



Similitud entre documents (Part II)

Programació Declarativa, Aplicacions

8 de novembre del 2024

Wilber Eduardo Bermeo Quito

Dr. Mateu Villaret Auselle

Universitat de Girona

Jordi Badia Auladell

41591544T

jordibadiauladell@gmail.com

Aniol Juanola Vilalta

41559862N

u1978893@campus.udg.edu

Continguts

1. Extractes comentats de codi	3
1.1. Funcions d'ordre superior	3
1.1.1. MRWrapper	3
1.1.2. Timer	3
1.2. Modificació del MapReduce	4
2. Jocs de proves	6
3. MapReduce	10
3.1. Comptar referències per document	11
3.2. Filtrar paraules i referències al content d'un ViquipediaFile	11
3.3. Filtrar documents que contenen una <i>query</i>	12
3.4. Calcular el PR	12
3.5. Trobar els documents mútuament referenciats	13
3.6. Calcular TF (<i>word frequency</i>) de cada document	13
3.7. Calcular IDF de cada paraula	14
3.8. Calcular TFIDF per document	14
3.9. Calcular similituds entre documents	15
4. Classes i objectes auxiliars	16
4.1. ViquipediaFile	16
4.2. MappingReduceFunctions	16
4.3. MRWrapper	16
4.4. MapReduceActors	16
4.5. Timer	16
4.6. ProcessFiles	16
5. Taula de rendiment	17
6. Creació i tancament del sistema d'Actors	18
7. Altres consideracions	18
7.1. Filtratge de referències	18
7.2. Ús de MapReduce	18
7.3. Reutilització de material de la primera entrega	19

1. Extractes comentats de codi

1.1. Funcions d'ordre superior

1.1.1. MRWrapper

Objecte que encapsula i gestiona la creació dels actors en cridar la funció MR, que rep per paràmetres una entrada, una funció de mapeig, una funció de reducció, i els nombres d'actors a processar el mapeig i la reducció. Cal destacar que la `mappingFunction` i la `reducingFunction` es passen amb *lazy* per a què la seva execució sigui duta a terme dins la classe `MapReduce`. Per a més detalls del seu funcionament, veure Secció 6.

```
object MRWrapper {  
  
    private val timeoutValue: FiniteDuration = 10000 seconds  
  
    def MR[K1, V1, K2, V2, V3](input: List[(K1, List[V1])],  
                               mappingFunction: (K1, List[V1]) => List[(K2, V2)],  
                               reducingFunction: (K2, List[V2]) => (K2, V3),  
                               mapperNumber: Int = 1,  
                               reducerNumber: Int = 1) : Map[K2, V3] = {  
  
        val system : ActorSystem = ActorSystem("MapReduceSystem")  
  
        val orchestrator = system.actorOf(Props(new MapReduce(input, mappingFunction,  
reducingFunction, mapperNumber, reducerNumber)), "orchestrator")  
  
        implicit val timeout: Timeout = Timeout(timeoutValue)  
  
        val futureResult = (orchestrator ? MapReduceCompute()).mapTo[Map[K2, V3]]  
  
        Await.result(futureResult, timeoutValue)  
    }  
}
```

1.1.2. Timer

Objecte *wrapper* de funcions per temporitzar l'execució de les mateixes. Cal passar la funció de forma *lazy* perquè l'execució es realitzi dins de la crida del timer.

```
object Timer {  
    def timeMeasurement[A](f: => A): A = {  
        val startTime = System.nanoTime()  
  
        val result = Try(f)  
  
        val endTime = System.nanoTime()  
  
        val elapsedTime = (endTime - startTime) / 1_000_000_000.0d // ns to s  
  
        println(f"Execution took $elapsedTime%.4f s")  
  
        result match {  
            case Success(value) => value  
            case Failure(e) => throw e  
        }  
    }  
}
```

1.2. Modificació del MapReduce

La modificació de la classe MapReduce proposada consisteix en permetre rebre com a paràmetre el nombre de mappers i reducers que es desitja utilitzar. Així doncs, la definició de la classe ha quedat d'aquesta forma:

```
class MapReduce[K1,V1,K2,V2,V3](  
    input:List[(K1,List[V1])],  
    mapping:(K1,List[V1]) => List[(K2,V2)],  
    reducing:(K2,List[V2])=> (K2,V3),  
    mapperNumber : Int,  
    reducerNumber: Int) extends Actor { //...
```

