UNIVERSITÉ D'AVIGNON
ET DES PAYS DE VAUCLUSE

M2 ILSEN – 2019/20

UE Ingénierie du document et de l'information

UCE Indexation & recherche

Vincent Labatut

TP 4 | Correction

1 Vérité terrain

Exercice 1

```
1  public static List<Posting> getPostingsFromFileNames(List<String> fileNames)
2  {  List<Posting> result = new LinkedList<Posting>();
3     File folder = new File(getCorpusFolder());
4     String fn[] = folder.list();
5     List<String> fn2 = new ArrayList<String>(Arrays.asList(fn));
6     Collections.sort(fn2);
7
8     for(String fileName: fileNames)
9     {  int docId = fn2.indexOf(fileName);
10       Posting posting = new Posting(docId);
11       result.add(posting);
12    }
13
14    return result;
15 }
```

Exercice 3

```
1  public GroundTruth() throws ParserConfigurationException, SAXException, IOException
2  {  String groundTruthFile = FileTools.getGroundTruthFile();
3     System.out.println("Reading ground truth file "+groundTruthFile);
4
5     // initialisation des listes
6     queries = new ArrayList<String>();
7     postingLists = new ArrayList<List<Posting>>();
8
9     // ouverture du fichier XML en lecture et parsing
10    DocumentBuilderFactory documentBuilderFactory =
11                                 DocumentBuilderFactory.newInstance();
12    DocumentBuilder documentBuilder =
13                            documentBuilderFactory.newDocumentBuilder();
14    Document document = documentBuilder.parse(groundTruthFile);
15    Element rootElt = document.getDocumentElement();
16
17    // on traite chaque requête listée dans le fichier
18    NodeList queryElts = rootElt.getElementsByTagName("query");
19    System.out.print("Found "+queryElts.getLength()+" queries ( ");
20    for(int i=0;i<queryElts.getLength();i++)
21    {  // on stocke la requête elle-même
22       Element queryElt = (Element)queryElts.item(i);
23       String queryStr = queryElt.getAttribute("expr").trim();
24       queries.add(queryStr);
25       List<String> nameList = new ArrayList<String>();
26
27       // on lit les noms des fichiers pertinents pour la requête
28       NodeList docElts = queryElt.getElementsByTagName("doc");
```

```
29        System.out.print(docElts.getLength()+" ");
30        for(int j=0;j<docElts.getLength();j++)
31        {  Element docElt = (Element)docElts.item(j);
32           String file = docElt.getTextContent().trim();
33           nameList.add(file);
34        }
35        // on convertit les noms de fichiers en postings
36        List<Posting> postingList =
37                              FileTools.getPostingsFromFileNames(nameList);
38        Collections.sort(postingList);
39        postingLists.add(postingList);
40     }
41     System.out.println(")");
42  }
```

2 Mesures de performances

Exercice 4

```
1  public AbstractEvaluator() throws ParserConfigurationException, SAXException,
       IOException
2  {  groundTruth = new GroundTruth();
3  }
```

Exercice 5

```
1  private Map<MeasureName,Float> evaluateQueryAnswer(int queryId, List<Posting>
       answer)
2  {  List<Posting> reference = groundTruth.getPostingList(queryId);
3
4     // on pourrait plutot utiliser la méthode processConjunction
5     // du moteur de recherche, mais ici, la vitesse n'est pas une priorité
6     List<Posting> inter = new ArrayList<Posting>(reference);
7     inter.retainAll(answer);
8
9     float precision = 0;
10    float recall = 1;
11    float fmeasure = 0;
12
13    int tp = inter.size();
14    int fp = answer.size() - tp;
15    int fn = reference.size() - tp;
16
17    if(!answer.isEmpty())
18       precision = tp/(float)(tp+fp);
19    if(!reference.isEmpty())
20       recall = tp/(float)(tp+fn);
21    if(precision!=0 && recall!=0)
22       fmeasure = 2*precision*recall / (precision+recall);
23
24    Map<MeasureName,Float> result = new HashMap<MeasureName,Float>();
25    result.put(MeasureName.PRECISION, precision);
26    result.put(MeasureName.RECALL, recall);
27    result.put(MeasureName.F_MEASURE, fmeasure);
28
29    return result;
30 }
```

