

SAE S1.01

Projet "Classification automatique" - Partie 1

Lisez d'abord attentivement les sections 1, 2 et 3 pour comprendre l'objectif du projet. Vous serez guidé pour atteindre cet objectif dans les sections suivantes.

1. Tâche

L'objectif du projet est de créer un programme de classification automatique de dépêches aussi fiable et aussi rapide que possible. Une dépêche est un court texte correspondant à une information journalistique, telle que :

Coupe de l'UEFA : l'OM victorieux, les Girondins défaits
L'OM a remporté son match à domicile sur le score de 3 à 0. Bordeaux a perdu dans les arrêts de jeu.

Le programme devra attribuer à chaque dépêche, l'une des 5 catégories suivantes :

- ENVIRONNEMENT-SCIENCES
- CULTURE
- ECONOMIE
- POLITIQUE
- SPORTS

2. Principes de fonctionnement du programme

2.1 Un lexique par catégorie

Pour réaliser cette classification, le programme va s'appuyer sur des mots marquant l'appartenance à une catégorie particulière. Par exemple, les mots *match* et *joueur* semblent indiquer l'appartenance à la catégorie *SPORTS*. Ainsi, pour chaque catégorie, un lexique correspondant à un ensemble de mots "marqueurs" est créé. De plus, à chaque mot est associé un poids correspondant au degré avec lequel le mot indique l'appartenance à la catégorie. Trois poids sont possibles : 1, 2 ou 3. Un poids maximal de valeur 3 indique que le mot marque très fortement l'appartenance à la catégorie.

Par exemple, le lexique de la catégorie *SPORTS* pourrait être composé des mots et poids associés suivants : *sport* :3, *coupe* :3, *match* :3, *joueur* :2, *domicile* :1. Dans la première partie du projet nous allons construire les lexiques manuellement. Pour cela, vous pourrez vous appuyer sur une centaine de dépêches données en exemple pour chaque catégorie. Il s'agira de choisir parmi les mots présents dans les dépêches d'une catégorie ceux qui vous paraissent les plus représentatifs de la catégorie, c'est-à-dire des mots spécifiques à cette catégorie (que l'on ne retrouve donc pas trop dans les autres catégories). Dans la seconde partie, nous construirons automatiquement les lexiques.

2.2 Calcul d'un score par catégorie

Pour chaque dépêche, un score est calculé pour chaque catégorie (ce qui fait donc 5 scores). Le score de la dépêche pour une catégorie particulière est calculé en faisant la somme du poids des mots présents dans le lexique de la catégorie. La catégorie attribuée à la dépêche est celle obtenant le score maximal (en cas d'égalité, on en choisit arbitrairement une parmi les meilleures). Par exemple, dans le cas de la dépêche donnée en exemple ci-dessus (*Coupe de l'UEFA...*), si on considère le lexique de la catégorie *SPORTS* donné ci-dessus (*sport* :3, *coupe* :3, ...), le score de la dépêche pour la catégorie *SPORTS* serait de $3(\text{coupe}) + 3(\text{match}) + 1(\text{domicile}) = 7$.

3. Les fichiers et leurs formats

3.1 Les fichiers dépêches

Les dépêches sont regroupées dans 2 fichiers *depeches.txt* et *test.txt*. Le fichier *test.txt* ne doit pas être utilisé pour construire les lexiques, il servira aux tests de votre programme de classification. Dans chacun de ces fichiers, on trouve exactement 100 dépêches par catégories. Chaque dépêche correspond à une suite de lignes dans un fichier texte qui se présentent comme suit :

```
.N 002
.D 140208
.C ENVIRONNEMENT-SCIENCES
.T Un test de prédisposition au cancer de la prostate suscite une
polémique. Délivré par Internet, uniquement à des médecins
selon l'entreprise, ce test est mis en vente au prix de 500 dollars (341
euros).
```

- La première ligne correspond au numéro de la dépêche (qui permet de l'identifier)
- La seconde correspond à la date de la dépêche
- La troisième correspond à la catégorie de la dépêche
- Les lignes suivantes correspondent au texte de la dépêche

Les dépêches sont séparées par une ligne vide.

