Corpus

Données Web

HUET Bryan, SAHIN Tolga

Année universitaire 2020

Table des matières

1 Introduction

Le projet de notre corpus est de réaliser un moteur de recherche via des indexage et à l'aide la méthode de pondération TF-IDF.

1.1 Le TF-IDF

Le tf-idf est le poids d'un mot dans un ensemble de documents. Ce poids augmentera en fonction de la fréquence brute des mots (TF) et de la fréquence des mots dans tous les documents. (DF) Si le tf-idf est grand alors plus le mot est fréquent. Tandis que, moins il l'est le mot est peu fréquent.

1.2 Indexage

Le second objectif ce ce projet est de rechercher la fréquence (TF-IDF) de différents mots dans un paquet de documents. De ce fait-là, nous procéderons à un indexage depuis différentes requêtes permettant de trier. Puis, nous calculerons le TF-IDF des mots.

2 Calcul du TF

2.1 Le TF

Le TF est la fréquence des mots dans un document. (Term Frequency)

2.2 Calcul du TF

On commence par récupérer l'ensemble de nos documents (3655 dans notre cas).

Listing 1 – Bouclage

```
for i in $(seq 3655) ; do cat $i | ...
```

On convertit tous nos String majuscule en miniscule.

Listing 2 – Miniscule

```
tr [:upper:] [:lower:] | ...
```

On supprime tous les mots qui utilise des caractères spéciaux.

Listing 3 – Caractères spéciaux

```
sed 's/[^a-z]/ /g' | ...
```

Si le nombre de fields est différent de 0 alors on print les champs de texte. Puis, on pipe l'output avec la requête sort qui permet de trier. Donc, on obtient une liste de mot ordonnée.

Listing 4 – Liste de mots triés

```
awk 'NF != 0 {print}'| sort | ...
```

La requête awk permet de calculer le TF. On calcule depuis notre liste de mots. Le nombre d'occurrences de chacun de ces mots. On retourne le mot et son TF. La requête sed permet de supprimer la première ligne avec le paramètre '1d'. Puis, de stocker les résultats du script (les TF des mots) dans des fichiers .tf jusqu'à la fin de notre boucle.

Listing 5 – Calcul du TF

```
awk '{if (mot == $1) tf ++;
else {print mot, tf; mot = $1; tf = 1}} END {print mot, tf}'
| sed 1d >$i.tf;done
```

2.3 Temps d'éxécution

Le script TF prend en moyenne à l'aide la requête time ce délai :

Listing 6 – Suppression

```
time bash df.sh

real 1m18,654s
user 1m0,027s
sys 0m28,940s
```

3 Le DF

3.1 Définition

Le DF (Document Frequency) est le nombre d'occurence de tous les mots dans un ensemble de documents.

3.2 Calcul du DF

Le début du script suit exactement le même principe que pour le TF.

Listing 7 – Bouclage

```
for i in $(seq 3655); do cat $i
| tr [:upper:] [:lower:]
| sed 's/[^a-z]/ /g'
| sed 's/ /\n/g' | ...
```

sort -u est aussi équivalent à sort | uniq. Ce paramètre permet de trier plus efficacement notre ensemble de documents. Les mots dupliqués sont éliminés. Donc, notre output devient la liste de mot de l'ensemble des documents. De plus, "done;" est important car il ferme notre boucle pour ne garder en sortie que cette liste de mot. On trie cette liste de mot cette fois-ci via la requête sort. On obtient une liste de mots par ordre alphabétique.

Listing 8 – Tris

```
awk 'NF != 0 {print}'| sort -u; done | sort | ...
```

Enfin, on termine le script de la même manière que pour le TF. On compte le nombre d'occurrence de notre liste de mot. Sauf que cette fois-ci notre output est stocké dans un seul fichier texte "df.txt" car on ne boucle plus. Cette output contient le mot et la fréquence de mot par ligne. Donc, l'output est le DF des mots dans nos documents.

Listing 9 – Occurences

```
awk '{if (mot == $1) tf ++;
else {print mot, tf; mot = $1; tf = 1}} END {print mot, tf}'
| sed 1d > ../df.txt
```

3.3 Temps d'exécution

Listing 10 – Temps d'exécution du script df.sh

```
time bash df.sh

real 1m1.042s
user 0m51,006s
sys 0m28,940s
```

4 Indexage et calcul du TF-IDF

L'indexage et le calcul du TF-IDF va s'effectuer en quatre script.

4.1 Premier script

Listing 11 – Requête 1

```
#!/bin/bash
for i in $(seq 3655); do cat "content/"$i.tf
| sed "s/ .*/ $i/"; done | ...
```

Listing 12 – Requête 2

```
sort -k1,1 -k2,2n | ...
```

Listing 13 – Requête 3

```
awk '{if ($1 != last){if (last!="")print last, tab[last]; last = $1; tab[last] = $2}
else tab[last] = tab[last] " " $2}END{print last, tab[last]}' > index
```

4.2 Second script

Listing 14 – Requete 1

```
for i in $(cat query.txt); do grep "^$i " index | ...
```

Listing 15 – Requête 2

```
sed 's/[^ ]* //; s/ /\n/g' | sort > $i.index; done;
```

Listing 16 – Requête 3

```
cp $(head -1 query.txt).index answer;
```

Listing 17 – Requête 4

```
for i in $(sed 1d query.txt);
do comm -1 -2 answer $i.index > tmp; mv tmp answer; done;
```

4.3 Troisième script

Listing 18 – Requête 1

```
for i in $(cat query.txt); do grep "^$i " df.txt; done | ...
```

Listing 19 – Requête 2

```
awk '{print $1, log(3655/$2)}' | awk '{print $2}' > query.tfidf
```

4.4 Quatrième script

Listing 20 – Requête 1

```
for i in $(sort answer); do echo -n "$i ";
for j in $(cat query); do grep "^$j " $i.tfidf | ...
```

Listing 21 – Requête 2

```
awk '{print $2}'; done > $i.pert; paste -d" " query.tfidf $i.pert | ...
```

Listing 22 – Requête 3

```
awk '{sum+=$1*$2;norm1+=$1*$1;norm2+=$2*$2}END{print sum/sqrt(norm1)/sqrt(norm2)}';
rm -f $i.pert; done
```

5 Temps d'exécution

Script 1 :

Listing 23 – Temps d'exécution du script 1

```
time bash step1.sh

real 0m4,744s
user 0m6,020s
sys  0m1,207s
```

Script 2 :

Listing 24 – Temps d'exécution du script 2

```
time bash step2.sh

real 0m0,265s
user 0m0,039s
sys  0m0,008s
```

Script 3 :

Listing 25 – Temps d'exécution du script 3

```
time bash step3.sh

real 0m0,013s
user 0m0,007s
sys  0m0,012s
```

Script 4 :

Listing 26 – Temps d'exécution du script 4

```
time bash step4.sh

real 0m0,197s
user 0m0,231s
sys  0m0,071s
```

Total :

Listing 27 – Temps d'exécution en tout

```
time bash tfidf.sh

real 0m5,219s
user 0m6,297s
sys  0m1,298s
```

6 Conclusion