Exercice 6

```java
private List<Map<MeasureName,Float>> evaluateQueryAnswers(List<List<Posting>>
    answers)
{  List<Map<MeasureName,Float>> result = new ArrayList<Map<MeasureName,Float>>();
   float totalPre = 0;
   float totalRec = 0;
   float totalFM = 0;

   // on traite chaque requête de la vérité terrain
   int queryId = 0;
   for(List<Posting> answer: answers)
   {  Map<MeasureName,Float> map = evaluateQueryAnswer(queryId, answer);
      totalPre = totalPre + map.get(MeasureName.PRECISION);
      totalRec = totalRec + map.get(MeasureName.RECALL);
      totalFM = totalFM + map.get(MeasureName.F_MEASURE);
      result.add(map);
      queryId++;
   }

   // on calcule les valeurs moyennes
   Map<MeasureName,Float> map = new HashMap<MeasureName,Float>();
   map.put(MeasureName.PRECISION, totalPre/answers.size());
   map.put(MeasureName.RECALL, totalRec/answers.size());
   map.put(MeasureName.F_MEASURE, totalFM/answers.size());
   result.add(map);

   return result;
}
```

Exercice 7

```java
protected void writePerformances(List<Map<MeasureName,Float>> values) throws
    FileNotFoundException, UnsupportedEncodingException
{  // on ouvre le fichier en écriture
   String fileName = FileTools.getPerformanceFile();
   File file = new File(fileName);
   FileOutputStream fos = new FileOutputStream(file);
   OutputStreamWriter osw = new OutputStreamWriter(fos,"UTF-8");
   PrintWriter writer = new PrintWriter(osw);

   // on traite chaque map l'une après l'autre
   for(Map<MeasureName,Float> map: values)
   {  boolean first = true;
      for(MeasureName measureName: MeasureName.values())
      {  float val = map.get(measureName);
         if(first)
            first = false;
         else
            writer.print("\t");
         writer.print(val);
      }
      writer.println();
   }

   // on ferme le flux
   writer.close();
}
```

Exercice 8

```java
public List<Map<MeasureName,Float>> evaluateEngine(AndQueryEngine engine)
```

```
  {  System.out.println("Evaluating the search engine");
     // on traite chaque requête d'évaluation
     List<List<Posting>> answers = new ArrayList<List<Posting>>();
     List<String> queries = groundTruth.getQueries();
     for(String query: queries)
     {  List<Posting> answer = engine.processQuery(query);
        answers.add(answer);
     }

     // on calcule les performances correspondant aux réponses
     List<Map<MeasureName,Float>> result = evaluateQueryAnswers(answers);

     return result;
  }
```

3 Évaluation

Exercice 9

```
  private static void testEvaluation() throws ParserConfigurationException,
      SAXException, IOException, ClassNotFoundException
  {  // chargement de l'index
     AbstractIndex index = AbstractIndex.read();
     AndQueryEngine engine = new AndQueryEngine(index);

     // chargement de la vérité terrain et évaluation
     BooleanEvaluator evaluator = new BooleanEvaluator();
     List<Map<MeasureName,Float>> perfs = evaluator.evaluateEngine(engine);
     List<String> queries = evaluator.getGroundTruth().getQueries();

       NumberFormat nf = NumberFormat.getNumberInstance(Locale.ENGLISH);
       nf.setMinimumFractionDigits(2);
       nf.setMaximumFractionDigits(2);

     // on calcule et affiche les performances
     for(MeasureName measureName: MeasureName.values())
        System.out.print(measureName+"\t");
     System.out.println();
     for(int i=0;i<queries.size();i++)
     {  String query = queries.get(i);
        Map<MeasureName,Float> map = perfs.get(i);
        for(MeasureName measureName: MeasureName.values())
           System.out.print(nf.format(map.get(measureName))+"\t\t");
        System.out.println(query);
     }
     System.out.println("-------------------------------------------------");
     Map<MeasureName,Float> map = perfs.get(perfs.size()-1);
     for(MeasureName measureName: MeasureName.values())
        System.out.print(nf.format(map.get(measureName))+"\t\t");
     System.out.println();
  }
```

