IFT3335 – TP2

Utiliser la classification pour la désambiguïsation de sens de mots

Ce TP est à réaliser en groupe de 2 personnes (ou seul).
Date limite de remise : 2 mai, avant 23 :39.
Chaque jour de retard aura une pénalité de 10% de la note.
Ce TP correspond à 15% de la note globale.

Ce TP a pour but de pratiquer les algorithmes de classification et de les utiliser pour la désambiguïsation de sens de mots. Les algorithmes de classification sont déjà implantés dans l'outil Weka. Votre travail dans ce TP consiste à utiliser Weka sur des collections de données et à examiner l'impact de différents algorithmes et les différentes caractéristiques (features).

1. Weka

Weka est un package d'apprentissage Open Source très populaire. Il implante différents algorithmes de data-mining et d'apprentissage (Naïve Bayes, Arbre de décision, SVM, réseau de neurones, entre autres). Il offre une interface GUI conviviale pour manipuler et inspecter les données et visualiser les résultats. Ce package peut fonctionner sur les plateformes Linux, Windows et Mac.

Le package (version 3-8) est téléchargeable sur ce site :

http://www.cs.waikato.ac.nz/ml/weka/

Vous êtes conseillés fortement de lire le tutoriel de Weka et la documentation de Weka.

Il y a aussi une série de cours en ligne par Ian Witten sur Weka (http://www.cs.waikato.ac.nz/ml/weka/mooc/dataminingwithweka/), et une autre vidéo d'introduction par Brandon Weinberg (http://www.youtube.com/watch?v=IY29uC4uem8). Ces vidéos fournissent une introduction rapide sur Weka.

2. Préparatifs

Pour vous familiariser avec Weka, explorer le librement. Explorer au moins les fonctions suivantes :

1. Preprocess

Ceci vous permet de charger les données et faire des prétraitements sur les données, e.g. la sélection des données à traiter, appliquer des filtres pour transformer les données et les attributs, etc.

Pour commencer, ouvrez un ensemble de données existant dans le package (e.g. data/weather.numeric.arff). Vous devez voir un ensemble d'attributs. En cliquant sur chaque attribut, vous pouvez voir la distribution de valeurs dans les données.

2. Classify

Une fois les données chargées, vous pouvez maintenant choisir un algorithme et l'appliquer sur les données. Pour essayer, choisissez (avec choose) une méthode (e.g. tree->J48, qui correspond à l'arbre de décision présenté dans le cours).

Choisissez d'abord « Use training set » dans « Test options » (qui tente d'entraîner un arbre en utilisant tous les exemples d'entraînement), et cliquer sur « Start ». Vous verrez à droite le résultat d'entraînement, avec l'arbre obtenu, ainsi que le résultat de classification sur ce même ensemble de données.

Vous pouvez ensuite choisir Cross-validation, Folds = 4, pour voir l'effet de validation croisée, c'est-à-dire de découper les données en 4 parties, et on fait 4 expériences en utilisant, à tour de rôle, une partie comme test et les 3 autres parties pour l'entraînement. La validation croisée est à utiliser si vous n'avez pas déjà une collection avec les sous-ensembles d'entraînement et de test déjà séparés. La validation croisée va faire cette séparation de façon aléatoire.

3. Vous pouvez maintenant tenter de sélectionner des attributs à utiliser pour la classification dans « Select attributes ». Il y a différentes méthodes pour sélectionner un sous ensemble d'attributs à utiliser dans la classification. Ceci est très utile quand vos données sont très bruitées, avec beaucoup d'attributs qui n'aident pas à la classification. Un nettoyage (une sélection) est très bénéfique dans ce cas. Cette sélection aide aussi à accélérer les traitements.

Pour essayer, choisissez dans « Attribute Evaluator » la méthode InfoGainAttributeEval (la sélection basée sur le gain d'information), et dans « Search Method » la méthode Ranker – qui ordonne les attributs selon leur valeur. En cliquant sur Ranker (une fois c'est choisi), on peut préciser les critères de sélection, par exemple, en fixant un seuil, ou en fixant un nombre d'attributs à garder.

Jouer librement avec les données Reuters incluses dans le package Weka. Notamment, vous devez transformer un texte en un ensemble d'attributs (chaque mot = 1 attribut). Après cette transformation, vous allez pouvoir utiliser les algorithmes de classification. Certaines options vous sont offertes dans cette transformation (avec filter) : filters→ unsupervised→NominalToString transforme un attribut en un string textuel, filters→ unsupervised→StringToWordVector transforme un texte en un ensemble d'attributs mots. De plus, dans ce filtre, il vous serait aussi possible de préciser si le résultat de cette transformation produit un ensemble d'attributs (mots) binaire (présent ou absent) ou avec un poids numérique (fréquence, tf transformé et avec idf).

