Principles of Java Language with Applications, PIC20a
E. Ryu
Fall 2018

UCLA

Homework 6
Due 5pm, Wednesday, December 5, 2018

Write your code into the `main` function of `class YelpAnalysis`. Submit `YelpAnalysis.java`. Submit `Business.java` if you use it. You are free to modify `Business.java`. Put everything in the package `hw6`, including the `class YelpAnalysis`.

Problem 1: (Big Yelp data analysis)

Go to
`https://www.yelp.com/dataset_challenge/dataset`
and agree to the terms and conditions. However, do not download their data set. Use the processed data set we provide.

`yelpDatasetParsed_full.txt` contains Yelp reviews from 85,538 out of their 66,000,000 business listings. The file has the following format:

```
{businessID, businessName, businessAddress, reviews}
{businessID, businessName, businessAddress, reviews}
...
```

The reviews consists of lowercase English letters and spaces without any punctuation or non-English characters. While programming and debugging, you may want to use the smaller file `yelpDatasetParsed_short.txt`, which only contains the first 100 businesses.

The goal of this assignment is to process this data set and extract meaningful words from the Yelp reviews that represent the businesses.

1. Read the file, and create a `Business Object` for each business. Each `Business` should contain its reviews as a `String`. You may use the `class Business` provided in `Business.java`.

We will determine whether or not a word meaningfully represents a business with the number of times it appears in the reviews. However words like 'a', 'that', 'is', or 'and' are most frequent, and these words clearly do not say much about the business.

Therefore, we use the *term frequency-inverse document frequency* (tf-idf) score:

$$\text{tf-idf}(w, D) = \frac{\text{number of times word } w \text{ appears in document } D}{\text{number of documents in the entire corpus that contain word } w}$$

The tf-idf score is high if a rare word appears many times in a certain document. The numerator is the "term frequency", and the denominator is the "document frequency".

2. For each word, count the number of documents the word appears in.

3. For the top 10 `Business`es with the most characters in their reviews, output (to the command line) the top 30 words with the highest tf-idf scores.

When you do so, you'll notice that some words with high tf-idf scores are so rare that they appear in 1 or 2 documents. Some of these words are misspellings or slangs that only make sense to locals. Filter these out.

4. If a word appears in less than 5 documents, assign a tf-idf score of 0.

Poorly written code will take too long to process the full data set. Perform some optimization.

5. Optimize your code so that it runs on the full data set within 10 minutes.

Your code could look something like

```java
public static void main(String[] args) {
  ...
  Map<String,Integer> corpusDFCount = ???;
  List<Business> businessList = ???;
  while (true) {
    Business b = readBusiness(???);
    if (b==null) //end of file and processed all businesses
      break;
    businessList.add(b);
  }

  for (Business b : businessList)
    addDocumentCount(corpusDFCount,b);

  //sort by character count
  Collections.sort(businessList, ???);

  //for the top 10 businesses with most review characters
  for (int i=0; i<10; i++) {
    Map<String,Double> tfidfScoreMap =
           getTfidfScore(corpusDFCount, businessQueue.remove(), 5);
    //Entry is a static nested interface of class Map
    List<Map.Entry<String,Double>> tfidfScoreList
                            = new ArrayList<>(tfidfScoreMap.entrySet());
    sortByTfidf(tfidfScoreList);
    System.out.println(businessList.get(i));
    printTopWords(tfidfScoreList, 30);
  }
}
```

This code can be further optimized using a `PriorityQueue`

```java
public static void main(String[] args) {
  ...
  Map<String,Integer> corpusDFCount = ???;
  PriorityQueue<Business> businessQueue = ???;
  while (true) {
    Business b = readBusiness(???);
    if (b==null) //end of file and processed all businesses
      break;
    addDocumentCount(corpusDFCount,b);
    businessQueue.add(b);
    if (businessQueue.size()>10)
      businessQueue.remove();
  }


  //for the top 10 businesses with most review characters
  for (int i=0; i<10; i++) {
    Business currB = businessQueue.remove();
    Map<String,Double> tfidfScoreMap =
           getTfidfScore(corpusDFCount, currB, 5);
    //Entry is a static nested interface of class Map
    List<Map.Entry<String,Double>> tfidfScoreList
                        = new ArrayList<>(tfidfScoreMap.entrySet());
    sortByTfidf(tfidfScoreList);
    System.out.println(currB);
    printTopWords(tfidfScoreList, 30);
  }
}
```

(This code is just a suggestion provided to clarify the instructions.)

Your output should look something like

```
-------------------------------------------------------------------------------
Business ID: 60454
Business Name: Bacchanal Buffet
Business Address: Caesars Palace Las Vegas Hotel And Casino 3570 Las Vegas Boul
evard South The Strip Las Vegas NV 89109
Character Count: 3780749
(bacchanal,10.75) (bacchanals,2.73) (buffet,1.48) (baccahanal,1.17) (bachannal,
1.07) (buffets,1.06) (bacchanel,1) (bachanal,0.92) (ginseng,0.91) (alvaro,0.89)
(baccanal,0.89) (carving,0.84) (legsclaws,0.83) (macarons,0.74) (oysters,0.64)
(platinumdiamond,0.6) (virkelig,0.6) (wicked,0.57) (bacchanalian,0.56) (crab,0.
54) (dionysus,0.5) (fastpass,0.5) (wicket,0.5) (gelato,0.48) (legs,0.46) (bucch
anal,0.45) (the,0.45) (maccaroons,0.44) (jonah,0.44) (vomitorium,0.44)
```

```
---------------------------------------------------------------------------------
Business ID: 20187
Business Name:  Mon Ami Gabi
Business Address:  3655 Las Vegas Blvd S The Strip Las Vegas NV 89109
Character Count: 3481415
...
```

Hint. When optimizing your code, consider the following advice.

- Instead of directly using `java.io.FileInputStream`, `java.io.BufferedInputStream`, or `java.io.Reader`, use `java.io.BufferedReader` with `java.io.FileReader`.
- You can read a line of the dataset with `readLine()` of `BufferedReader`. You can separate a `String` about commas with `split(...)` of `String`.
- `Strings` are immutable, and using `+` to concatenate `Strings` is inefficient. When you're building up a large string, use `java.lang.StringBuffer` or `java.lang.StringBuilder`.
- If you add additional fields to `class Business`, make sure your additional fields aren't too large. In particular, you're doing something wrong if each `Business` owns (as a field) a `Collection` or a `Map`.

Remark. As a point of reference, my code runs in 90 seconds. Almost all of this time is spent in the `while` loop reading the `Business`es and accumulating the document count.

Remark. The suggested code for this assignment is not very object-oriented, and that's fine. Object-oriented programming is a tool, and a tool should only be used when it fits the task.

Remark. In our data analysis, we entirely ignore the grammar the reviews were written in. *Natural language processing* is a field of artificial intelligence that studies how to process human language with a computer. Techniques from natural language processing would use the grammatical information and produce better results.

Acknowledgements. We thank Yelp, Inc. for generously providing their data. The original data set can be found at
`https://www.yelp.com/dataset_challenge`
Our use of the data set in this assignment is academic in nature and thus is allowed by Yelp, Inc. This data set was processed by Swati Arora into the provided format. You may not redistribute this processed data set.