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

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 missatgesMappersPendents = 0
var nreducers: Int = reducerNumber
var missatgesReducersPendents = 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 lencarrec")
  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...
  missatgesMappersPendents = input.length
```

Quan es rep el valor dels mappers, es redueix el nombre de missatges pendents 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)))

    missatgesMappersPendents -= 1

    // Quan ja hem rebut tots els missatges dels mappers:
    if (missatgesMappersPendents==0)
    {

        missatgesReducersPendents = dict.size // actualitzem els reducers pendents
```

```
        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: 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):
1000
Number of documents set to 1000.

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 (1000)
6. Quit
Option: 1
Counting the average number of references...
Time spent calculating average number of references (and reading files): Execution took
3,1651 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 (16)
```

```
4. Toggle number of reducers (16)
5. Toggle number of non mutually referenced documents to look for (1000)
6. Quit
Option: 2
Please enter your query:
tars

Step 1: List((polònia,0.0045507672387440495), (imperi rus,0.003790313002656819),
(ucraïnesos,0.0034970494492750533), (moscou,0.0031945660656479673))
Step 2: 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))

HIGHEST PAGE RANK DOCUMENTS (4):
polònia                                                 : 2,251e-03
imperi rus                                              : 2,130e-03
ucraïnesos                                              : 2,082e-03
moscou                                                  : 2,037e-03
=======


LIST OF NON-MUTUALLY REFERENCED DOCUMENTS WITH > 0.5 COSINE SIMILARITY:

viquipèdia:llista dels 1000 articles fonamentals            - viquipèdia:llista dels 1000 articles fonamentals/1-1-09      : 0,9858
viquipèdia:llista dels 1000 articles fonamentals/1-04-08    - viquipèdia:llista dels 1000 articles fonamentals/1-11-08     : 0,9409
viquipèdia:llista dels 1000 articles fonamentals/1-04-08    - viquipèdia:llista dels 1000 articles fonamentals/inicial     : 0,8865
viquipèdia:llista dels 1000 articles fonamentals/1-11-08    - viquipèdia:llista dels 1000 articles fonamentals/inicial     : 0,8356
viquipèdia:llista d'articles que totes les llengües haurien - viquipèdia:llista dels 1000 articles fonamentals/ordre alfab : 0,7837
viquipèdia:llista dels 1000 articles fonamentals/1-11-08    - viquipèdia:llista dels 1000 articles fonamentals/ordre alfab : 0,7728
viquipèdia:llista dels 1000 articles fonamentals/1-04-08    - viquipèdia:llista dels 1000 articles fonamentals/ordre alfab : 0,7207
viquipèdia:llista d'articles que totes les llengües haurien - viquipèdia:llista dels 1000 articles fonamentals/1-11-08     : 0,7087
viquipèdia:llista d'articles que totes les llengües haurien - viquipèdia:llista dels 1000 articles fonamentals/1-04-08     : 0,7003
albània                                                     - història d'albània                                           : 0,6424
viquipèdia:llista d'articles que totes les llengües haurien - viquipèdia:llista dels 1000 articles fonamentals/inicial     : 0,6006
història de sibèria                                         - sibèria                                                      : 0,6002
viquipèdia:llista d'articles que totes les llengües haurien - viquipèdia:llista d'articles que totes les llengües haurien  : 0,5925
viquipèdia:llista dels 1000 articles fonamentals/inicial    - viquipèdia:llista dels 1000 articles fonamentals/ordre alfab : 0,5314
viquipèdia:llista d'articles que totes les llengües haurien - viquipèdia:llista dels 1000 articles fonamentals/ordre alfab : 0,5311
viquipèdia:llista d'articles que totes les llengües haurien - viquipèdia:llista dels 1000 articles fonamentals/1-11-08     : 0,5022

=======

SIMILARITY CALCULATION: Execution took 4,4076 s
PR + SIMILARITY CALCULATION: Execution took 10,5850 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 (1000)
6. Quit
Option: 2
Please enter your query:
piramides

Only one document matches this query: piràmides de gizeh
PR + SIMILARITY CALCULATION: Execution took 7,9219 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 (1000)
6. Quit
Option: 2
Please enter your query:
ioqpweuriqpoweurpoiqweur

Query was not found in any of the documents.
PR + SIMILARITY CALCULATION: Execution took 7,8766 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 (1000)
6. Quit
Option: 2
Please enter your query:
guerra

Step
1: List((segona guerra mundial,0.01260090692960522), (alemanya,0.0034715157443619537),
(frança,0.0033478499052430966), (1945,0.0025125036638271076))
Step                                                                    2:
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))

HIGHEST PAGE RANK DOCUMENTS (4):
segona guerra mundial                          : 2,385e-03
alemanya                                       : 8,805e-04
