SAE S1.01
Projet "Classification automatique" - Partie 2

1. Génération automatique de lexiques

Dans notre système de classification de dépêches (partie 1 du projet), nous avons utilisé des lexiques (un par catégorie) que nous avons construits manuellement. L'objet de la seconde partie du projet est de construire ces lexiques automatiquement. Contrairement à la première partie du projet, vous aurez une certaine liberté dans la définition de la méthode. Le but du jeu étant de trouver la méthode donnant les meilleurs résultats. Vous allez développer une méthode **d'apprentissage automatique** (https://fr.wikipedia.org/wiki/Apprentissage_automatique) sous-domaine de l'**intelligence artificielle**, donnant à votre programme la capacité d'apprendre à partir de données.

2. Description générale de la méthode de génération automatique

La méthode que nous proposons pour construire automatiquement les lexiques consiste à analyser automatiquement le fichier ***depeches.txt*** (ne surtout pas utiliser ***test.txt*** gardé pour les tests) en vue d'extraire pour chaque catégorie, les mots les plus représentatifs. La méthode consiste plus précisément à calculer un score pour tous les mots présents dans au moins une dépêche de la catégorie. Ce score est fonction de la fréquence du mot dans la catégorie (c'est-à-dire le nombre de fois où le mot apparaît dans des dépêches de cette catégorie) et de sa spécificité (qui dépend du nombre de fois où le mot apparaît dans une dépêche d'une autre catégorie).

Ce score peut par exemple, être simplement : *le nombre de fois où le mot apparaît dans des dépêches de la catégorie - le nombre de fois où le mot apparaît dans des dépêches d'autres catégories* (ce qui peut donc donner un score négatif). Mais vous pouvez imaginer autre chose. Sur la base du score de chacun des mots, pour une catégorie donnée, on décidera d'ajouter ou non le mot au lexique de cette catégorie et d'attribuer le cas échéant un poids de 1, 2 ou 3 au mot. Là encore, à vous de décider la façon d'attribuer ces poids.

3. Développements

3.1 Initialisation d'un vecteur de mot/score

Dans **Classification.java**, *développez la méthode* :

```
public static ArrayList<PaireChaineEntier> initDico(ArrayList<Depeche>
depeches, String categorie)
```

qui retourne une `ArrayList<PaireChaineEntier>` *contenant tous les mots présents dans au moins une dépêche de la catégorie* `categorie`. *Attention, même si le mot est présent plusieurs fois, il ne doit apparaître qu'une fois dans la* `ArrayList` *retournée. Dans les entiers, nous stockerons les scores associés à chaque mot et dans un premier temps, nous initialiserons ce score à* `0`.

3.2 Calcul des scores des mots

Dans **Classification.java**, *développez la méthode* :

```
public static void calculScores(ArrayList<Depeche> depeches, String
categorie, ArrayList<PaireChaineEntier> dictionnaire)
```

qui met à jour les scores des mots présents dans `dictionnaire`. *Lorsqu'un mot présent dans* `dictionnaire` *apparaît dans une dépêche de* `depeches`, *son score est : décrémenté si la dépêche n'est pas dans la catégorie* `categorie` *et incrémenté si la dépêche est dans la catégorie* `categorie`.

3.3 Attribution d'un poids en fonction d'un score

Dans **Classification.java**, *développez la méthode* :

```
public static int poidsPourScore(int score)
```

qui retourne une valeur *de poids (1,2 ou 3) en fonction du score* `score` *(à vous de décider quel poids pour quel score).*

3.4 Génération d'un fichier lexique

a) Dans **Classification.java**, *développez la méthode* :

```
public static void generationLexique(ArrayList<Depeche> depeches,
String categorie, String nomFichier)
```

qui crée *pour la catégorie* `categorie` *le fichier lexique de nom* `nomFichier` *à partir du vecteur de dépêches* `depeches`. *Cette méthode doit construire une* `ArrayList<PaireChaineEntier>` *avec* `initDico`, *puis mettre à jour les scores dans ce vecteur avec* `calculScores` *et enfin utiliser le vecteur résultant pour créer un fichier lexique en utilisant la fonction* `poidsPourScore`. *Prenez exemple sur la classe* `ExempleEcritureFichier` *pour l'écriture dans un fichier.*

b) Dans la procédure `main` *de* **Classification.java**, *appelez 5 fois la procédure* `generationLexique` *pour générer les lexiques de chaque catégorie.*

