

# Pràctiques - Programació declarativa. Aplicacions

Curs 2022/23

GEINF

---

## Pràctica ViquipediaSimil

Angel Molero 41512839R – [u1967593@campus.udg.edu](mailto:u1967593@campus.udg.edu)

Gabriel Lopez 41511387K – [u1968394@campus.udg.edu](mailto:u1968394@campus.udg.edu)

Professor: Mateu Villaret Auselle

---

# Índex

<b>A</b>	<b>Introducció</b>	<b>2</b>
<b>B</b>	<b>Descripció de les classes</b>	<b>2</b>
<b>C</b>	<b>Extractes comentats del codi</b>	<b>2</b>
C.1	Primera Part . . . . .	2
C.2	Segona Part . . . . .	3
C.2.1	Modificació MapReduce . . . . .	4
<b>D</b>	<b>Càlculs realitzats</b>	<b>4</b>
<b>E</b>	<b>Jocs de proves</b>	<b>5</b>
E.1	Resultats Execucions Primera Part . . . . .	5
E.1.1	Freqüència de paraules . . . . .	5
E.1.2	Sense stop-words . . . . .	6
E.1.3	Distribució de paraules . . . . .	6
E.1.4	Ngrames . . . . .	7
E.1.5	Vector space model . . . . .	7
<b>F</b>	<b>Taula de rendiment</b>	<b>8</b>
F.1	Especificacions . . . . .	8
F.2	Taula dels actors . . . . .	8
<b>G</b>	<b>Alltres consideracions</b>	<b>8</b>

## A Introducció

L'objectiu de la pràctica consisteix en analitzar el contingut de diversos fitxers i comprovar la seva similitud. S'ha realitzat de dues formes diferents, la primera de forma iterativa i la segona mitjançant mappers i reducers. Encara que l'objectiu sigui el mateix, el resultat final serà diferent, ja que, a la primera s'utilitzen un número reduït de documents a analitzar, mentre que a la segona s'utilitzen 5000.

Dins el directori **primeraPart** es pot veure el codi realitzar per la primera part i un document del mateix.

## B Descripció de les classes

- **Main.scala**: Dins l'objecte **Main** trobem els mètodes principals dels diferents mappers i reducers, i on s'executa el codi inicial.
- **InfoFicheros.scala**: Dins l'objecte **InfoFicheros** guardem aspectes dels diferents fitxers analitzats i alguns mètodes auxiliars. Així calculem alguns aspectes un únic cop i millorem el rendiment.
- **MapReduceActors.scala**: En aquest fitxer trobem les classes **Mapper** i **Reducer** on s'han modificat per complir amb els objectius de la pràctica.

## C Extractes comentats del codi

### C.1 Primera Part

S'han agafat les funcions d'ordre superior més rellevants.

- **toFloat**: Mètode utilitzat per convertir el valor a float. Un exemple d'ús:

```
private def calculFreq(numParaules: Int, i: Int): BigDecimal = {  
    BigDecimal((i.toFloat / numParaules) * 100)  
}
```

- **foldLeft**: Mètode que agafa una funció d'ordre superior i l'utilitzarà per col·lapsar la col·lecció. Un exemple d'ús, per calcular la freqüència total de les paraules. Ens serveix per recorre i guardar el valor anterior:

```
private def raizSumatorio(freq1: List[(String, Double)]): Double = {  
    = Math.sqrt(freq1.foldLeft(0.0)((x, y) => x + y._2 * y._2).doubleValue)  
}
```

- **map**: Mètode que aplica una funció a cada element de la col·lecció. L'hem utilitzat per calcular la freqüència normal. Exemple d'ús:

```
private def frecuenciaNormal(l1: List[(String, Int)], freqNormMax1: Double, sumaTotal: Double): List[(String, Double)] = {  
    l1.map(x => (x._1, redondear(BigDecimal((x._2.toFloat / sumaTotal) * 100), ndigits = 2) / freqNormMax1))  
}
```

- **setScale**: Mètode utilitzat per redondejar. Exemple d'ús:

```
def redondear(num: BigDecimal, ndigits: Int): Double = {
  num.setScale(ndigits, BigDecimal.RoundingMode.HALF_UP).doubleValue
}
```

- **compareTo**: Mètode comparar dues cadenes. L'utilitzem per saber quina de les dues cadenes hem d'avançar. Exemple d'ús:

```
def vectorAlignment(v1: List[(String, Double)], v2: List[(String, Double)], result: Double): Double = {
  if (v1.isEmpty || v2.isEmpty) result
  else {
    val comp = v1.head._1.compareTo(v2.head._1)
    if (comp == 0) vectorAlignment(v1.tail, v2.tail, result + v1.head._2 * v2.head._2)
    else if (comp < 0) vectorAlignment(v1.tail, v2, result)
    else vectorAlignment(v1, v2.tail, result)
  }
}
```