frança                                         : 8,127e-04
1945                                           : 6,073e-04
=======


LIST OF NON-MUTUALLY REFERENCED DOCUMENTS WITH > 0.5 COSINE SIMILARITY:

partit socialdemòcrata d'alemanya              - unió demòcrata cristiana d'alemanya          : 0,8539
partit democràtic lliure                       - partit socialdemòcrata d'alemanya            : 0,7966
incendi del reichstag                          - llei de capacitació                          : 0,7844
decret de l'incendi del reichstag              - incendi del reichstag                        : 0,7791
partit democràtic lliure                       - unió social cristiana de baviera             : 0,7763
partit socialdemòcrata d'alemanya              - unió social cristiana de baviera             : 0,7723
panzer ii                                      - panzer iii                                   : 0,7372
batalla de l'estret de dinamarca               - cuirassat bismarck                           : 0,7217
josef dietrich                                 - paul hausser                                 : 0,6934
panzer ii                                      - panzer iv                                    : 0,6898
ferdinand porsche                              - porsche                                      : 0,6788
panzer i                                       - panzer iii                                   : 0,6450
josef dietrich                                 - schutzstaffel                                : 0,6358
corea                                          - corea del sud                                : 0,6201
guerra d'hivern                                - guerra de continuació                        : 0,6145
paul hausser                                   - schutzstaffel                                : 0,6080
corea del nord                                 - londres                                      : 0,5954
messerschmitt bf 109                           - messerschmitt bf 110                         : 0,5947
panzer i                                       - panzer iv                                    : 0,5922
```

```
berlín                                                  - saxònia                                                  : 0,5917
medalla del 40è aniversari de la victòria en la gran guerra  - medalla dels treballadors distingits                : 0,5835
paul hausser                                            - waffen-ss                                                : 0,5808
energia nuclear                                         - reactor nuclear                                          : 0,5796
medalla dels treballadors distingits                    - medalla pel servei de combat                             : 0,5765
medalla del 30è aniversari de la victòria en la gran guerra  - medalla dels treballadors distingits                : 0,5706
grup volkswagen                                         - volkswagen escarabat                                     : 0,5697
junkers ju 52                                           - junkers ju 88                                            : 0,5548
front oriental de la segona guerra mundial              - segona guerra mundial                                    : 0,5452
ciutat de luxemburg                                     - luxemburg                                                : 0,5380
berlín                                                  - tailàndia                                                : 0,5370
medalla del 20è aniversari de la victòria en la gran guerra  - medalla dels treballadors distingits                : 0,5335
rin                                                     - àfrica                                                   : 0,5308
regne de romania                                        - romania                                                  : 0,5308
12a divisió panzer ss hitlerjugend                      - schutzstaffel                                            : 0,5288
panzer                                                  - panzer iv                                                : 0,5244
12a divisió panzer ss hitlerjugend                      - paul hausser                                             : 0,5058
arma nuclear                                            - energia nuclear                                          : 0,5048
12a divisió panzer ss hitlerjugend                      - josef dietrich                                           : 0,5044
royal navy                                              - àfrica                                                   : 0,5012

=======

SIMILARITY CALCULATION: Execution took 112,5994 s
PR + SIMILARITY CALCULATION: Execution took 122,6219 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 (1000)
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

Per a cada fitxer compta el nombre de referències i en retorna la suma.

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

val averageReferenceCount = if (result.nonEmpty) result.values.sum.toDouble /
result.size else 0.0d
```

