SAE S1.01
Projet
"Classification automatique"

Table of Contents

Introduction

In this project, the objective was to create a java program capable of performing news classification. These have to be classified by categories thanks to the lexicon assigned to each category. To choose the category to which the news belongs, a value is assigned per word of the lexicon. The more the word belongs to the category, the more value it has. When defining the category of the news item, the category with the highest sum of word values within the news item is chosen. Then, a comparison is made with the real category of the news item to make a percentage of effectiveness.

Our work

For the realization of this project, we developed several functions that helped us to achieve the expected result. First, we create the object PairStringInt to be able to treat the news better when classifying them and to facilitate the process. Then, we develop the method InitLexicon to be able to choose the string selected for each category and to be able to choose the value of each string. Along with that function, we developed the IntForString function that helps us to know the value of each string in the list of strings. After that, we needed to be able to know the score of each word to associate a category with the news. That’s why we developed the Score function that helps us to calculate the score of the news for each category. Then, we worked on a function that allows us to find the best score for a news article so we can associate it with a category. Finally, we have created some other utility functions and the main function which gathers the previous ones and which allows us to classify the news.
After this, we work on our lexicon file generator.
We have also equipped our functions so that we can count the number of comparisons. Thanks to this, we could start to improve processing times by reducing the number of comparisons using sorted vectors and using dichotomous searches. The execution time was reduced by about 4 times and the number of comparisons was divided by about 52.
Unfortunately, we did not have time to try to add more data using different RSS fluxes, but we have coded the function to transform an RSS file in XML format into news that we can read with our program.

Our results

hand built lexicons

First, we tried very short lexicon files for each category. Here are the files:

CULTURE	ECONOMY	ENVIRONMENT-SCIENCES	POLITICAL	SPORTS
culture:3 concert:3 musicien:2 exposition:2 photographie:1	économie:3 bénéfices:2 gouvernement:2 banque:2 croissance:1	science:3 environnement:3 cellule:2 satellite:2 médicaments:2 réchauffement:1	politique:3 président:2 gouvernement:2 Etat:3 maire:3	sport:3 coupe:3 match:3 jeu:2 domicile:1

And here is the response with the percentages of good answers:

CULTURE	ECONOMY	ENVIRONMENT-SCIENCES	POLITICAL	SPORTS
20%	18%	90%	22%	32%

We can see that the environment-sciences category has 90% of good answers but the others do not have good percentages. This is because there are not enough words in our lexicon files so sometimes, the score of a news entry stays at 0 and in this case, the environment-sciences category is chosen by default.

Then, we tried with a few more words, around 20 for each file:

CULTURE	ECONOMY	ENVIRONMENT-SCIENCES	POLITICAL	SPORTS
culture:3 concert:3 musée:3 musique:3 phonographiques:3 opéra:3 ténor:3	économie:3 bénéfices:3 banque:3 banques:3 salariés:3 salarié:3 marché:3 euros:3	science:3 sciences:3 tumeur:3 cancer:3 prostate:3 médecins:3 medecin:3 OGM:3	politique:3 président:3 présidence:3 ministre:3 maire:3 Sarkozy:3 Royal:3 Bayrou:3	sport:3 coupe:3 match:3 matches:3 athlètes:3 athlète:3 olympique:3 JO:3

théâtre:3 oeuvre:3 art:3 danse:3 chante:3 galeries:3 galerie:3 comédie:3 musicien:2 musiciens:2 auteur:2 chorégraphe:2 exposition:2 peintre:2 album:2 photographie:1	dollars:3 emplois:2 gouvernement:2 gouvernements:2 directeur:2 avocat:2 grève:2 délocalisation:2 développement:2 croissance:1 fabrication:1 toucher:1	transgéniques:3 particules:3 automédication:3 transplantations:3 tabac:2 étude:2 environnement:2 cellule:2 satellite:2 orbite:2 éoliennes:2 CO2:2 médicaments:2 pollution:2 greffes:2 réchauffement:1 missile:1	scrutin:3 Etat:3 gouvernement:3 sondage:3 municipales:3 débat:3 electeurs:3 correctionnelle:3 enquête:2 journaliste:2 Français:2 centriste:2 socialiste:2 centre:1 droite:1 gauche:1 voix:1	terrain:3 tournoi:3 championnat:3 foot:3 football:3 basket-ball:3 rallye:3 entraîneur:3 sélectionneurs:2 équipe:2 jeu:2 dopage:2 score:2 domicile:1 Manchester:1

Here is the response with the percentage of good answers:

CULTURE	ECONOMY	ENVIRONMENT-SCIENCES	POLITICAL	SPORTS
59%	52%	80%	61%	66%

The results are starting to look better. We can see some improvements in general.