En la implementació original, s'igualava el nombre de mappers i de reducers a la llargada de l'input rebut i de la mida del diccionari respectivament. La nostra modificació ha consistit en crear dues variables noves que equivalen al nombre de missatges a enviar a mappers i reducers, que sí que equival a les mides descrites anteriorment.

```
var nmappers: Int = mapperNumber  
var missatgesMappersPendants = 0  
var nreducers: Int = reducerNumber  
var missatgesReducersPendants = 0
```

Així doncs, quan la classe MapReduce rep el missatge MapReduceCompute(), genera el nombre especificat de mappers i, utilitzant la funció mod, els envia els missatges de forma balancejada.

```
case MapReduceCompute() =>  
    //println("Hem rebut l'encàrrec")  
    client = sender() // Ens apuntem qui ens ha fet l'encàrrec per enviar-li el missatge  
    més tard.  
  
    mappers = for (i <- 0 until nmappers) yield {  
        context.actorOf(Props(new Mapper(mapping)), "mapper" + i)  
    }  
  
    for(((p1,p2),i)<-input.zipWithIndex) mappers(i % nmappers) ! toMapper(p1: K1,  
p2 :List[V1])  
  
    mappers.foreach(_ ! PoisonPill) // once the mappers have finished working, we kill  
them to save resources
```

```
// Necessitem controlar quan s'han acabat tots els mappers per poder llençar els  
reducers després...
```

```
missatgesMappersPendants = input.length
```

Quan es rep el valor dels mappers, es redueix el nombre de missatges pendants fins que és 0. Llavors, es generen els reducers i se'ls envia el seu contingut.

```
case fromMapper(list_clau_valor:List[(K2,V2)]) =>  
    for ((clau, valor) <- list_clau_valor)  
        dict += (clau -> (valor :: dict(clau)))  
  
    missatgesMappersPendants -= 1  
  
    // Quan ja hem rebut tots els missatges dels mappers:  
    if (missatgesMappersPendants==0)  
    {  
  
        missatgesReducersPendants = dict.size // actualitzem els reducers pendants
```

```

reducers = for (i <- 0 until nreducers) yield
  context.actorOf(Props(new Reducer(reducing)), "reducer"+i)

for (((key:K2, lvalue:List[V2]), i) <- dict.zipWithIndex)
  reducers(i % nreducers) ! toReducer(key, lvalue)

reducers.foreach(_ ! PoisonPill) // once the reducers have finished working,
we kill them to save resources

}

```

Finalment, a mesura que es reben els missatges dels reducers els recull i, un cop rebuts tots, els retorna al “client” i s’atura a ell mateix. Per a més detall sobre com s’inicien i es destrueixen els actors, veure Secció 6.

```

case fromReducer(entradaDiccionari:(K2,V3)) =>
  resultatFinal += entradaDiccionari
  missatgesReducersPendents -= 1

if (missatgesReducersPendents == 0) {
  client ! resultatFinal
  context.stop(self)
}

```

2. Jocs de proves

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (1)
4. Toggle number of reducers (1)
5. Toggle number of non mutually referenced documents to look for (100)
6. Quit

Option: 1

Counting the average number of references...

Time spent calculating average number of references (and reading files): Execution took 3,2809 s

Average number of unique references: 142,28

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (1)
4. Toggle number of reducers (1)
5. Toggle number of non mutually referenced documents to look for (100)
6. Quit

Option: 2

Please enter your query:

piramides

Only one document matches this query: piràmides de gizeh

Execution took 6,5747 s

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (1)
4. Toggle number of reducers (1)
5. Toggle number of non mutually referenced documents to look for (100)
6. Quit

Option: 2

Please enter your query:

qpowiuerpoqiwiuerpoiower

Query was not found in any of the documents.

Execution took 8,6571 s

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (1)
4. Toggle number of reducers (1)
5. Toggle number of non mutually referenced documents to look for (100)
6. Quit

Option: 2

Please enter your query:

