IUT2 UNIVERSITÉ Grenoble Alpes
Département INFO

Report - SAE 1.01

Antonin Guilbaud – Amaury Gau
G-B1 – Groupe 2

Table of contents

1 Introduction to the subject

This programming project allows the creation of an automatic news dispatch classification program that is reliable and fast. These news dispatches, which are short journalistic information texts, are divided into five distinct categories: *ENVIRONNEMENT-SCIENCES*, *CULTURE*, *ECONOMIE*, *POLITIQUE* et *SPORTS*. The main objective is simple: to assign each dispatch to the most appropriate category.

The program's operation is based on two key points. Firstly, specific lexicons for each category are manually constructed. These lexicons contain "reference" words associated with importance weights for classification ranging from 1 to 3. Then, a score is calculated for each category by adding the weights of the words present in the corresponding lexicon. The category assigned to a dispatch is the one with the highest score.

The source data is presented in text files, grouping the dispatches in the depeche.txt file. Each dispatch is identified by a number, a date, a category, and a short text.

Initially, the lexicons are manually constructed by taking words from "example" dispatches for each category. The second part focuses on automatically building these lexicons.

As output, the program will generate an answers file with the category assigned to each dispatch, accompanied by the percentage of correct answers by category and an overall average.

2 Update of what we done / haven’t done.

After reading the topic, we declared the attributes "Chaine" and "Entier" in PaireChaineEntier.java, along with the associated accessors and constructors.
Next, we implemented the "initLexique" algorithm. In this program, we initialize the lexicon by reading a specified text file (nomFichier). For each line of the file, it extracts a word and an associated weight by searching for the last character " : " in the line. These pairs (word and weight) are then added to a list of PaireChaineEntier objects, which constitutes the lexicon.
After implementing the "entierPourChaine" method, we developed the "score(Depeche d)" method. In this program, we calculate the score of a news dispatch (d) for a given category. We iterate through each word in the dispatch, using the "entierPourChaine" function to obtain the weight associated with each word in the lexicon. These weights are then summed to form the final score of the dispatch in the specified category.

Next, we coded the "chaineMax" method. In this function, we identify the string associated with the largest integer in a list of string-integer pairs. We iterate through the list to compare the integers, updating the maximum string-integer pair ("chaineMax"). If the largest integer is equal to zero, we return the string "TEST"; otherwise, we return the string associated with the maximum.
Subsequently, we created the 5 categories and stored them in an ArrayList of Category. We then initialized the lexicons of the 5 categories from their respective files and performed the necessary tests.

After coding "indicePourChaine" and "moyenne," we did "classementDepeches." This method was the most time-consuming for us and can be cut into into three parts. Firstly, it iterates through all the news dispatches, calculates scores for each category, and selects the category with the highest score for each dispatch. The corresponding category predictions are then written to a file. At the same time, these predictions are added to a list (resultat) for future use in calculating the success rate. In the second part of the code, we go through all the categories. For each category, we count the number of correct answers by comparing the predicted category with the actual category of the corresponding dispatches. The success rate for each category is then calculated and written to the output file. This code block also contributes to the calculation of the average success rate. In the last part of the code, we calculate the average success rate by iterating through the results obtained for each dispatch. It compares the predicted category with the actual category, and if they are equal, increments the number of correct answers. Then, it writes the average success rate to the output file based on the total number of processed dispatches. Finally, it closes the file.

In part 2, we coded "initDico," which returns an ArrayList of PaireChaineEntier containing all the words present in at least one news dispatch of the specified category. We also coded the calculation of word scores and the assignment of a weight based on a score. Then we developed the "generationLexique" method.

We experimented with the results using automatically generated lexicon files, aiming to achieve the best possible results by changing values in "poidsPourScore."

However, due to a lack of knowledge and time, we were unable to integrate other learning data. Our program works correctly, and this does not impact the rest.

For part 3.6, we attempted to perform a merge sort on vectors and tried to improve search algorithms, attempting to switch to dichotomous. All our codes from now on are traces of research. Specifically, "initDico" must call the "triFusion" method, which needs to be coded. In "calculScores," we call "triFusion" to sort the dictionary. We separated dichotomous sorting, merge sorting, and created a "fusion" method. We also added the lines "long startTime = System.currentTimeMillis();" and "long endTime = System.currentTimeMillis();" at certain places, which will allow us to make comparisons on execution time later.

Unfortunately, we did not have time to explore the comparison with KNN.

