de données :

- `test3classes.arff` : **150** instances et **41** attributs (une selection d'attributs 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$\alpha\beta$

Ce TER nous a permis de :

- Prendre en main de 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