tars

```

Step 5: List((polònia,0.0045507672387440495), (imperi rus,0.003790313002656819),
(ucraïnesos,0.0034970494492750533), (moscou,0.0031945660656479673))
Step 4: List((polònia,0.002309639758663209), (imperi rus,0.002160326822398943),
(ucraïnesos,0.0020923419915771704), (moscou,0.0020540110553391136))
Step 3: List((polònia,0.0022507641552203324), (imperi rus,0.002130263113444011),
(ucraïnesos,0.0020817207602701945), (moscou,0.0020373769498682782))
List((polònia,0.0022507641552203324), (imperi rus,0.002130263113444011),
(ucraïnesos,0.0020817207602701945), (moscou,0.0020373769498682782))
((viquipèdia:llista dels 1000 articles fonamentals,viquipèdia:llista dels 1000 articles
fonamentals/1-1-09),0.9857512924379477)
((viquipèdia:llista dels 1000 articles fonamentals/1-04-08,viquipèdia:llista dels 1000
articles fonamentals/1-11-08),0.940878480278472)
((viquipèdia:llista dels 1000 articles fonamentals/1-04-08,viquipèdia:llista dels 1000
articles fonamentals/inicial),0.8864904974929853)
((viquipèdia:llista dels 1000 articles fonamentals/1-11-08,viquipèdia:llista dels 1000
articles fonamentals/inicial),0.8356246755835129)
((viquipèdia:llista d'articles que totes les llengües haurien
de tenir/arxiu2,viquipèdia:llista dels 1000 articles fonamentals/ordre
alfabètic),0.7836624121320085)
((viquipèdia:llista dels 1000 articles fonamentals/1-11-08,viquipèdia:llista dels 1000
articles fonamentals/ordre alfabètic),0.772828098246082)
((viquipèdia:llista dels 1000 articles fonamentals/1-04-08,viquipèdia:llista dels 1000
articles fonamentals/ordre alfabètic),0.7207331642906591)
((viquipèdia:llista d'articles que totes les llengües haurien de tenir/
arxiu2,viquipèdia:llista dels 1000 articles fonamentals/1-11-08),0.7087038592936435)
((viquipèdia:llista d'articles que totes les llengües haurien de tenir/
arxiu2,viquipèdia:llista dels 1000 articles fonamentals/1-04-08),0.7003386077121152)
((albània,història d'albània),0.6423708628685365)
((viquipèdia:llista d'articles que totes les llengües haurien de tenir/
arxiu2,viquipèdia:llista dels 1000 articles fonamentals/inicial),0.600622063412773)
((història de sibèria,sibèria),0.6001996073016831)
((viquipèdia:llista d'articles que totes les llengües haurien de tenir,viquipèdia:llista
d'articles que totes les llengües haurien de tenir/arxiu2),0.5924956431990658)
((viquipèdia:llista dels 1000 articles fonamentals/inicial,viquipèdia:llista dels 1000
articles fonamentals/ordre alfabètic),0.5314329095161409)
((viquipèdia:llista d'articles que totes les llengües haurien de tenir,viquipèdia:llista
dels 1000 articles fonamentals/ordre alfabètic),0.5311116051890166)
((viquipèdia:llista d'articles que totes les llengües haurien de tenir,viquipèdia:llista
dels 1000 articles fonamentals/1-11-08),0.502160042894336)
Execution took 25,3864 s
Execution took 34,2571 s

```

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (1)
4. Toggle number of reducers (1)
5. Toggle number of non mutually referenced documents to look for (100)
6. Quit

Option: 2

Please enter your query:

ocells

```

Step      5:      List((alguerès,0.0025846702317290555),      (joan      miró.
l'escala      de      l'evasió,0.002582196989225138),      (cardiff,0.002572182719241543),
(gal·les,0.0025500829964034055))
Step 4: List((alguerès,0.002272457192243289), (cardiff,0.002269400903767342), (joan
miró. l'escala de l'evasió,0.002267026265413942), (gal·les,0.0022631673181467186))
List((alguerès,0.002272457192243289), (cardiff,0.002269400903767342), (joan miró.
l'escala de l'evasió,0.002267026265413942), (gal·les,0.0022631673181467186))
((joan miró i ferrà,joan miró. l'escala de l'evasió),0.6180988940859181)
((joan miró i ferrà,sèrie barcelona),0.6180555245620877)
Execution took 18,9151 s
Execution took 27,5799 s

```

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (1)
4. Toggle number of reducers (1)
5. Toggle number of non mutually referenced documents to look for (100)
6. Quit

Option: 3

Enter the number of mappers (actual = 1):

16

Number of mappers set to 16.

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (16)
4. Toggle number of reducers (1)
5. Toggle number of non mutually referenced documents to look for (100)
6. Quit

Option: 4

Enter the number of reducers (actual = 1):

16

Number of reducers set to 16.

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (16)
4. Toggle number of reducers (16)
5. Toggle number of non mutually referenced documents to look for (100)
6. Quit

Option: 5

Enter the number of non mutually referenced documents (actual = 100):

25

Number of documents set to 25.