3 Result presentation and analyze.

- Initially, with the lexicons created at the start of the project, we get low percentages (with depeches.txt):
ENVIRONNEMENT-SCIENCES : 36%
CULTURE : 41%
ECONOMIE : 32%
POLITIQUE : 53%
SPORTS : 59%
MOYENNE : 44%

- With the improvement of these same lexicons, we get better results, but the percentage remains low (with depeches.txt):
ENVIRONNEMENT-SCIENCES : 50%
CULTURE : 54%
ECONOMIE : 42%
POLITIQUE : 54%
SPORTS : 66%
MOYENNE : 53%

- With test.txt, we get better results, but the percentage stays low:
ENVIRONNEMENT-SCIENCES: 70%
CULTURE: 76%
ECONOMIE: 71%
POLITIQUE: 67%
SPORTS: 87%
MOYENNE: 74%

- With the addition of automatically created lexicons, the percentages are much more conclusive (with depeches.txt):
ENVIRONNEMENT-SCIENCES: 98%
CULTURE: 99%
ECONOMIE: 95%
POLITIQUE: 98%
SPORTS: 100%
MOYENNE: 98%

As far as time is concerned, there is a gain of around 500 ms in classification generation after using fusion sorting and dichotomic search.

▪ *Time classification of generation time **before** trifusion and dichotomic search :*

```
Fichier créé avec succès : Lexique_ENVIRONNEMENT-SCIENCES_Auto.txt, La Génération à durée : 1401ms
Fichier créé avec succès : Lexique_CULTURE_Auto.txt, La Génération à durée : 330ms
Fichier créé avec succès : Lexique_ECONOMIE_Auto.txt, La Génération à durée : 304ms
Fichier créé avec succès : Lexique_POLITIQUE_Auto.txt, La Génération à durée : 360ms
Fichier créé avec succès : Lexique_SPORTS_Auto.txt, La Génération à durée : 416ms
La Génération du classement à durée : 1008ms , Résultats enregistrés dans le fichier resultats_classement.txt
```

▪ *Time classification of generation time **after** trifusion and dichotomic search ::*

```
Fichier créé avec succès : Lexique_ENVIRONNEMENT-SCIENCES_Auto.txt, La Génération à durée : 314ms
Fichier créé avec succès : Lexique_CULTURE_Auto.txt, La Génération à durée : 149ms
Fichier créé avec succès : Lexique_ECONOMIE_Auto.txt, La Génération à durée : 160ms
Fichier créé avec succès : Lexique_POLITIQUE_Auto.txt, La Génération à durée : 168ms
Fichier créé avec succès : Lexique_SPORTS_Auto.txt, La Génération à durée : 152ms
La Génération du classement à durée : 348ms , Résultats enregistrés dans le fichier resultats_classement.txt
```

We can conclude that the addition of automatic lexicon generation achieves a success rate of around 100%. Adding merge sorting and dicothomic search saves time, but this gain could be negligible. We are critical of our results because we expected to save more time. We probably made some mistakes.

4 Analysis of the complexity (in number of comparisons) of the Score in Depeche and calculScores in Classification.

The score method in the "depeche" class calculates the score of a dispatch for a given category by comparing the words in the dispatch with the lexicon of the category. Suppose there are on average n words in a dispatch and m words in the lexicon of a category.

In the worst case, where each word of the dispatch must be compared with each word of the category lexicon, the total number of comparisons would approximately equal n * m.

The calculScores method in the Classification class uses the score method to calculate the score of each dispatch for each category. Suppose there are a total of p dispatches and q categories.

In the worst case, where each dispatch must be compared with each category, the total number of comparisons would approximately equal p * q * n * m.

5 Conclusion on how to improve our classification system.

We could consider using more advanced algorithms to calculate the score of dispatches for each category. Techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) could be used to better represent the importance of words in a given category.

Additional data pre-processing features could also be added, such as removing stop words, word normalization. Lemmatization could also be used, which allows lexical analysis to group words from the same family.

Another possible improvement would be to explore the use of machine learning models. These models can learn from training data and gradually improve their ability to accurately classify dispatches.

We could also consider expanding our lexicon by including category-specific terms.

There are many avenues for improving our classification system. By exploring more advanced algorithms, improving data pre-processing, using machine learning models, or expanding our lexicon.

Finally, this SAE allowed us to improve our JAVA skills. We discovered new JAVA analysis techniques through this concrete example. A news publisher could benefit from this classification to correctly sort their articles and save time. This project presented us with many challenges that we enjoyed solving.

6 Source

Vérificateur orthographe anglais : https://spellcheckplus.com/fr/

Site utile : https://benoitlemaire.wordpress.com/2017/11/13/programme-de-recherche-de-mot-dans-un-fichier-texte/

https://waytolearnx.com/2020/03/lire-un-fichier-ligne-par-ligne-en-java.html

https://www.google.fr/