Auteur 5.1 : **Dufour Emilien**

Groupe : Drapp Thibault, Badiane Issa, Dufour Emilien

5 janvier 2018

PLP rapport de TP

Exercice 5

Question 5.1 : TF-IDF par Emilien DUFOUR

Objectif :

Nous devons réaliser un programme Map-Reduce calculant le TF-IDF de chaque mot d'un ensemble de documents.

Méthode :

Soit $TFIDF_{i,j}$, le TFIDF du mot m_i du document d_j du corpus C.

$TFIDF_{i,j} = wordcount(m_i,d_j)/card(d_j) * log10(card(C)/card(\{d \in C \mid m_i \in d\}))$

On effectue une série de 3 Map-Reduce sur notre Corpus.

	Key	**Value**
Map 1	[mot i, doc j]	1
Shuffle 1	[mot i, doc j]	{1,1,….,1}
Reduce 1	[mot i, doc j]	$Wordcount_{i,j}$
Map 2	doc j	[mot i, $wordcount_{i,j}$]
Shuffle 2	doc j	{[mot 1, $wordcount_{1,j}$],…,[mot N, $wordcount_{N,j}$]}
Reduce 2	[mot i, doc j]	[$Wordcount_{i,j}$, $lengthdoc_j$]
Map 3	mot i	[doc j, $wordcount_{i,j}$, $lengthdoc_j$]
Shuffle 3	mot i	{[doc 1, $wordcount_{i,1}$, $lengthdoc_1$],…,[doc M, $wordcount_{i,M}$, $lengthdoc_M$]}
Reduce 3	[mot i, doc j]	[$Wordcount_{i,j}$, $lengthdoc_j$, $nbrDocHavingMot_i$]
	—>	[TF-IDF i,j]

Résumé et remarques importantes :

On applique donc en cascade 3 fois l'architecture java pour les jobs MapReduce.

La complexité réside dans la gestion des clefs composites et dans l'implémentation de plusieurs boucles sur les mêmes « values » dans le Reducer.

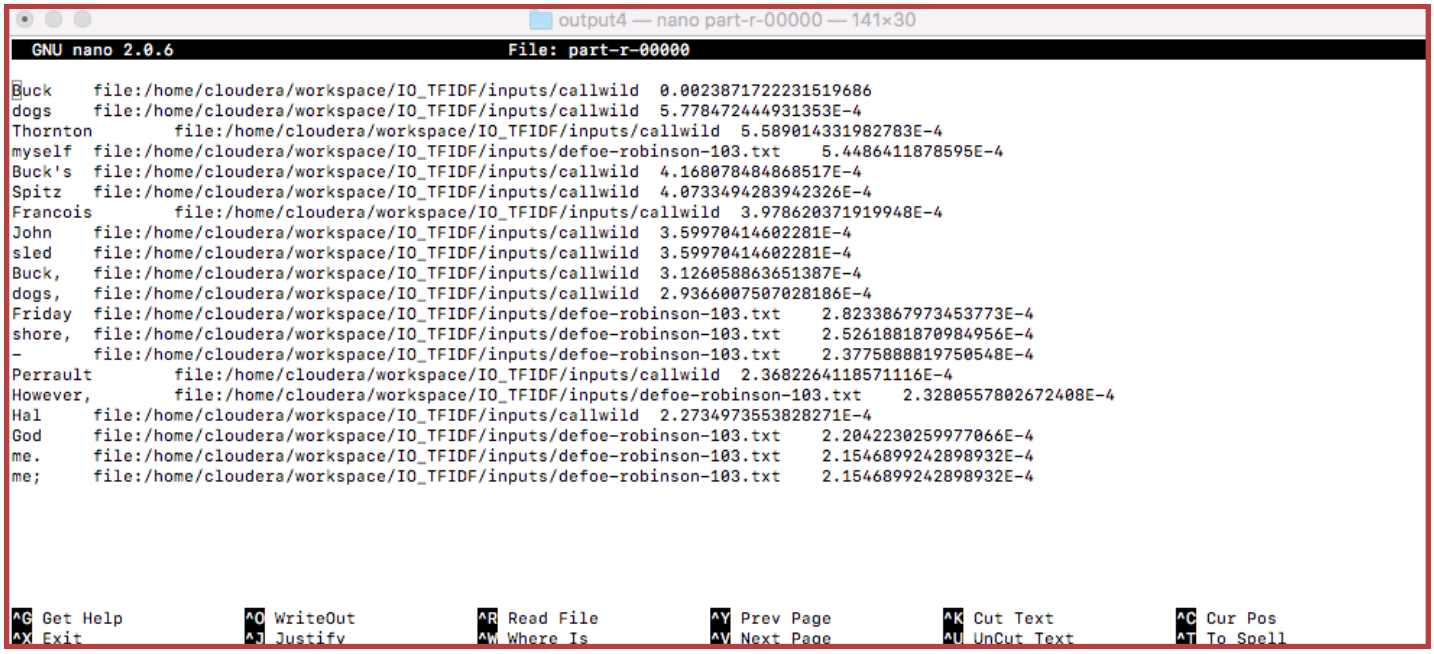
- Pour les clefs composites, on crée donc des classes dont les attributs sont les éléments de nos vecteurs clefs. Ces classes implémentent l'interface « WritableComparale<CompositeKey> ».