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (16)
4. Toggle number of reducers (16)
5. Toggle number of non mutually referenced documents to look for (25)
6. Quit

Option: 2

Please enter your query:

guerra

Step

5: List((segona guerra mundial,0.01260090692960522), (alemanya,0.0034715157443619537), (frança,0.0033478499052430966), (1945,0.0025125036638271076))

Step

4:

List((segona guerra mundial,0.003188357799981785), (alemanya,0.0014370245897468224), (frança,0.0013161799052430556), (1945,9.748403925854115E-4))

Step

3:

List((segona guerra mundial,0.0023850261601587445), (alemanya,8.805487195685109E-4), (frança,8.127315849427308E-4), (1945,6.073370780005965E-4))

List((segona guerra mundial,0.0023850261601587445), (alemanya,8.805487195685109E-4), (frança,8.127315849427308E-4), (1945,6.073370780005965E-4))

Execution took 2,2100 s

Execution took 29,8398 s

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (16)
4. Toggle number of reducers (16)
5. Toggle number of non mutually referenced documents to look for (25)
6. Quit

Option: 5

Enter the number of non mutually referenced documents (actual = 25):

75

Number of documents set to 75.

Select an option:

1. Count the average number of references of all documents
2. Recommendation based on query
3. Toggle number of mappers (16)
4. Toggle number of reducers (16)
5. Toggle number of non mutually referenced documents to look for (75)
6. Quit

Option: 2

Please enter your query:

segona

```

Step
5: List((segona guerra mundial,0.01260090692960522), (alemanya,0.0034715157443619537),
(frança,0.0033478499052430966), (1945,0.0025125036638271076))
Step
4:
List((segona guerra mundial,0.003188357799981785), (alemanya,0.0014370245897468224),
(frança,0.0013161799052430556), (1945,9.748403925854115E-4))
Step
3:
List((segona guerra mundial,0.0023850261601587445), (alemanya,8.805487195685109E-4),
(frança,8.127315849427308E-4), (1945,6.073370780005965E-4))
List((segona guerra mundial,0.0023850261601587445), (alemanya,8.805487195685109E-4),
(frança,8.127315849427308E-4), (1945,6.073370780005965E-4))
((front oriental de la segona guerra mundial,segona guerra mundial),0.512708640185483)
Execution took 25,8714 s
Execution took 65,6282 s

```

Select an option:

1. Count the average number of references of all documents
 2. Recommendation based on query
 3. Toggle number of mappers (16)
 4. Toggle number of reducers (16)
 5. Toggle number of non mutually referenced documents to look for (75)
 6. Quit
- Option: 6
Exiting...

Process finished with exit code 0

3. MapReduce

Mitjançant l'encapsulació explicada a Secció 1.1.1., hem utilitzat la tècnica del MapReduce pels següents casos:

1. Comptar referències per document
2. Filtrar paraules i referències al content d'un ViquipediaFile
3. Filtrar documents que contenen una *query*
4. Calcular el PR
5. Trobar els documents mútuament referenciats
6. Calcular TF (*word frequency*) de cada document
7. Calcular IDF de cada paraula
8. Calcular TFIDF per document
9. Calcular similituds entre documents

Cadascun d'aquests casos ha requerit un *input*, una funció de mapeig i una de reducció, que es procedeix a explicar en els següents apartats.

Nota: en alguns casos, a fi d'optimitzar execucions, algunes funcions de mapeig i de reducció reben més de dos paràmetres. Aquests son donats en el moment de cridar MR, i son valors comuns per realitzar càlculs per qualsevol element.

3.1. Comptar referències per document

```
val input = for (file <- ProcessFiles.getListOfFiles("viqui_files")) yield (file, Nil)
val result = Timer.timeMeasurement({
    MRWrapper.MR(input,
        MappingReduceFunctions.mappingCountReferences,
        MappingReduceFunctions.reduceCountReferences)
})
```

1. Input

input: List[(File, List[Any])]

2. Mapping

```
def mappingCountReferences(file: File, unusedList: List[Any]): List[(File, Int)] = {
    val refs = ProcessFiles.parseViquipediaFile(file.getPath).refs

    List((file, refs.size))
}
```

3. Reducing

```
def reduceCountReferences(file: File, refs: List[Int]): (File, Int) = {
    (file, refs.sum)
}
```

3.2. Filtrar paraules i referències al content d'un ViquipediaFile

```
val contentFilteredDocuments = Timer.timeMeasurement {
    MRWrapper.MR(
        PRs,
        MappingReduceFunctions.mappingFilterDocuments,
        MappingReduceFunctions.reduceFilterDocuments
    )
}
```

1. Input

input: List[((ViquipediaFile, Double), List[String])]