Une pratique courante dans le domaine de classification de textes et de recherche d'information est de tronquer les mots pour ne garder que les racines. Par exemple, le mot « computer » sera tronqué en « comput ». Ceci a pour but de créer une représentation unique pour une famille de mots semblables (computer, computing, compute, computes, computed). Ce processus est appelé « stemming » (troncature). Il y a des méthodes de stemming standard disponibles, dont la méthode de Porter (appelé SnowballStemmer dans Weka). Le programme correspondant peut être téléchargé à :

http://weka.wikispaces.com/file/view/snowball-20051019.jar/82917267/snowball-20051019.jar

Si le package Weka que vous avez téléchargé ne fournit pas la troncature SnowballStemmer, vous pouvez l'intégrer dans Weka en faisant ceci :

```
java -cp /Users/as/Documents/Work/weka-3-8/snowball-20051019.jar:weka-3-8/weka.jar weka.gui.Main
```

Lisez le tutoriel et regarder les vidéos pour en apprendre plus. Référez-vous aussi à la présentation de la démonstration sur la classification (et Weka).

3. La tâche de désambiguïsation de sens de mots

La désambiguïsation de sens de mots consiste à déterminer le sens d'un mot ambigu. C'est une tâche de base pour la compréhension de textes. On effectue cette tâche souvent en utilisant une approche de classification : On suppose qu'on dispose d'un semble de textes (phrases) contenant des occurrences du mot ambigu, et que le sens de chaque occurrence du mot est annoté manuellement. En utilisant ces textes comme exemples, on vise à entraîner un classifieur. Ce classifieur sera appliqué aux nouveaux textes de test.

Pour ce TP, Weka est encore utilisé comme l'outil de classification. Mais il faut que vous fassiez un petit programme pour transformer les textes de départ en un fichier de forma .arff utilisé par Weka.

3.1. Corpus

Pour ce TP, nous allons utiliser un ensemble de phrases annotées, contenant le mot ambigu *interest*, qui peut correspondre à 6 sens différents, selon le dictionnaire Longman :

Sense 1 = 361 occurrences (15%) - readiness to give attention
Sense 2 = 11 occurrences (01%) - quality of causing attention to be given to
Sense 3 = 66 occurrences (03%) - activity, etc. that one gives attention to
Sense 4 = 178 occurrences (08%) - advantage, advancement or favor
Sense 5 = 500 occurrences (21%) - a share in a company or business
Sense 6 = 1252 occurrences (53%) - money paid for the use of money

Le texte annoté est le résultat d'une analyse de partie-de-discours + annotation de sens du mot ambigu. Voici un exemple :

```
[ yields/NNS ] on/IN [ money-market/JJ mutual/JJ funds/NNS ]
continued/VBD to/TO slide/VB ,/, amid/IN [ signs/NNS ] that/IN [
portfolio/NN managers/NNS ] expect/VBP [ further/JJ declines/NNS ]
in/IN [ interest_6/NN rates/NNS ] ./.
$$
[ longer/JJR maturities/NNS ] are/VBP thought/VBN to/TO indicate/VB
[ declining/VBG interest_6/NN rates/NNS ] because/IN [ they/PP ]
permit/VBP [ portfolio/NN managers/NNS ] to/TO retain/VB
relatively/RB [ higher/JJR rates/NNS ] for/IN [ a/DT longer/JJR
period/NN ] ./.
```

Dans cet exemple, les crochets [ ] enferment un groupe nominal, chaque mot est suivi de sa catégorie grammaticale (e.g. /NNS), et le mot ambigu, *interest*, est annoté de son sens (_6). Les ponctuations ont pour catégorie elles-mêmes (comme dans ./.). Les phrases sont séparées par une ligne de $$.

Ce corpus contient 2369 instances de mot *interest*. Une description de ce corpus peut être trouvée ici : http://www.d.umn.edu/~tpederse/Data/README.int.txt. Le corpus est pris du site http://www.d.umn.edu/~tpederse/data.html. Le format montré ci-dessus est le format de « original.interest ». Il y a quelques autres formats pour le même corpus (S1 – SemEval-1) et S2 (SemEval-2), ainsi que le format avec la structure syntaxique (parse). Vous pouvez utiliser un format qui vous convient. Notamment, si vous voulez utiliser les informations sur la structure syntaxique, le format avec cette structure peut être utilisé.

