Rapport du TER GMIN401 :

Intégration et optimisation d'algorithmes de classifications supervisées pour Weka

Par : ALIJATE Mehdi - NEGROS Hadrien - TURKI Batoul

31 Janvier 2014

Table des matières

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.

1 Introduction

La classification de documents est le mécanisme consistant à classer automatiquement des ressources la classe prédéfinie lui correspondant le mieux.
Plusieurs formes de classification existent (par genre, par opinion, par thème...etc), et se font via des algorithmes de classifications spécifiques. Ceux-ci se basent sur des méthodes principalement numériques (probabilistes), avec des algorithmes utilisant les mathématiques ou basés sur la recherche d'information.
Ce TER vise justement à intégrer des algorithmes de classifications supervisées de documents dans la suite logiciel WEKA [1], se basant sur un nouveau modèle de classification à partir d'un faible nombre de document, intégrant de nouvelles pondérations adaptées.
Tout d'abord, il faudra explorer l'API de WEKA, pour prendre en main du code source, la maniabilité des classes et explorer une méthode d'ajout d'un algorithme de classification. Ensuite, nous nous pencherons sur le développement des différentes classes en établissant une méthodologie concrétisant le travail mené au laboratoire du LIRMM, s'en suivra une phase d'intégration et de tests.

2 Exploration de WEKA

Après la réunion du 24/01/14, nous avons établi un plan de travail pour bien mener et répartir les tâches de ce TER. Il a été décidé de le diviser en trois grandes parties. La première, qui est décrite ci-dessous consiste à explorer et prendre en main l'API de WEKA, afin de pouvoir y rajouter les algorithmes que l'on aura développé lors de la deuxième partie, et qui seront testés et intégrés lors de la troisième.

2.1 L'API Weka/Sources avec Eclipse

Pour explorer l'API, nous nous sommes aidés de l'IDE Eclipse, qui permet facilement parcourir les sources d'une librairie externe. Après avoir étudié l'arborescence des classes de l'API, nous avons pu cibler les différentes classes et méthodes qui nous intéressent, et étudié leurs fonctionnement. Nous nous sommes aidé de ce wiki [2].

2.2 L'utilisation des classes

Une fois familiarisés avec l'API Weka, on a creusé un peu plus du côté des classes qui pourraient nous être utiles pour ce TER. Il s'agit des certaines classes présentes dans le package "weka.classifiers". En effet, notre but étant d'intégrer des algorithmes de classification, il est utile de savoir comment tournent les algorithmes de classifications, leur paramétrage et l'architecture pour organiser les ressources pour ces derniers.
Quelques tests ont été menés notamment pour bayes naif multinomial, que nous avons fait tourné sur différentes données, et avec différentes options.

2.3 Ajout d'un algorithme dans Weka

Après avoir étudié en détail la classe *NaiveBayesMultinomial*, nous avons remarqué que le calcul des pondérations (dans l'implémentation de Weka, seul la mesure intra-classe Tf est utilisée) se fait dans la méthode **buildClassifier**. Nous allons donc créer une sous classe de *NaiveBayesMultinomial*, contenant une méthode surchargeant **buildClassifier** dans laquelle nous calculerons toutes les pondérations supplémentaires.
Une fois tout cela creusé et vu en détails, il faudra intégrer l'algorithme dans l'écosystème de Weka, c'est à dire, pour le rendre disponible dans l'Explorateur, expérimentateur, etc . Weka prend en charge les classes dérivées dans le package, ceci est géré par le *GenericPropertiesCreator*. Il faudra donc dire à Weka où trouver

1. Weka est une suite populaire de logiciels d'apprentissage automatique. Écrite en Java, développée à l'université de Waikato, Nouvelle-Zélande. Weka est un Logiciel libre disponible sous la Licence publique générale GNU.
2. http://weka.wikispaces.com/

notre nouveau classificateur et il s'occupera de l'afficher dans la *GenericObjectEditor*.
Nous y reviendrons plus en détails lors de la troisième étape de notre TER : L'intégration des algorithmes dans WEKA.

3 De nouvelles méthodes de classification

Dans cette partie, nous allons vous présenter les différentes pondérations que nous allons utiliser pour construire nos classifieurs. Nous allons d'abord définir les mesures intra-classe inspirées du TF-IDF, puis les mesures inter-classe développées au *LIRMM*. Ces mesures vont nous permettre de définir si un terme (un élément d'un document) est plus ou moins représentatif de la classe.
Toutes ces mesures ont étés définies dans l'article *De nouvelles pondérations adaptées à la classification de petits volumes de données textuelles.* [1].

3.1 Pondérations intra-classe

Les pondérations que nous définissons ci-dessous sont dites **intra-classe** car les différentes valeurs que nous utilisons pour les calculer sont dépendante 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

3.2 Pondérations inter-classe

Les pondérations inter-classes en revanche 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.

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

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 contient t_i
- En ajoutant 1, permet de prévenir le cas où t_i est uniquement utilisé dans C_j (quand $|d : t_i \notin C_j| = |d : t_i| - |d : t_i \in C_j| = 0$)

3.3 Algorithmes de classifications

Un algorithme de classification permet de calculer la probabilité de l'appartenance d'un document aux différentes classes du corpus, et donc de l'affecter à la plus probable. Nous allons implémenter un classifieur *Naive Bayes* et *Class-Feature-Centroid*[1] en utilisant les mesures définies plus haut. Pour calculer la probabilité w_{ij} d'un terme i dans une classe j, nous allons combiner les différentes pondérations de 4 façons :
- $w_{ij}^{Tf-Class} = inner\text{-}weight_{ij}^{Tf} \times inter\text{-}weight_{ij}^{class}$
- $w_{ij}^{Df-Class} = inner\text{-}weight_{ij}^{Df} \times inter\text{-}weight_{ij}^{class}$
- $w_{ij}^{Tf-Doc} = inner\text{-}weight_{ij}^{Tf} \times inter\text{-}weight_{ij}^{doc}$
- $w_{ij}^{Df-Doc} = inner\text{-}weight_{ij}^{Df} \times inter\text{-}weight_{ij}^{doc}$

Nous allons aussi mettre en place une combinaison de ces mesures dépendante de deux paramètres α et β :

$$w_{ij}^{\alpha\beta} = (\alpha \times innerweight_{ij}^{Tf} + (1-\alpha) \times innerweight_{ij}^{Df}) \times (\beta \times interweight_{ij}^{class} + (1-\beta) \times interweight_{ij}^{doc})$$

On remarque qu'en faisant varier α et β, on peut retrouver les quatre premières formules.

4 Développement des différentes classes

4.1 Méthodologie

//TODO

4.2 Extension de Naive Bayes Multinomial

//TODO

4.3 Class-Feature-Centroide

//TODO

5 Intégration et tests

5.1 Intégration dans l'écosystème de Weka

//TODO

5.2 Tests

//TODO

5.3 Résultats

//TODO

6 Discussion et Conclusion

//TODO

7 Sources

//TODO

Références

[1] M.Roche F.Bouillot, P.Poncelet. De nouvelles pondérations adaptées à la classification de petits volumes de données textuelles. In *Actes des 14ièmes Journées Francophone "Extraction et Gestion des Connaissances" (EGC 2014)*, 2014.