3.2 Les fichiers lexiques

Cinq fichiers correspondant aux différents lexiques devront être créés à l'aide d'un éditeur de texte. Chaque ligne de ce fichier correspondra à un mot suivi de ":" et du poids associé au mot. Le contenu du lexique "SPORTS" pourrait par exemple avoir la forme suivante :

```
sport:3
coupe:3
match:3
jeu:2
domicile:1
```

3.3 Le fichier réponses

Un fichier donnant le résultat de la classification devra être généré automatiquement. Ce fichier aura impérativement la forme suivante :

```
001:ENVIRONNEMENT-SCIENCE
002:ENVIRONNEMENT-SCIENCE
003:ECONOMIE
...
500:SPORTS
ENVIRONNEMENT-SCIENCES:      89%
CULTURE:                     93%
ECONOMIE:                    68%
POLITIQUE:                   80%
SPORTS:                      76%
MOYENNE :                    81.2%
```

Les 500 premières lignes correspondent au numéro de la dépêche suivie de la catégorie attribuée par le programme. Les 5 lignes suivantes correspondent au pourcentage de réponses correctes (il suffit de comparer la réponse du programme à la catégorie réelle à laquelle la dépêche appartient). La dernière ligne correspond à la moyenne des pourcentages de réponses correctes toutes catégories confondues.

4. Organisation du travail

Le projet sera réalisé en binôme et donnera lieu à la rédaction d'un rapport et d'une petite présentation orale lors de la dernière séance. En plus de votre rapport, il vous faudra rendre les fichiers correspondant à vos programmes ainsi que les fichiers correspondants aux lexiques et aux résultats (voir le détail dans l'énoncé de la partie 2). Commencez par lancer le script *debut-sae-s1-01* permettant la récupération du répertoire *projet-sae-s1-01* contenant le projet IntelliJ, nommé *ClassificationAutomatique*. Ouvrez-le. La procédure main est dans la classe *Classification*. En l'état, elle charge et affiche les dépêches. Testez-la.

Parcourez les fichiers sources récupérés pour découvrir les différentes classes, structures de données et les différentes méthodes mises à votre disposition. Les données permettant d'initialiser le vecteur des dépêches sont dans le fichier texte *depeches.txt* que vous pouvez ouvrir avec n'importe quel éditeur de texte. Le fichier *test.txt* contient aussi des dépêches. Il sera exclusivement utilisé pour les tests à la fin des développements (5.6).

5. Développements

5.1 Création des fichiers lexiques

Commencez par créer manuellement un fichier lexique par catégorie. On pourra, dans un premier temps, se limiter à une vingtaine de mots par lexique pour tester le programme et produire de premiers résultats. On pourra ensuite enrichir ces lexiques tout au long du projet. On construira ce lexique au regard des mots présents dans les dépêches de *depeches.txt* correspondant à la catégorie, mais on pourra aussi le compléter de mots a priori pertinents (pensez aux pluriels et en général aux différentes formes d'un mot). Le fichier *test.txt* qui contient aussi des dépêches ne doit pas être utilisé pour construire les lexiques, il sert uniquement aux tests de votre programme et ne doit être utilisé qu'à la fin (5.6).

5.2 Chargement des lexiques en mémoire et accès aux lexiques

- a) Dans **PaireChaineEntier.java**, ajoutez

les attributs suivants :

```
private String chaine;  
private int entier;
```

et écrivez les accesseurs et constructeurs associés.

- b) Dans **Categorie.java**, développez la méthode :

```
public void initLexique(String nomFichier)
```

qui initialise l'attribut `lexique` (une `ArrayList<PaireChaineEntier>`) à partir du fichier lexique dont le nom est passé en paramètre.

