Information Retrieval
Programming Assignment 3 Report

資管四 B98705046 遲翔

- **執行環境：**

  ruby 1.9.3（Windows 7）

- **執行方式：**

  在 Windows console 端執行指令 *ruby pa3.rb*（須安裝 ruby 1.9.3）。

- **執行時間（筆記型電腦，i5 處理器、4GB 記憶體）：**

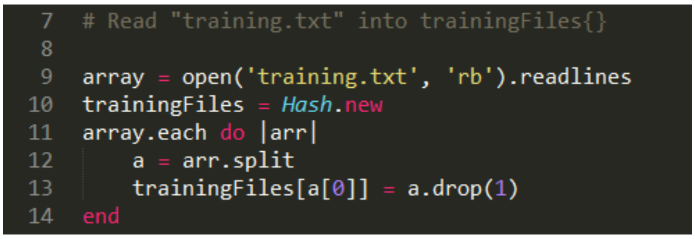
	第一次	第二次	第三次
Feature Selection	8’30”	11’20”	10’00”
Training & Testing	5’30”	4’30”	4’22”
總時間	14’00”	15’50”	14’22”

平均執行時間：14’44”

- **程式執行步驟：**

1. “training.txt”

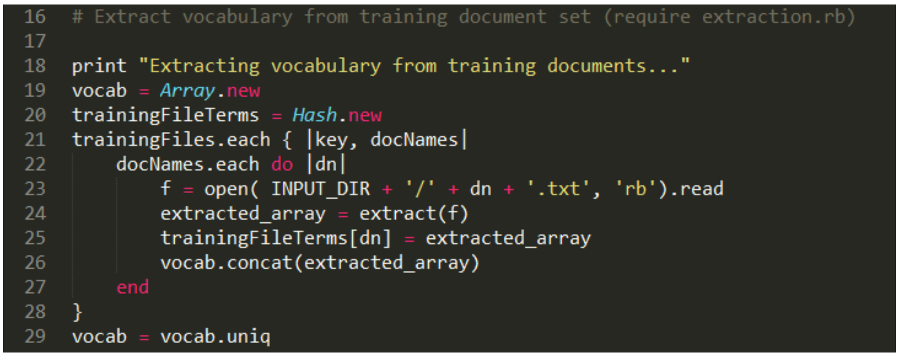
首先讀取老師提供的”training.txt”檔案，把每個 class 和檔案的關係存成{“1” => [11, 19, 29,…], “2” => [1, 2, 3,…], …} 的 hash 形式，方便之後讀取。



```ruby
# Read "training.txt" into trainingFiles{}

array = open('training.txt', 'rb').readlines
trainingFiles = Hash.new
array.each do |arr|
    a = arr.split
    trainingFiles[a[0]] = a.drop(1)
end
```

2. Extract vocabulary

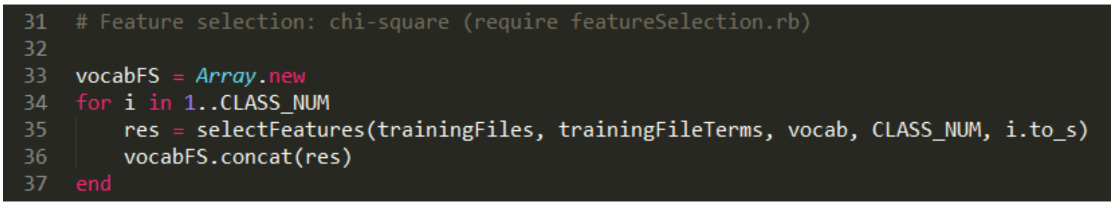
把 training document 的內容讀取出來，並且用之前寫的 extract 函式（寫在”extraction.rb”檔案中）做關鍵字的萃取，刪除多餘的 stopword 以及做 stemming，把初步的關鍵詞彙存在 *vocab* 這個變數中。