La sortie complète est la suivante :

```
PRECISION   RECALL    F_MEASURE
0.00        0.00      0.00        Arthroscopic treatment of cruciate ligament injuries
0.00        0.00      0.00        Complications of arthroscopic interventions
0.00        0.00      0.00        Pathophysiology and prophylaxis of arthrofibrosis
0.00        0.00      0.00        HIV Epidemiology, Risk Assessment
0.00        0.00      0.00        Patient-controlled analgesia indications and limits
```

```
0.00        0.00      0.00        Priming with non-depolarizing muscle relaxants
0.83        0.62      0.71        Complications after laparoscopic cholecystectomy
0.00        0.00      0.00        Heparin induced thrombocytopenia, diagnosis and...
0.50        0.17      0.25        Diagnostic in Lyme disease
0.35        0.29      0.32        Treatment of acute myocardial infarction
1.00        0.11      0.19        Catheter ablation and cardiac mapping
1.00        0.06      0.12        Diagnostic approach in injuries of the shoulder
0.00        0.00      0.00        Differential diagnosis in infertility
0.00        0.00      0.00        Approach of the correction of deformities in...
0.33        0.24      0.28        Treatment of squamous cell carcinoma
1.00        0.07      0.12        Associated diseases with insulin dependent diabetes...
1.00        0.19      0.31        Therapy in chronic low back pain
0.53        0.41      0.46        Treatment of ventricular tachycardia
0.33        0.05      0.08        Indication for implantable cardioverter defibrillator
0.00        0.00      0.00        Diagnostic in acute and chronic myocarditis
0.50        0.07      0.12        Cause of dysphagia
1.00        0.14      0.25        Treatment of sensorineural hearing loss (SNHL)
1.00        0.12      0.22        Complications of surgical repair of aortic aneurysm
0.00        0.00      0.00        Treatment of psychosomatical patients
0.00        0.00      0.00        New approach in cruciate ligament surgery
---------------------------------------
0.38        0.10      0.14
```

On peut voir que ces performances sont plutôt mauvaises : le moteur ne trouve simplement aucun document pour un grand nombre de requêtes. Ceci correspond aux cas où la précision et le rappel sont tous les deux nuls.

En moyenne, la précision est nettement plus élevée que le rappel, ce qui signifie que les documents renvoyés, quand il y en a, ont tendance à être pertinents. En comparaison, le moteur rate de nombreux documents pertinents (qui ne sont donc pas renvoyés).

Ceci est probablement dû à cooccurrence de deux facteurs : on effectue une comparaison exacte des chaînes lors de la comparaison des termes, et on requiert que tous les termes apparaissent dans un document pour que la requête soit vérifiée. Les performances pourraient s'améliorer si on relâche une de ces requêtes, ou les deux.

Exercice 10

```
1   private void splitQuery(String query, List<List<Posting>> result)
2   {  // on tokénize la requête
3      Tokenizer tokenizer = index.getTokenizer();
4      List<String> types = tokenizer.tokenizeString(query);
5
6      // on normalise chaque type
7      Normalizer normalizer = index.getNormalizer();
8      System.out.print(" Normalizing:");
9      for(String type: types)
10     {  // la normalisation du type donne le terme (ou null)
11        String term = normalizer.normalizeType(type);
12        int postNbr = 0;
13        if(term!=null)
14        {  // on récupère l'entrée associée au terme dans l'index
15           IndexEntry entry = index.getEntry(term);
16           // si pas dans l'index, on utilise une liste vide
17           if(entry==null)
18              result.add(new ArrayList<Posting>());
19           // sinon, on prend sa liste de postings
20           else
21           {  List<Posting> postings = entry.getPostings();
22              result.add(postings);
```

```
            postNbr = postings.size();
          }
        }
        System.out.print(" \""+term+"\""+"("+postNbr+")");
      }
      System.out.println();
    }
```

Exercice 11

```
  private List<Posting> processConjunctions(List<List<Posting>> lists)
  {  // on ordonne la liste de postings
     Collections.sort(lists, COMPARATOR);
     System.out.print(" Ordering posting list:");
     for(List<Posting> list: lists)
        System.out.print(" ("+list.size()+")");
     System.out.println();
     ...
  }
```

Exercice 12

```
  private List<Posting> processConjunction(List<Posting> list1, List<Posting> list2)
  {  ...
     while((it1.hasNext() || posting1!=null) && (it2.hasNext() || posting2!=null))
     {  ...
     }

     System.out.println(" Processing conjunction: ("+list1.size()+") AND
         ("+list2.size()+") >> ("+result.size()+")");
     return result;
  }
```

Exercice 13

Reprenons l'exemple de la première requête d'évaluation, donné dans le sujet. Ci-dessous, le code couleur indique les correspondances entre les termes composant la requête et l'ordre dans lequel ils sont traités :

```
Processing query "Arthroscopic treatment of cruciate ligament injuries"
  Normalizing: "arthroscopic"(83) "treatment"(2092) "of"(7667) "cruciate"(54)
"ligament"(99) "injuries"(256)
  Ordering posting list: (54) (83) (99) (256) (2092) (7667)
  Processing conjunction: (54) AND (83) >> (15)
  Processing conjunction: (15) AND (99) >> (14)
  Processing conjunction: (14) AND (256) >> (3)
  Processing conjunction: (3) AND (2092) >> (0)
Query processed, returned 0 postings, duration=1 ms
```