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`

Per a cada fitxer filtra el contingut mitjançant la crida `filterViquipediaFile()`.

```
val contentFilteredDocuments =
    MRWrapper.MR(
        PRs.map(p => (p, Nil)),
        MappingReduceFunctions.mappingFilterDocuments,
        MappingReduceFunctions.reduceFilterDocuments
    )
```

on `PRs` és una `List[ViquipediaFile]`.

1. Input

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

2. Mapping

```
def mappingFilterDocuments(vf: ViquipediaFile, unusedList: List[Any]):
List[(ViquipediaFile, List[Any])] = {
    List((filterViquipediaFile(vf), unusedList))
}
```

3. Reducing

```
def reduceFilterDocuments(vf: ViquipediaFile, unusedList: List[List[Any]]):
(ViquipediaFile, Nil.type ) = {
    (vf, Nil)
}
```

3.3. Filtrar documents que contenen una *query*

Per a cada document i una *query*, retorna si el document conté aquesta *query* després de filtrar-ne les paraules.

```
val occurrencesPerFile = MRWrapper.MR(
    ProcessFiles.getListOfFiles("viqui_files").map(file => (file, Nil)),
    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

Per a cada referència genera el seu valor actual de PR. Afegeix el mateix document amb 0.0 per a què es tingui en compte al reduce. El reduce recull els valors de cada referència i calcula el PR total.

```
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[((String, Double), List[String])]
```

2. Mapping

```
def mappingCalculatePR(filePR: (String, Double), refs: List[String]): List[(String,
Double)] = {
    filePR match {
        case (doc, pr) =>
            if (refs.isEmpty) List((doc, 0.0))
            else refs.map(ref => (ref, pr / refs.size)) :+ (doc, 0.0)
    }
}
```

3. Reducing

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

3.5. Trobar els documents mútuament referenciats

Per a cadascun dels documents, es genera un valor `true` del document cap a les referències i `false` de les referències cap al document. Al reduce, si el document conté un valor `true` i un de `false` implica que està mútuament referenciat.

```
val mutuallyReferencedDocPairs =
    MRWrapper.MR(
            contentFilteredDocuments.toList,
            MappingReduceFunctions.mappingObtainMutuallyRefDocuments,
            MappingReduceFunctions.reduceObtainMutuallyRefDocuments
        )
        .filter(_._2).keySet
        .map { case (a, b) => if (a < b) (a, b) else (b, a) }
```

1. Input

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

2. Mapping

```
def mappingObtainMutuallyRefDocuments(file: ViquipediaFile, unusedList: List[Any]):
List[((String, String), Boolean)] = {
    if (file.refs.isEmpty)
        List(((file.title, ""), false))
    else
        file.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

Per a cada paraula de cada document, es genera una tupla document-paraula amb valor 1. El reduce retorna la suma de 1s, és a dir la *word frequency*.

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

1. Input

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

2. Mapping

```
def mappingCalculateWordFreq(file: ViquipediaFile, unusedList: List[Any]):
List[((ViquipediaFile, String), Int)] = {
    file.content.split("\\W").flatMap(word => List(((file, word), 1))).toList
}
```

3. Reducing

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

3.7. Calcular IDF de cada paraula

El mapper calcula a quants documents surt la paraula, i el reducer calcula el logaritme i la divisió.

```
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 <- content.split("\\W").distinct) yield (word, 1)).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

Realitza la multiplicació `tf_idf` i el reducer ho retorna en format de `Map` per a tenir un accés $O(1)$ a l'hora de calcular la similitud.

```
val tfIdfPerDoc = MRWrapper.MR(
    wordFreq.map { case ((doc, word), freq) =>
        (((doc.title, 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(doc: String, wordTfIdfList: List[(String, Double)]): (String,
Map[String, Double]) = {
    (doc, wordTfIdfList.toMap)
}
```

3.9. Calcular similituds entre documents

El mapper retorna el Map corresponent al document número 1.

El reducer, un cop té el `tf_idf` de cada document, calcula el producte escalar i la normalització sense generar vectors auxiliars, mitjançant acumuladors, per estalviar memòria i optimitzar l'espai de la crida. Finalment retorna el valor de similitud (*cosine sim*) entre la parella de documents.