3.2. Le processus de désambiguïsation

Pour déterminer le sens du mot, on utilise les informations sur son contexte. Le concept de contexte à la base peut signifier :

- L'ensemble des mots avant et les mots après (dans un sac de mots, sans ordre). Dans le premier exemple, si on tient compte des 2 mots avant et 2 mots après, ces mots sont {*declines, in*, *rate*, .}.
- Les catégories des mots autour. Pour les même 4 mots, nous allons avoir : NNS, IN, NNS, et . (la ponctuation). Ces catégories sont généralement prises en compte en ordre (*C_{-2}=NNS, C_{-1}=IN, C_1=NNS,* et *C_2=.*) afin de tenir compte de la structure syntaxique.

Ces deux groupes de caractéristiques sont ceux que vous devez utiliser au minimum. Mais il y a certaines variations possibles (que vous pouvez tester) : On peut sélectionner les mots autour qui ne sont pas des mots outils (stopword en anglais) comme *in* ou les ponctuations. Ainsi, les 2 mots avant et les 2 mots après significatifs qu'on peut sélectionner peuvent être {*expect, declines, rates*} (*in* et *further* sont des mots outils – voir la liste de stopwords en anglais). Vous avez aussi la possibilité de tronquer ces mots, donnant en utilisant un algorithme de stemming. Note : le même algorithme de stemming intégré dans Weka (l'algorithme Porter) existe aussi en version *stand-alone*. Vous pouvez le trouver sur le site de snowball : http://snowball.tartarus.org/.

Dans la littérature, d'autres types de caractéristiques ont été proposés et utilisés. On vous conseil de consulter la page suivante pour une présentation sommaire :
https://en.wikipedia.org/wiki/Word-sense_disambiguation
La présentation faite dans le cours vous donne aussi quelques autres types de caractéristique utiles. Vous êtes encouragés à les explorer. L'utilisation des caractéristiques supplémentaires sera prise en compte. Si ces caractéristiques supplémentaires sont d'un nombre assez important, des points de boni peuvent être accordés dans la correction.

3.3. Les tâches à réaliser

1. Vous devez faire un programme qui extrait les caractéristiques à partir des textes annotés, et stocker ces caractéristiques en format .arff (utilisé par Weka). Ce formatage a été présenté dans la séance de démonstration. Un squelette de programme d'extraction de caractéristique (par Philip Resnik) est donné. Ce squelette est écrit en Perl. Ce programme vise à transformer le format original. Vous pouvez l'adapter pour faire la transformation des autres formats (si vous les utiliser).
2. Vous devez tester la performance de différents algorithmes de classification. Pour ce TP, on vous demande de tester les algorithmes suivants : Naive Bayes, arbre de décision (J48), SVM (SMO – une implantation de SVM) et MultiLayerPerceptron (en essayant différents nombre de neurones cachés, par exemple 5, 10, 30). Utilisez d'abord les 2 types de caractéristique décrits en haut. Utilisez différentes taille de fenêtre de contexte (1, 2, 3, …, même phrase), et testez comment la performance de désambiguïsation varie selon cette taille. Utilisez la validation croisée 10-fold pour tester la performance.
3. Prenez votre initiative pour tester d'autres caractéristiques.

4. À rendre

Vous devez rendre le programme utilisé pour l'extraction des caractéristiques (votre programme adapté de celui de Resnik, ou un autre programme). Si vous utilisez d'autres programmes pour extraire des caractéristiques supplémentaires, vous devez aussi rendre ces programmes. Faites une petite description de ces programmes dans votre rapport.

En plus des programmes, vous devez aussi rendre un rapport, de longueur entre 5-10 pages. Dans votre rapport, vous devez décrire brièvement les expériences que vous faites, les résultats obtenus, et surtout des comparaisons et des analyses sur les résultats. Vos analyses doivent porter sur la performance des différents algorithmes, l'impact de différentes options - stemming, le nombre de neurones cachés, la taille de fenêtre pour la désambiguïsation, etc. Décrivez librement ce que vous observez d'intéressant dans ces expériences.

Barèmes d'évaluation

- Programme d'extraction de caractéristiques : 3 points
- Test avec Naive Bayes : 2 points
- Test avec arbre de décision : 2 points
- Test avec SMO : 2 points
- Test avec MultiLayerPerceptron : 2 points
- Rapport : 4 points (y compris la description, les analyses et les comparaisons entre différentes options)