automatically built lexicons

To go further, we have built lexicons automatically from the file depeches.txt that was provided to us.
For each category, we create a dictionary containing all the words of the news. Then, we go through the words of all the news and search if they are in our dictionary. If it is and the news where we found the word is in the same category of our dictionary, we increase its score and if it is not in the same category, we decrease its score. Finally, we set a weight between 1 and 3 for each score.
First, we start by increasing the score by 1 when we meet it in the good category, decreasing the score by 1 when we meet it in the bad category, and setting the weight to 1 for the words that have a score inferior to 0, 2 for the words that have a score inferior to 2 and 3 for the rest.
Here are the results when we tested it in the depeches.txt file:

CULTURE	ECONOMY	ENVIRONME NT-SCIENCES	POLITICAL	SPORTS
99%	96%	97%	98%	100%

Very good score but it's normal since here we are testing on the files with which we have built our lexicon.
So we tried on the other file that was given to us, test.txt. Here are the results:

CULTURE	ECONOMY	ENVIRONME NT-SCIENCES	POLITICAL	SPORTS
75%	71%	68%	72%	87%

These results look very good with an average of 74.6 % but we can surely do better by playing with the score and weight values.
So we made some tests and the best results were with the following values:
increasing the score by 1 when we meet it in the good category, decreasing the score by 2 when we meet it in the bad category, and setting the weight to 1 for the words that have a score inferior to 0, 2 for the words that have a score inferior to 2 and 3 for the rest.
Here are the results:

CULTURE	ECONOMY	ENVIRONME NT-SCIENCES	POLITICAL	SPORTS
77%	70%	76%	70%	88%

With an average of 76.2%, these are the best results we could get by generating our lexicon files with the news from the depeche.txt file.

Complexity analysis

Methode Score in Depeche

intForString

lines	Instruction
1	intForString(V[0..n] : vectPairStringInt ; string : String) → int
2	int i <- 0 ;
3	int result <- 0 ;
4	while i < n and (string != V[i].getString()) do
5	i = i + 1 ;
6	endwhile ;
7	if i < n then
8	result = V[i].getInt() ;
9	else
10	result = 0;
11	end if;
12	return result ;
13	end ;

Here, the barometers are lines 4 and 5.
$Cost_4(n) = n+1$
$Cost_5(n) = n$
So the cost of this algorithm is in θ(n). It is in the family of linear algorithms.

score

lines	Instruction
1	score(d : Depeche) → int
2	int i <- 0 ;

3	int score <- 0 ;
4	vectString V[0..n] <- d.getWords
5	start
6	while i < n do
7	score = score + intForString(lexique, V[i]) ;
8	i = i + 1 ;
9	endwhile ;
10	return score ;
11	end ;

Here, the barometers are lines 6 and 7.
$Cost_6(n) = n+1$
$Cost_7(n) = n * Cost_{intForString} = n * n = n^2$
So the cost of this algorithm is in $\theta(n^2)$. It is in the family of quadratic algorithms.

Methode CalculeScore in Classification

indexForString

lines	Instruction
1	function indexForString(V[0..n] : vectPairStringInt ; string : String) → int
2	int i <- 0 ;
3	int result <- -1 ;
4	start
5	while i < n and (string != V[i].getString()) do
6	i = i + 1 ;
7	endwhile ;
8	if i < n then
9	result = i;

10	else
11	result = -1;
12	endif;
13	return result ;
14	end ;

Here, the barometers are lines 5 and 6.
$Cost_5(n) = n+1$
$Cost_6(n) = n$
So the cost of this algorithm is in θ(n). It is in the family of linear algorithms.

calculScores

lines	Instruction
1	function calculScores(V[0..n] : vectdepeches , categorie : String, D[0..m] : vectPairStringInt)
2	int indexDic;
3	vectString W[0..n];
4	start
5	for i ranging from 0 to n do
6	W[0..k] <- V[i].getWords() ;
7	for j ranging from 0 to k do
8	indexDic <- indexForString(D[0..m], W[j]) ;
9	if indexDic != -1 then
10	if V[i].getCategorie() = categorie then
11	D[indexDic].setInt(D[indexDic].getInt() + 1);
12	else
13	D[indexDic].setInt(D[indexDic].getInt() - 2);
14	endif;

```
15              endfor;
16          endfor;
17      end;
```

Here, the barometers are in lines 5, 7 and 8.
$\text{Cost}_5(n) = n+1$
$\text{Cost}_7(n) = n*(n+1)$
$\text{Cost}_8(n) = n^2 * \text{Cost}_{\text{indexForString}} = n^2 * n = n^3$
So the cost of this algorithm is in $\theta(n^3)$. It is in the family of polynomial algorithms.

Conclusion

To recapitulate, we can say that our news classification program works quite well, but there are several points that we can improve to get better and more accurate results. Adding the amount of data to have a wider lexicon in each category would make the classification more efficient. In addition, we could optimize the score function to calculate more efficiently the scores of the words in the news.