```
MRWrapper.MR(
        nonMutuallyReferencedDocPairs.map { case (doc1Title, doc2Title) =>
            ((doc1Title, doc2Title), Nil)
        }.toList,
        MappingReduceFunctions.mappingSimilarity(tfIdfPerDoc, _, _),
        MappingReduceFunctions.reduceSimilarity
)
.filter(_._2 >= 0.5)
.toList
.sortBy(-_._2)
.foreach(p => println(f"${p._1._1.take(60)}%-60s - ${p._1._2.take(60)}%-60s :
${p._2}%.4f"))
```

1. Input

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

2. Mapping

```
def mappingSimilarity(tfIdfPerDoc: Map[String, Map[String, Double]], pair: (String,
String), unusedList: List[Any]): List[((String, String), (Map[String, Double],
Map[String, Double]))] = {
    //tf_idf vectors
    val doc1Map = tfIdfPerDoc.getOrElse(pair._1, Map.empty)
    val doc2Map = tfIdfPerDoc.getOrElse(pair._2, Map.empty)

    List(((pair._1, pair._2), (doc1Map, doc2Map)))
}
```

3. Reducing

```
def reduceSimilarity(docPair: (String, String), tfidfs: List[(Map[String, Double],
Map[String, Double])]): ((String, String), Double) = {
    // Obtenir els maps de la mapping function
    val tfidf1 = tfidfs.head._1
    val tfidf2 = tfidfs.head._2

    // Acumuladors
    var dotProduct = 0.0
    var magnitude1 = 0.0
    var magnitude2 = 0.0

    // Iterar primer tf-idf
    for ((term, value1) <- tfidf1) {
        val value2 = tfidf2.getOrElse(term, 0.0)

        // Actualitzar dotProduct i magnituds.
        dotProduct += value1 * value2
        magnitude1 += value1 * value1
    }

    // Iterar segon tf-idf
    for (value2 <- tfidf2.values) {
        magnitude2 += value2 * value2
    }

    // Calcul final
    val similarity =
        if (magnitude1 == 0 || magnitude2 == 0) 0.0
        else dotProduct / (math.sqrt(magnitude1) * math.sqrt(magnitude2))

    (docPair, similarity)
}
```

4. Classes i objectes auxiliars

4.1. ViquipediaFile

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 ViquipediaFile(
  title: String,
  content: String,
  refs: List[String],
  file: File) {
       override def toString: String = s"ViquipediaFile(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()`, `toMapper[K1,V1](K1,V1)`, `fromMapper[K2,V2](K2,V2)`, `toReducer[K2,V2](K2,V2)` i `fromReducer[K2,V3](K2,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 `ViquipediaFile` (Secció 4.1.) i les funcions:

- `getListOfFiles(dir: String): List[File]` que retorna una llista de fitxers donat un directori.
- `parseViquipediaFile(filename: String): ViquipediaFile` 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.
- `filterViquipediaFile(vf: ViquipediaFile): ViquipediaFile` 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			
		1	**4**	**10**	**20**
Mappers	**1**	3,2010 s[1]	2,5055 s	2,5074 s	2,5407 s
	4	2,5295 s	2,5617 s	2,5717 s	2,5782 s
	10	2,6145 s	2,6059 s	2,5945 s	2,6500 s
	20	2,6122 s	2,6193 s	2,6270 s	2,5944 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			
		1	**4**	**10**	**20**
Mappers	**1**	7,5094 s	9,929 s	9,861 s	10,0913 s
	4	10,1213 s	10,1772 s	10,1333 s	10,2018 s
	10	10,3297 s	10,4065 s	10,3070 s	10,1897 s
	20	10,3039 s	10,3180 s	10,3750 s	10,2744 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).[2]

		Reducers			
		1	**4**	**10**	**20**
Mappers	**1**	4,9087 s	4,6784 s	3,6732 s	4,8571 s
	4	5,1221 s	5,1612 s	5,2118 s	5,2812 s
	10	5,3122 s	5,1907 s	5,3282 s	5,4040 s
	20	5,2363 s	4,9710 s	5,5246 s	5,3389 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.

[1] El valor és lleugerament superior perquè és la primera vegada que es llegeixen els fitxers i cal fer *buffering*.

[2] El 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. Immediatament 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'autodestrueix, 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 Viquipedia 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 son ni dígits, 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 costos 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 `ngrames` per a trobar si un document conté la *query* era innecessari, i amb la funció `filterWords()` ha estat suficient.