Indications : Pour la lecture du fichier vous pourrez prendre exemple sur la méthode `lectureDepeches` de la classe `classification.java`. Pensez aussi à utiliser les méthodes `substring` et `indexOf` de la classe `String` et `parseInt` de la classe `Integer` pour vous faciliter la recopie des informations du fichier vers la `ArrayList<PaireChaineEntier>` représentant le lexique en mémoire.

- c) Dans **Classification.java**, testez la méthode `initLexique` en créant préalablement une catégorie et en affichant le lexique chargé.
- d) Dans **UtilitairePaireChaineEntier.java**, développez la méthode :

```
public static int entierPourChaine(ArrayList<PaireChaineEntier>  
listePaires, String chaine)
```

qui retourne l'entier associé à la chaîne de caractères `chaine` dans `listePaires` si elle est présente et 0 sinon.

- e) Dans **Classification.java**, testez la méthode `entierPourChaine` en demandant la saisie d'un mot à l'utilisateur et en affichant le poids associé dans le lexique chargé.

5.3 Calcul du score d'une dépêche pour une catégorie

- a) Dans **Categorie.java**, développez la méthode :

```
public int score(Depeche d)
```

qui retourne l'entier correspondant au score de la dépêche `d` pour la catégorie sur laquelle l'appel est réalisé. Pensez à utiliser l'attribut `mots` de la classe `Depeche` correspondant à un tableau contenant les mots de la dépêche.

- b) Dans **Classification.java**, testez la méthode `score` en affichant les scores de certaines dépêches pour une catégorie donnée.

5.4 Recherche de la catégorie de score maximal

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

```
public static String chaineMax(ArrayList<PaireChaineEntier>
listePaires)
```

qui retourne la chaîne associée au plus grand entier de listePaires

b) Dans **Classification.java**,

- Créez les 5 catégories et rangez les dans une ArrayList de Catégorie
- Initialiser les lexiques des 5 catégories à partir des fichiers respectifs
- Pour une dépêche donnée, remplissez un vecteur de scores correspondant à une ArrayList<PaireChaineEntier> dans laquelle la chaîne de caractères est le nom de la catégorie et l'entier est le score de la catégorie pour la dépêche.
- En utilisant la méthode chaineMax de UtilitairePaireChaineEntier, affichez le nom de la catégorie ayant le score maximal.

5.5 Création d'un fichier de réponses

a) Afin de comptabiliser le nombre de bonnes réponses pour chaque catégorie, nous allons une nouvelle fois utiliser une ArrayList<PaireChaineEntier> dans laquelle la chaîne de caractères est le nom de la catégorie et l'entier est le nombre de bonnes réponses. Dans **UtilitairePaireChaineEntier.java**, développez la méthode :

```
public static int indicePourChaine(ArrayList<PaireChaineEntier>
listePaires, String chaine)
```

qui retourne l'indice de chaîne dans listePaires si chaîne est présente et -1 sinon.

et la méthode :

```
public static float moyenne(ArrayList<PaireChaineEntier> listePaires)
```

qui retourne la moyenne des entiers stockés dans listePaires.

b) Dans **Categorie.java**, développez la méthode :

```
public static void classementDepeches(ArrayList<Depeche>
depeches, ArrayList<Categorie> categories, String nomFichier)
```

qui pour chacune des dépêches de depeches, calcule le score pour chaque catégorie de categories et écrit dans le fichier de nom nomFichier, le nom de la catégorie ayant le plus grand score ainsi que les pourcentages conformément au format attendu pour les fichiers réponses (voir ci-dessus). Prenez exemple sur la classe ExempleEcritureFichier pour l'écriture dans un fichier.

5.6 Tests et amélioration des résultats

Complétez et améliorez vos lexiques sur la base du contenu du fichier **depeches.txt**. Notez les résultats obtenus en ayant remplacé **depeches.txt** par **test.txt**. Ne trichez pas, le fichier **test.txt** ne doit pas être utilisé pour construire les lexiques. Pour rendre la construction des lexiques automatique, passez à la partie 2 du projet.