2. Mapping

```
def mappingFilterDocuments(pair: (ViquipediaFile, Double), refs: List[String]) = {
    List((filterViquipediaFile(pair._1), refs))
}
```

3. Reducing

```
def reduceFilterDocuments(vf: ViquipediaFile, refs: List[List[String]]) = {
    (vf, refs.head)
}
```

3.3. Filtrar documents que contenen una query

```
val input = for (file <- ProcessFiles.getListOfFiles("viqui_files")) yield (file, Nil)
val occurrencesPerFile = Timer.timeMeasurement({
    MRWrapper.MR(input,
        MappingReduceFunctions.mappingFilterContains(query, _, _),
        MappingReduceFunctions.reduceFilterContains)
    })
```

1. Input

input: List[(File, List[Any])]

I a més la query: String que passa directament com a primer paràmetre de la funció de mapeig.

2. Mapping

```
def mappingFilterContains(query: String, file: File, unusedList: List[Any]):
List[(ViquipediaFile, Boolean)] = {
    val vf = ProcessFiles.parseViquipediaFile(file.getPath)
    val filteredQuery = DocumentSimilarity.filterWords(query)
    val aux = DocumentSimilarity.filterWords(vf.content)
    List((vf, filteredQuery.forall(aux.contains(_))))
}
```

3. Reducing

```
def reduceFilterContains(file: ViquipediaFile, containsKeyword: List[Boolean]):
(ViquipediaFile, Boolean) = {
    (file, containsKeyword.forall(bool => bool))
}
```

3.4. Calcular el PR

```
var aux = filteredProves.map(vf => ((vf.title, 1.0d / proves.size), vf.refs))
val ret = MRWrapper.MR(aux,
    MappingReduceFunctions.mappingCalculatePR,
    MappingReduceFunctions.reduceCalculatePR(proves.length, 0.85, _, _))
```

1. Input

input: List[(File, List[Any])]

2. Mapping

```
def mappingCalculatePR(filePR: (String, Double), refs: List[String]): List[(String,
Double)] = {
    filePR match {
        case (str, pr) =>
            refs.map(ref => (ref, pr / refs.size)).appended(str, 0)
    }
}
```

3. Reducing

```
def reduceCalculatePR(totalNumberOfPages: Int, brakeFactor: Double, page: String,
weights: List[Double]): (String, Double) = {
    (page, ((1 - brakeFactor) / totalNumberOfPages) + brakeFactor * weights.sum)
}
```

3.5. Trobar els documents mútuament referenciats

```
val mutuallyReferencedDocPairs =  
    MRWrapper.MR(  
        contentFilteredDocuments.toList,  
        MappingReduceFunctions.mappingObtainMutuallyRefDocuments,  
        MappingReduceFunctions.reduceObtainMutuallyRefDocuments  
    )  
    .filter(_._2) // Keep only mutually referenced pairs  
    .map { case ((a, b), _) => if (a < b) (a, b) else (b, a) }  
    .toSet
```

1. Input

input: `List[(ViquipediaFile, List[String])]`

2. Mapping

```
def mappingObtainMutuallyRefDocuments(file: ViquipediaFile, refs: List[String]):  
List[((String, String), Boolean)] = {  
    if (refs.isEmpty)  
        List(((file.title, ""), false))  
    else  
        refs.flatMap(ref => List(((file.title, ref), true), ((ref, file.title),  
false)))  
}
```

3. Reducing

```
def reduceObtainMutuallyRefDocuments(docs: (String, String), values: List[Boolean]):  
((String, String), Boolean) = {  
    (docs, values.contains(true) && values.contains(false))  
}
```

3.6. Calcular TF (*word frequency*) de cada document

```
val wordFreq = MRWrapper.MR(  
    nonMutuallyReferencedDocs.toList.map(doc => (doc, Nil)),  
    MappingReduceFunctions.mappingCalculateWordFreq,  
    MappingReduceFunctions.reduceCalculateWordFreq  
)
```

1. Input

input: `List[(ViquipediaFile, List[Any])]`

2. Mapping

```
def mappingCalculateWordFreq(file: ViquipediaFile, unusedList: List[Any]):  
List[((String, String), Int)] = {  
    DocumentSimilarity.filterWords(file.content).flatMap(word =>  
List(((file.title, word), 1))).toList  
}
```

3. Reducing

```
def reduceCalculateWordFreq(fileWord: (String, String), counter: List[Int]):  
((String, String), Int) = {  
    (fileWord, counter.sum)  
}
```