On voit qu'il y a seulement 3 documents contenant `arthroscopic`, `ligament` et `injuries`. Mais aucun d'entre eux ne contient `treatment`. Cependant, peut-être contiennent-ils des mots de la même famille, même s'ils ne sont pas exactement similaires. La racinisation (ou mieux, la lemmatisation) permettrait de tester cette hypothèse.

4 Racinisation

Exercice 14

```
  public String normalizeType(String string)
  {  String result = string;
     ...
```

```
4       // on renvoie null en cas de chaîne vide
5       if(result.isEmpty())
6          result = null;
7       // sinon on tente de raciniser
8       else if(stemmer!=null)
9          result = stemmer.stemType(result);
10
11      return result;
12    }
```

Exercice 15

On obtient les performances suivantes :

```
PRECISION  RECALL  F_MEASURE
1.00       0.14    0.25       Arthroscopic treatment of cruciate ligament injuries
0.00       0.00    0.00       Complications of arthroscopic interventions
0.00       0.00    0.00       Pathophysiology and prophylaxis of arthrofibrosis
0.00       0.00    0.00       HIV Epidemiology, Risk Assessment
0.00       0.00    0.00       Patient-controlled analgesia indications and limits
0.00       0.00    0.00       Priming with non-depolarizing muscle relaxants
0.83       0.62    0.71       Complications after laparoscopic cholecystectomy
0.00       0.00    0.00       Heparin induced thrombocytopenia, diagnosis and...
0.50       0.17    0.25       Diagnostic in Lyme disease
0.33       0.29    0.31       Treatment of acute myocardial infarction
1.00       0.11    0.19       Catheter ablation and cardiac mapping
1.00       0.06    0.12       Diagnostic approach in injuries of the shoulder
0.50       0.04    0.08       Differential diagnosis in infertility
0.00       0.00    0.00       Approach of the correction of deformities in orthopedics
0.39       0.41    0.40       Treatment of squamous cell carcinoma
1.00       0.07    0.12       Associated diseases with insulin dependent diabetes...
1.00       0.22    0.36       Therapy in chronic low back pain
0.45       0.41    0.43       Treatment of ventricular tachycardia
0.25       0.05    0.08       Indication for implantable cardioverter defibrillator
0.00       0.00    0.00       Diagnostic in acute and chronic myocarditis
0.75       0.20    0.32       Cause of dysphagia
1.00       0.14    0.25       Treatment of sensorineural hearing loss (SNHL)
1.00       0.12    0.22       Complications of surgical repair of aortic aneurysm
0.46       0.38    0.41       Treatment of psychosomatical patients
0.00       0.00    0.00       New approach in cruciate ligament surgery
---------------------------------------
0.46       0.14       0.18
```

On observe une légère amélioration, à la fois de la précision (+0,08 points) et du rappel (+0,04 points). La racinisation aide donc à la fois à diminuer le taux de documents non-pertinents renvoyés par le moteur (faux positifs), et le taux de documents pertinents qu'il manque (faux négatifs). Toutefois, la performance reste relativement faible.

Regardons la première requête d'évaluation en détail :

```
Processing query "Arthroscopic treatment of cruciate ligament injuries"
  Normalizing: "arthroscop"(100) "treatment"(2132) "of"(7667) "cruciat"(54)
"ligament"(144) "injuri"(465)
  Ordering posting list: (54) (100) (144) (465) (2132) (7667)
  Processing conjunction: (54) AND (100) >> (17)
  Processing conjunction: (17) AND (144) >> (16)
  Processing conjunction: (16) AND (465) >> (6)
  Processing conjunction: (6) AND (2132) >> (2)
  Processing conjunction: (2) AND (7667) >> (2)
Query processed, returned 2 postings, duration=2 ms
```

Le nombre de document restant après l'évaluation de chaque opérateur ET est supérieur à

ce qu'on observait précédemment. Cela montre bien que la racinisation rend implicitement plus flexible la comparaison entre mots : deux mots qui auraient été considérés comme différents sans elle peuvent avoir la même racine, et donc être considérés comme similaire lorsqu'on les racinise.