- **sliding**: Utilitzada per dividir la llista en n grames. Exemple d'ús:

```
def ngrames(n: Int, texto: String): List[(String, Int)] = {
  val modtexto = texto.toLowerCase().replaceAll( regex = "[^a-z]", replacement = " ").split( regex = " ").filter(_ != "")
  val grames = modtexto.sliding(n).toList
  val nrep = grames.map(x => (x.mkString(" "), 0))
  nrep.groupBy(_._1).map(x => (x._1, x._2.size)).toList.sortWith(_._2 > _._2)
}
```

- **groupBy**: Utilitza el predicat rebut per agrupar i formar un Map. L'utilitzem agafant els nGrames (1 2) i formats com una cadena i els agrupem per key i així agrupem per nGrama. Exemple d'ús:

```
def ngrames(n: Int, texto: String): List[(String, Int)] = {
  val modtexto = texto.toLowerCase().replaceAll( regex = "[^a-z]", replacement = " ").split( regex = " ").filter(_ != "")
  val grames = modtexto.sliding(n).toList
  val nrep = grames.map(x => (x.mkString(" "), 0))
  nrep.groupBy(_._1).map(x => (x._1, x._2.size)).toList.sortWith(_._2 > _._2)
}
```

- **distinct**: Aquest mètode l'utilitzem per guardar només les paraules úniques, i així no repetir. Exemple d'ús:

```
def nonstopfreq(texto: String): List[(String, Int)] = {
  val stopwords = scala.io.Source.fromFile("src/main/scala/english-stop.txt").mkString
  val stop = stopwords.split( regex = "\n").toList
  val modtexto = texto.toLowerCase().replaceAll( regex = "[^a-z]", replacement = " ").split( regex = " ").filter(_ != "")
  val f = modtexto.filter(!stop.contains(_))
  f.distinct.map(p => (p, f.count(x => x == p))).sortWith(_._2 > _._2).toList
}
```

## C.2 Segona Part

A la segona part no cal destacar cap funció d'ordre superior, totes les utilitzades són molt comunes.

### C.2.1 Modificació MapReduce

## D Càlculs realitzats

- input:
  - mappingLlegir i reduccingLlegir:
    - \* K1:
    - \* V1:
    - \* K2:
    - \* V2:
    - \* V3:
  - mappingRef i reduccingRef:
    - \* K1:
    - \* V1:
    - \* K2:
    - \* V2:
    - \* V3:
  - mappingCombNoRef i reduccingCombNoRef:
    - \* K1:
    - \* V1:
    - \* K2:
    - \* V2:
    - \* V3:
  - mappingWC i reduccingWC:
    - \* K1:
    - \* V1:
    - \* K2:
    - \* V2:
    - \* V3:
  - mappingTF i reduccingTF:
    - \* K1:
    - \* V1:
    - \* K2:
    - \* V2:
    - \* V3:
  - mappingTfIdf i reduccingTfIdf:
    - \* K1:
    - \* V1:
    - \* K2:
    - \* V2:
    - \* V3:
  - mappingGirar i reduccingGirar:
    - \* K1:
    - \* V1:

- \* K2:
- \* V2:
- \* V3:
- mappingPalDoc i reduccingPalDoc:
  - \* K1:
  - \* V1:
  - \* K2:
  - \* V2:
  - \* V3:
- mappingRaizSumatorio i reduccingRaizSumatorio:
  - \* K1:
  - \* V1:
  - \* K2:
  - \* V2:
  - \* V3:
- mappingCosinoSimil i reduccingCosinoSimil:
  - \* K1:
  - \* V1:
  - \* K2:
  - \* V2:
  - \* V3:

## E Jocs de proves

### E.1 Resultats Execucions Primera Part

#### E.1.1 Freqüència de paraules

```
Num de Paraules: 30419 Diferents: 3007
Paraules ocurrences frecuencia
-----
the 1818    5.98
and  940    3.09
to   809    2.66
a    690    2.27
of   631    2.07
it   610    2.01
she  553    1.82
i    545    1.79
you  481    1.58
said  462    1.52
```

### E.1.2 Sense stop-words

```
Num de Paraules: 10038 Diferents: 2623
Paraules ocurrences frequencia
-----
alice    403 4.01
guttenberg  93 0.93
project  87 0.87
queen    75 0.75
thought  74 0.74
time     71 0.71
king     63 0.63
turtle   59 0.59
began    58 0.58
tm       57 0.57
```

### E.1.3 Distribució de paraules

```
Les 10 frequencies mes frequents:
1330 paraules apareixen 1 vegades
468 paraules apareixen 2 vegades
264 paraules apareixen 3 vegades
176 paraules apareixen 4 vegades
101 paraules apareixen 5 vegades
74 paraules apareixen 8 vegades
72 paraules apareixen 6 vegades
66 paraules apareixen 7 vegades
39 paraules apareixen 9 vegades
35 paraules apareixen 10 vegades

Les 5 frequencies menys frequents:
1 paraules apareixen 68 vegades
1 paraules apareixen 178 vegades
1 paraules apareixen 227 vegades
1 paraules apareixen 940 vegades
1 paraules apareixen 100 vegades
```