- Pour les boucles dans le Reducer : il s'agit de générer à l'issue du Reducer plus de clefs que celles reçues. C'est le cas par exemple du Reducer du round2 : on émet des couples clef : <mot i, doc j> valeur : <wordcount ij , tailleDoc j> alors que le Mapper émettait clef : <doc j> valeur : <mot i, wordcount ij> (voir schéma ci-dessus).

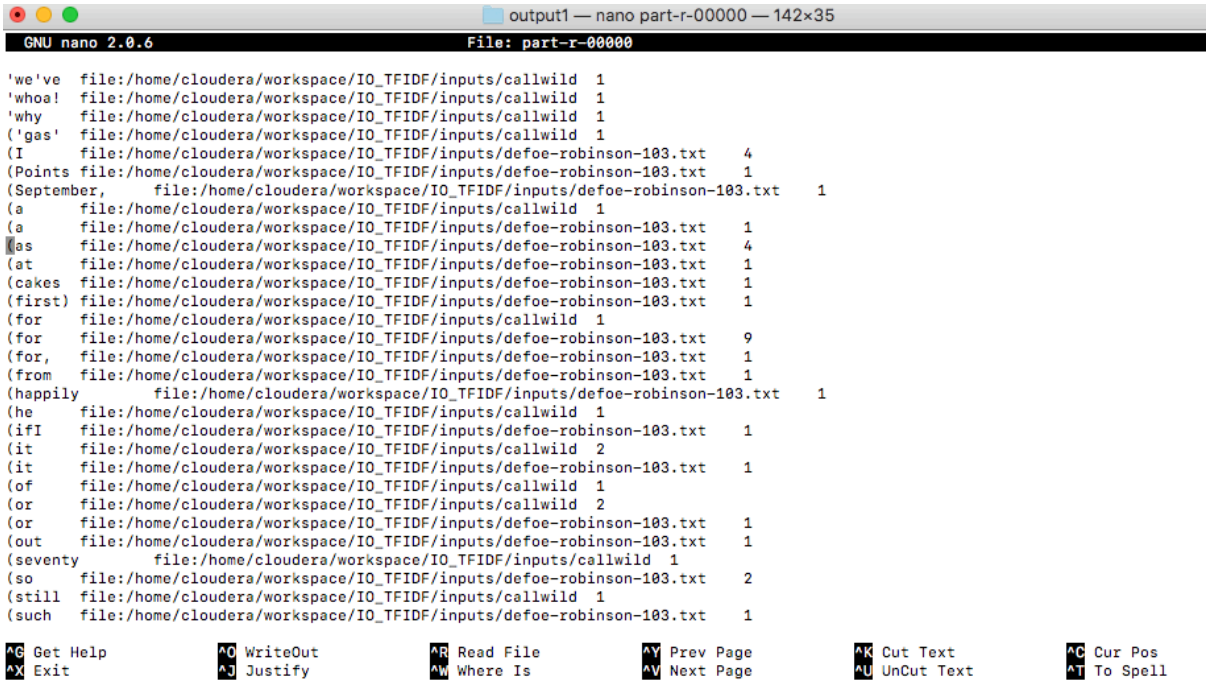
Il faut donc que le Reducer fasse deux boucles sur le même objet « Iterator ». Pour cela on crée un cache qui contiendra les « values » et que l'on remplit dans la première boucle. Il faudra faire attention à remplir le cache en créant des copies des objets contenus dans «Iterator», afin de ne pas faire pointer les objets de notre cache vers le même objet.