3.7. Calcular IDF de cada paraula

```
val documentInverseFreq = MRWrapper.MR(
    nonMutuallyReferencedDocs.toList.map(doc => (doc.content, Nil)),
    MappingReduceFunctions.mappingCalculateInvDocFreq,

MappingReduceFunctions.reduceCalculateInvDocFreq(nonMutuallyReferencedDocs.size, _, _)
)
```

1. Input

input: List[(String, List[Any])]

2. Mapping

```
def mappingCalculateInvDocFreq(content: String, unusedList: List[Any]):
List[(String, Int)] = {
    (for (word <- DocumentSimilarity.filterWords(content)) yield (word,
1)).distinct.toList
}
```

3. Reducing

```
def reduceCalculateInvDocFreq(numberOfDocs: Int, word: String, counter: List[Int]):
(String, Double) = {
    val sum = counter.sum
    (word, Math.log10(numberOfDocs.toDouble / sum))
}
```

3.8. Calcular TFIDF per document

```
val tfIdfPerWord = MRWrapper.MR(
    wordFreq.map { case ((doc, word), freq) =>
        ((doc, word), freq), Nil)
    }.toList,
    MappingReduceFunctions.mappingTfIdfPerDoc(documentInverseFreq, _, _),
    MappingReduceFunctions.reduceTfIdfPerDoc
)
```

1. Input

input: List[((String, String), Int), List[Any]]

2. Mapping

```
def mappingTfIdfPerDoc(inverseFreqs: Map[String, Double], wordCountPerDoc: ((String,
String), Int), unusedList: List[Any]): List[((String, String), Double)] = {
    wordCountPerDoc match {
        case ((doc, word), freq) =>
            List(((doc, word), freq * inverseFreqs.getOrElse(word, 0d)))
    }
}
```

3. Reducing

```
def reduceTfIdfPerDoc(docWord: (String, String), tfIdf: List[Double]): ((String,
String), Double) = {
    (docWord, tfIdf.head)
}
```

3.9. Calcular similituds entre documents

```
MRWrapper.MR(
    nonMutuallyReferencedDocPairs.map { case (doc1Title, doc2Title) =>
        val doc1 = nonMutuallyReferencedDocs.find(_.title == doc1Title).get
        val doc2 = nonMutuallyReferencedDocs.find(_.title == doc2Title).get
        ((doc1, doc2), Nil)
    },
    MappingReduceFunctions.mappingSimilarity(tfIdfPerWord, _, _),
    MappingReduceFunctions.reduceSimilarity
)
```

1. Input

input: `List[((ViquipediaFile, ViquipediaFile), List[Any])]`

2. Mapping

```
def mappingSimilarity(tfIdfPerWord: Map[(String, String), Double], pair:
(ViquipediaFile, ViquipediaFile), unusedList: List[Any]): List[((String, String),
(Map[String, Double], Map[String, Double]))] = {

    val doc1TfIdf = tfIdfPerWord.collect {
        case ((doc, word), tfidf) if doc == pair._1.title => (word, tfidf)
    }

    val doc2TfIdf = tfIdfPerWord.collect {
        case ((doc, word), tfidf) if doc == pair._2.title => (word, tfidf)
    }

    List(((pair._1.title, pair._2.title), (doc1TfIdf, doc2TfIdf)))
}
```

3. Reducing

```
def reduceSimilarity(docPair: (String, String), tfidfs: List[(Map[String, Double],
Map[String, Double])]): ((String, String), Double) = {
    val doc1 = docPair._1
    val doc2 = docPair._2
    val tfidf1 = tfidfs.head._1
    val tfidf2 = tfidfs.head._2

    val allTerms = (tfidf1.keySet ++ tfidf2.keySet).toList

    val vector1 = allTerms.map(term => tfidf1.getOrElse(term, 0.0))
    val vector2 = allTerms.map(term => tfidf2.getOrElse(term, 0.0))

    val dotProduct = vector1.zip(vector2).map { case (a, b) => a * b }.sum
    val magnitude1 = math.sqrt(vector1.map(a => a * a).sum)
    val magnitude2 = math.sqrt(vector2.map(b => b * b).sum)

    val similarity = if (magnitude1 == 0 || magnitude2 == 0)
        0.0
    else
        dotProduct / (magnitude1 * magnitude2)

    ((doc1, doc2), similarity)
}
```

4. Classes i objectes auxiliars

4.1. VikipediaFile

Classe que estructura les dades importants de cada fitxer: amb un títol, el seu contingut, la llista de referències (en forma de text) i el File al moment de llegir.

```
case class VikipediaFile(
  title: String,
  content: String,
  refs: List[String],
  file: File) {
  override def toString: String = s"VikipediaFile(title: $title, filePath: $file,
  nº refs: ${refs.length})"
}
```

4.2. MappingReduceFunctions

Objecte que conté les funcions de mapeig i reducció de tota la aplicació, explicades a la Secció 3.

```
object MappingReduceFunctions {
  mappingX(..., ...): ... = {

  }
  reducingX(..., ...): ... = {

  }
  ...
}
```

4.3. MRWrapper

Objecte explicat a la Secció 1.1.1.