### E.1.4 Ngrams

```
(project gutenber tm,57)
(the mock turtle,53)
(i don t,31)
(the march hare,30)
(said the king,29)
(the project gutenber,29)
(said the hatter,21)
(the white rabbit,21)
(said the mock,19)
(said to herself,19)
```

### E.1.5 Vector space model

- Resultat execució (**n = 0**):

	pg11-net.txt	pg11.txt	pg12-net.txt	pg12.txt	pg2500-net.txt	pg2500.txt	pg74-net.txt	pg74.txt
pg11-net.txt	1,000	0,951	0,864	0,831	0,213	0,210	0,219	0,217
pg11.txt	0,951	1,000	0,823	0,876	0,209	0,278	0,214	0,258
pg12-net.txt	0,864	0,823	1,000	0,962	0,202	0,198	0,208	0,207
pg12.txt	0,831	0,876	0,962	1,000	0,202	0,262	0,208	0,247
pg2500-net.txt	0,213	0,209	0,202	0,202	1,000	0,971	0,269	0,268
pg2500.txt	0,210	0,278	0,198	0,262	0,971	1,000	0,265	0,299
pg74-net.txt	0,219	0,214	0,208	0,208	0,269	0,265	1,000	0,988
pg74.txt	0,217	0,258	0,207	0,247	0,268	0,299	0,988	1,000

- Resultat execució (**n = 2**):

	pg11-net.txt	pg11.txt	pg12-net.txt	pg12.txt	pg2500-net.txt	pg2500.txt	pg74-net.txt	pg74.txt
pg11-net.txt	1,000	0,965	0,771	0,748	0,425	0,432	0,582	0,581
pg11.txt	0,965	1,000	0,752	0,792	0,439	0,493	0,585	0,613
pg12-net.txt	0,771	0,752	1,000	0,962	0,443	0,446	0,629	0,626
pg12.txt	0,748	0,792	0,962	1,000	0,453	0,504	0,626	0,651
pg2500-net.txt	0,425	0,439	0,443	0,453	1,000	0,933	0,633	0,636
pg2500.txt	0,432	0,493	0,446	0,504	0,933	1,000	0,644	0,671
pg74-net.txt	0,582	0,585	0,629	0,626	0,633	0,644	1,000	0,990
pg74.txt	0,581	0,613	0,626	0,651	0,636	0,671	0,990	1,000

- Resultat execució (**n = 3**):

	pg11-net.txt	pg11.txt	pg12-net.txt	pg12.txt	pg2500-net.txt	pg2500.txt	pg74-net.txt	pg74.txt
pg11-net.txt	1,000	0,894	0,320	0,284	0,061	0,057	0,209	0,198
pg11.txt	0,894	1,000	0,284	0,434	0,055	0,225	0,185	0,293
pg12-net.txt	0,320	0,284	1,000	0,893	0,071	0,068	0,243	0,230
pg12.txt	0,284	0,434	0,893	1,000	0,065	0,227	0,214	0,313
pg2500-net.txt	0,061	0,055	0,071	0,065	1,000	0,620	0,142	0,134
pg2500.txt	0,057	0,225	0,068	0,227	0,620	1,000	0,139	0,265
pg74-net.txt	0,209	0,185	0,243	0,214	0,142	0,139	1,000	0,943
pg74.txt	0,198	0,293	0,230	0,313	0,134	0,265	0,943	1,000



## F Taula de rendiment

Per la realització de les diferents proves, s'ha utilitzat l'equip amb les següents especificacions.

### F.1 Especificacions

Nombre del dispositivo	DESKTOP-MHA7BS0
Procesador	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
RAM instalada	16,0 GB (15,8 GB usable)

60 segundos

Uso

9%

Velocidad

3,22 GHz

Procesos

275

Subprocesos

3876

Identificadores

148879

Tiempo activo

24:09:39:23

Velocidad de base:

2,59 GHz

Sockets:

1

Núcleos:

6

Procesadores lógicos:

12

Virtualización:

Habilitado

Caché L1:

384 kB

Caché L2:

1,5 MB

Caché L3:

12,0 MB

[Monitor de recursos](#)

### F.2 Taula dels actors

Les proves s'han realitzat amb tots els fitxers disponibles, i la taula representa per cada número d'actors, quants segons triga.

Número d'actors			
1	4	10	20
0.5	0.5	0.5	0.5

## G Altres consideracions

Com s'indiquen a les proves anteriors, s'han realitzat amb tots els fitxers disponibles i s'ha aconseguit un molt bon temps. Per tant, molt contents amb el resultat final.