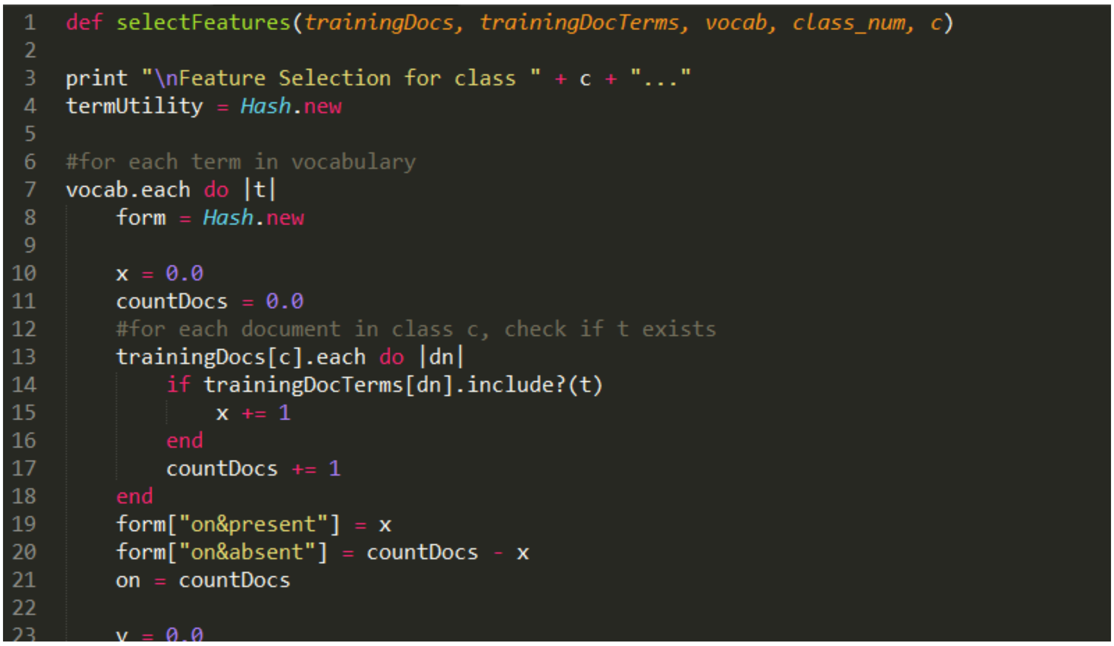
```ruby
# Extract vocabulary from training document set (require extraction.rb)

print "Extracting vocabulary from training documents..."
vocab = Array.new
trainingFileTerms = Hash.new
trainingFiles.each { |key, docNames|
    docNames.each do |dn|
        f = open( INPUT_DIR + '/' + dn + '.txt', 'rb').read
        extracted_array = extract(f)
        trainingFileTerms[dn] = extracted_array
        vocab.concat(extracted_array)
    end
}
vocab = vocab.uniq
```

3. Feature selection

呼叫 selectFeatures 函式（寫在”featureSelection.rb”檔案中）。這邊選擇使用投影片中的「χ^2 feature selection」方法，針對每一個 class，對前面 *vocab* 變數中的每一個 term 計算出現於該 class 的次數、出現於非該 class 中的次數，再計算 χ^2 統計值，並且在每個 class 中選出前 38 或 39 個 χ^2 統計值最高的 term（比較重要、比較有判別性的 term），放入最後真正會使用的關鍵字集合中。最後會湊滿 500 個關鍵字，存在 *vocabFS* 這個變數中。



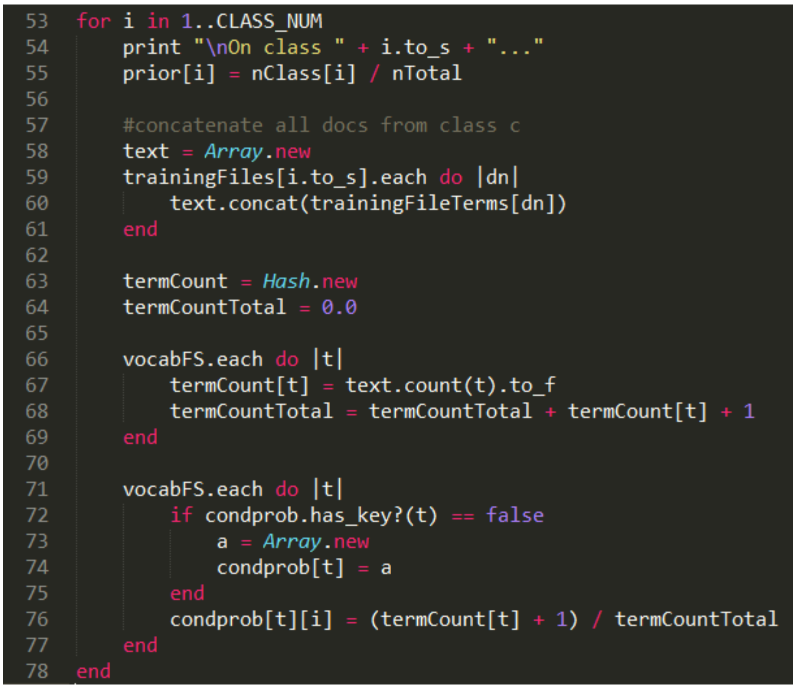
```ruby
# Feature selection: chi-square (require featureSelection.rb)

vocabFS = Array.new
for i in 1..CLASS_NUM
    res = selectFeatures(trainingFiles, trainingFileTerms, vocab, CLASS_NUM, i.to_s)
    vocabFS.concat(res)
end
```



```ruby
def selectFeatures(trainingDocs, trainingDocTerms, vocab, class_num, c)

print "\nFeature Selection for class " + c + "..."
termUtility = Hash.new

#for each term in vocabulary
vocab.each do |t|
    form = Hash.new

    x = 0.0
    countDocs = 0.0
    #for each document in class c, check if t exists
    trainingDocs[c].each do |dn|
        if trainingDocTerms[dn].include?(t)
            x += 1
        end
        countDocs += 1
    end
    form["on&present"] = x
    form["on&absent"] = countDocs - x
    on = countDocs

    y = 0.0
```