Résultats :

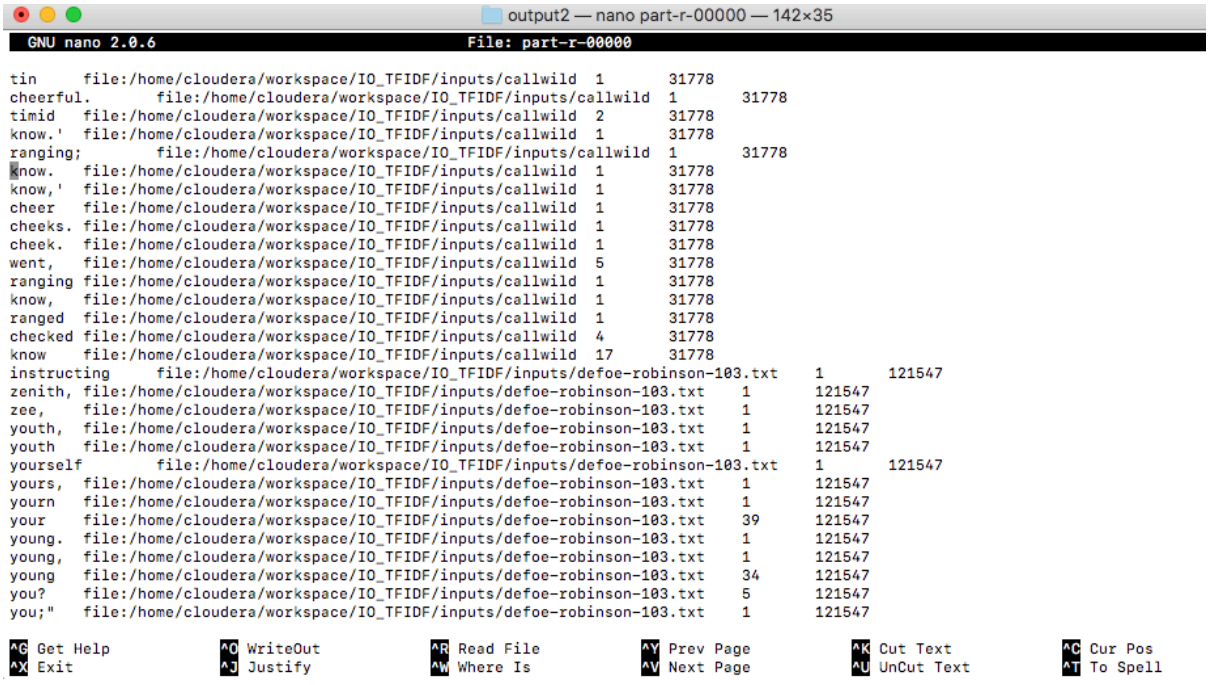
Voici les 20 mots ayant le TF-IDF le plus élevé, classés en ordre décroissant.



On a d'abord calculé le wordcount de toutes les paires (**mot i , doc j**).



Ensuite on calcule le nombre de mots par document et on l'associe aux couples (**mot i , doc j , wordcount ij**).



Ensuite on calcule le nombre de documents contenant un certain mot. En associant ces quantités aux couples (mot i , doc j , wordcount ij , tailleDoc j), on détermine les couples **(mot i , doc j , TFIDF ij)**.

Enfin on sélectionne les 20 couples **(mot i , doc j , TFIDF ij)** ayant les plus forts TFIDF.