4.4. MapReduceActors

Objecte donat per l'enunciat que conté les classes genèriques Mapper, Reducer i MapReduce que gestiona els missatges entre actors a fi de realitzar el comput del *MapReduce*. Ho fa mitjançant les *case class* MapReduceCompute(), toMapperK1,V1, fromMapperK2,V2, toReducerK2,V2 i fromReducerK2,V3. Ha sofert algunes modificacions explicades a la Secció 1.2.

4.5. Timer

Objecte explicat a la Secció 1.1.2.

4.6. ProcessFiles

Objecte encarregat de tractar amb fitxers: tant llegir-los com processar-los. Està format per la classe VikipediaFile (Secció 4.1.) i les funcions:

- getListOfFiles(dir: String): List[File] que retorna una llista de fitxers donat un directori.
- parseVikipediaFile(filename: String): VikipediaFile que llegeix el document segons el títol i en filtra les seves referències.
- loadCatalanStopWords(): Set[String] que llegeix el fitxer de *stop words* en català i les estructura per poder tractar-les.
- filterVikipediaFile(vf: VikipediaFile): VikipediaFile que donat un fitxer, en filtra les referències, deixant només el contingut rellevant.

5. Taula de rendiment

Les següents taules han estat calculades a partir d'una màquina amb les següents característiques:

- **CPU:** AMD Ryzen 7 5800X (8 cores, 16 threads)
- **RAM:** 4x8GB DDR4 3200Mhz
- **SSD:** Crucial NVME MX500: 1TB, fins a 560MB/s (M.2 2280SS, 3D NAND, SATA)
- **GPU:** Nvidia RTX 2070 SUPER (no hauria de ser rellevant pel problema)

		Reducers			
Mappers		1	4	10	20
	1	3,3039 s	3,2581 s	3,2676 s	3,3067 s
	4	3,2120 s	3,2761 s	3,2098 s	3,2583 s
	10	3,2151 s	3,2320 s	3,3442 s	3,3134 s
	20	3,3349 s	3,3502 s	3,3339 s	3,4076 s

Taula 1: Taula resultant de processar el nombre mitjà de referències per 1, 4, 10 i 20 actors (inclou el temps de lectura dels fitxers).

		Reducers			
Mappers		1	4	10	20
	1	13,5981 s	13,6977s	14,3011 s	14,6217 s
	4	14,1457 s	13,8410 s	13,9532 s	13,5111 s
	10	15,5013 s	13,6125 s	14,1392 s	14,0626 s
	20	14,0000 s	14,9467 s	14,8281 s	13,8578 s

Taula 2: Taula resultant de processar el PR per la query “guerra” per 1, 4, 10 i 20 actors (inclou el temps de lectura dels fitxers).¹

		Reducers			
Mappers		1	4	10	20
	1	64,8530 s	67,5586 s	70,0484 s	66,0386 s
	4	69,1306 s	66,6735 s	67,9857 s	69,4181 s
	10	74,0833 s	70,5694 s	70,1450 s	72,5849 s
	20	73,8949 s	71,1827 s	68,3373 s	68,9002 s

Taula 3: Taula resultant de processar el nombre de documents similars no mútuament referenciats per la query “guerra” per 1, 4, 10 i 20 actors (no inclou el temps de lectura dels fitxers).

Cal destacar que la query “guerra” és la més exigent, doncs tots els documents la contenen. Per tant, podem assegurar que el càlcul de la similitud ha estat realitzat amb 100 documents i que el càlcul del PR ha tingut en compte tots els documents.

Malauradament, els resultats no han estat els que esperavem; de fet, han estat tot el contrari. Sembla ser que el nombre d'actors, almenys en el nostre sistema, és totalment irrellevant. En executar el codi amb un únic actor o varis, es duu a terme paral·lelització en el mateix ordinador i s'obtenen unes mesures molt semblants. Així doncs, sembla ser que el fet de tenir diferents actors únicament seria útil quan es disposen de varis dispositius de còmput als quals es pot enviar la informació.