4. Training

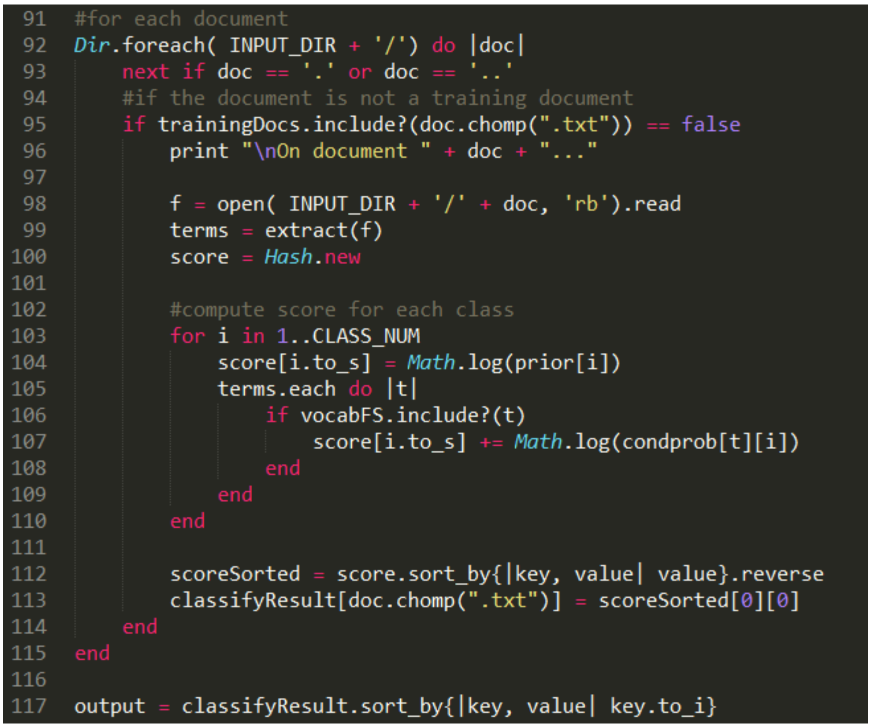
確定關鍵字集合之後就進入 training 階段。針對每一個 class，計算每一個 term 在該 class 中出現的條件機率，也就是說會有 C * M = 13 * 500 = 6500 個條件機率值，存在 condprob 這個變數中，以{“apple” => [0.01, 0.02,…], “tree” => [0.1, 0.032,…], …} 這樣的 hash 形式儲存。



```ruby
for i in 1..CLASS_NUM
    print "\nOn class " + i.to_s + "..."
    prior[i] = nClass[i] / nTotal

    #concatenate all docs from class c
    text = Array.new
    trainingFiles[i.to_s].each do |dn|
        text.concat(trainingFileTerms[dn])
    end

    termCount = Hash.new
    termCountTotal = 0.0

    vocabFS.each do |t|
        termCount[t] = text.count(t).to_f
        termCountTotal = termCountTotal + termCount[t] + 1
    end

    vocabFS.each do |t|
        if condprob.has_key?(t) == false
            a = Array.new
            condprob[t] = a
        end
        condprob[t][i] = (termCount[t] + 1) / termCountTotal
    end
end
```

5. Testing

Training 結束後進入 testing phase。將 training documents 以外的其他文件一個一個打開，利用前面計算出來的條件機率，對每份文件計算其在 13 個 class 的分數，選出分數最高的 class，作為這份文件所屬的 class。



```ruby
#for each document
Dir.foreach( INPUT_DIR + '/') do |doc|
    next if doc == '.' or doc == '..'
    #if the document is not a training document
    if trainingDocs.include?(doc.chomp(".txt")) == false
        print "\nOn document " + doc + "..."

        f = open( INPUT_DIR + '/' + doc, 'rb').read
        terms = extract(f)
        score = Hash.new

        #compute score for each class
        for i in 1..CLASS_NUM
            score[i.to_s] = Math.log(prior[i])
            terms.each do |t|
                if vocabFS.include?(t)
                    score[i.to_s] += Math.log(condprob[t][i])
                end
            end
        end

        scoreSorted = score.sort_by{|key, value| value}.reverse
        classifyResult[doc.chomp(".txt")] = scoreSorted[0][0]
    end
end

output = classifyResult.sort_by{|key, value| key.to_i}
```