3.5 Utilisation des lexiques générés automatiquement

Les fichiers lexiques générés automatiquement peuvent maintenant être utilisés par le programme de classification écrit dans la partie 1 du projet. Testez-les sur le fichier ***test.txt*** et comparez les résultats obtenus avec ceux initialement obtenus avec les lexiques que vous aviez remplis manuellement. Expérimentez différents calculs de score (`calculScores`) et de poids (`poidsPourScore`) en vue d'obtenir les meilleurs résultats possibles. Intégrez d'autres données d'apprentissage (par exemple le contenu de flux rss : https://www.lemonde.fr/rss/une.xml , https://www.francetvinfo.fr/rss/,...) .

3.6 Amélioration des temps de traitement

Tous les vecteurs utilisés dans ce projet ne sont pas triés. Pourtant, le tri des vecteurs de lexiques et la réécriture des procédures d'accès en tenant compte accélèrerait les traitements (recherche dichotomique par exemple). Complétez vos programmes dans cet objectif et évaluez le gain (facteur d'accélération). Pour calculer les durées d'exécution, prenez exemple sur *ExempleTime.java*.

3.7 Comparaison avec KNN (https://fr.wikipedia.org/wiki/M%C3%A9thode_des_k_plus_proches_voisins)

Vous avez développé votre propre méthode de classification automatique. Vous allez maintenant comparer vos résultats à ceux d'une méthode connue de classification automatique : la méthode des K plus proches voisins. Cette méthode attribue à une dépêche la classe majoritaire parmi les K (K=1 ou 2 ou 3 ou…à déterminer expérimentalement) dépêches les plus ressemblantes dans l'ensemble d'apprentissage. Bien que l'algorithme original s'appuie sur un calcul de distance, pour simplifier, on pourra ici calculer directement les plus proches voisins en se basant sur le nombre de termes communs (en dehors des mots vide comme « le », « la », « les », « de », …). On considèrera donc que les dépêches les plus proches sont celles partageant le plus grand nombre de termes.

3.8 Prise en compte automatique d'autres jeux de données

Votre programme fonctionne pour le jeu de données composé des nouvelles des 5 catégories : `CULTURE`, `ECONOMIE`, `SCIENCES`, `POLITIQUE`, `SPORT`. Créez une classe `Classification2` correspondant à une modification de la classe `Classification` de façon à ce que votre programme découvre les catégories automatiquement à partir du vecteur `depeches` sans les avoir explicitement codées « en dur ». Testez-le avec l'autre jeu de données (***depeches2.txt***, ***test2.txt***) correspondant à une actualité culturelle et contenant d'autres catégories. Testez aussi en supprimant une ou plusieurs catégories des fichiers de données.

.

3.9 Ce qu'il faut rendre

A l'issue de ce projet vous devez rendre :

- un petit rapport (format pdf) de 4 à 5 pages **rédigé en anglais** comportant :
  - une introduction présentant le sujet
  - un point sur ce que vous avez réalisé (ce que vous avez fait, ce que vous n'avez pas fait)
  - une présentation des résultats obtenus et une discussion de ces résultats.
  - Une analyse de la complexité (en nombre de comparaisons) des méthodes `Score` de `Depeche` et `calculScores` de `Classification`
  - une conclusion sur les perspectives d'amélioration de votre système de classification.
- vos programmes
- les lexiques construits manuellement
- les lexiques construits automatiquement
- le fichier réponse pour les lexiques construits manuellement
- le fichier réponse pour les lexiques construits automatiquement

Il suffira de placer tout cela dans votre dossier ***projet-sae-s1-01*** et de taper la commande ***fin-sae-s1-01***

3.10 Présentation Orale

Lors de la dernière séance vous ferez une petite présentation orale de votre projet (5-10 minutes). Prévoyez un programme principal enchaînant la génération automatique des lexiques et la classification. Ce programme devra afficher les résultats de la classification et les temps d'exécutions de la génération des lexiques et de la classification. Votre enseignant testera ensuite par quelques questions votre maîtrise du projet. Vous rendrez votre projet à la fin de votre présentation orale.