¹El temps inclou les diferents iteracions necessàries fins a convergir, que en aquest cas és de 3.

6. Creació i tancament del sistema d'Actors

La nostra proposta de creació i destrucció d'actors és la següent.

1. Quan es crida `MRWrapper.MR`, es genera un nou sistema d'actors anomenat `MapReduceSystem`. Aquest sistema es manté actiu durant l'*scope* de la crida `MR`.
2. Es crea l'actor *orchestrator*, que quan rep la crida `MapReduceCompute` crea els mappers i reducers especificats.
3. La quantitat inicial de claus del map d'input s'envia a cadascun dels actors, de forma balancejada.
4. Inmediatament després de l'enviament dels missatges, se'ls envia també una `PoisonPill` que els destruirà un cop els missatges de la seva cua acabin. D'aquesta forma, s'alliberen recursos del sistema un cop els mappers han acabat el càlcul.
5. La mateixa estratègia s'aplica amb els reducers, un cop tots els mappers han acabat i s'ha rebut i processat el mapeig.
6. Finalment, un cop els mappers han acabat, la classe `MapReduce` envia el resultat final a l'*orchestrator* i s'autodestruïx, ja que no espera rebre cap missatge més.

La única limitació que té aquesta proposta d'implementació és que un `MapReduce` no podria cridar un altre `MapReduce` degut a un conflicte de noms. Si bé es podrien implementar modificacions per a permetre-ho (gràcies als Singletons de Scala), no ho hem trobat necessari.

7. Altres consideracions

7.1. Filtratge de referències

Si bé és cert que pel càlcul del *PageRank* seguim les indicacions donades pel professor, per a calcular la similitud entre documents no hem sabut trobar enlloc què cal filtrar exactament per a realitzar la comparació (comptar o no amb certs elements dins el llenguatge de marcat que utilitza Viquipèdia i podrien ser útils tals com peus de foto, notes, etc).

Hem pres la decisió de, donat el contingut dels fitxers, eliminar les referències entre `[[]]` que contenen uns ":" (ja que aquestes inclouen un fitxer, una imatge, etc.) i les anotacions entre `{ { } }` que contenen informació majoritàriament irrellevant. La resta de referències les hem mantingut, doncs moltes vegades el text d'aquestes és rellevant per a la comparació dels textos.

Tanmateix, del contingut dels fitxers es lleven els caràcters que no són ni dígit, ni lletres, ni espais. S'ha realitzat d'aquesta forma per evitar que paraules com "avió" i "avió:" siguin considerades diferents. En últim lloc, filtrem també les *stopwords* de la llengua catalana que no aporten significat, malgrat que la seva importància ens els documents seria mínima degut a la alta probabilitat que apareguin en quasi tots els documents del conjunt.

En últim lloc, hem detectat que alguns dels documents contenen, dins el contingut, codi HTML per a formatar taules i altres estructures. La filtració d'aquests elements ens ha semblat molt complexa i distant de l'objectiu de la pràctica. Així doncs, és possible que apareguin paraules com `div`, `tr` i semblants quan es generen aquests elements (que en aparèixer de forma freqüent no haurien de prendre protagonisme). Cal remarcar que, degut al filtratge, elements com `<`, `>` o similars mai romandran amb les paraules usades per a calcular l'*IDF*.

7.2. Ús de MapReduce

Si bé és cert que l'enunciat explicita que cal utilitzar un ús abundant de MapReduces per a paral·lelitzar la feina al màxim, hem considerat que, en certes circumstàncies, cridar un `MapReduce`, generar un mapa i el diccionari i retornar els fitxers seria més costós que calcular el propi output que necessitem des del "gestor". Aquest podria ser el cas, per exemple, de la mitjana en el primer apartat, on el recompte

de referències es fa a través del MapReduce però el càlcul de la mitjana com a tal el fa el gestor, ja que de totes formes no es pot fer la divisió final fins que s'obté la suma.

7.3. Reutilització de material de la primera entrega

El codi de la primera entrega ens ha estat útil per a realitzar el filtratge de paraules del contingut de cadascun dels fitxers llegits, mitjançant la funció `filterWords()`.

No gens menys, el cos de la funció `cosineSim()` s'ha reestructurat en varis MapReduces per a paral·lelitzar el codi per a múltiples documents.

Com a últim punt a destacar, hem decidit que utilitzar la funció `ngrams` per a trobar si un document conté la *query* era innecessari, i amb la funció `filterWords()` ha estat suficient.