Trier University
Department IV – Computer Sciences
Prof. Dr.-Ing. Ralf Schenkel (H 539, schenkel@uni-trier.de)

Trier, June 2, 2021

Assignment 6 of the course Big Data Analytics

Summer semester 2021

Deadline 10:15h on Wednesday, June 23, 2021

Task 1: Shingling $10 + 15 + 20$ points

(a) Write a Java class that implements the `Shingler` interface provided in Moodle. This interfaces has only one method

```
public Set<String> computeShingles(String text, int k)
```

that takes a String as input and returns all distinct k-shingles of that text in a Java `Set`. Your implementation of the interface should compute **word-level** shingles. Normalize the input text by removing punctuation and converting it to lowercase, similar to the **WordCount** example. Represent a shingle as a `String` object. Write a `main` method that prints the generated shingles for the following two input texts

- Data Science is an upcoming field that combines methods from computer science, mathematics and statistics.
- Data Science is an emerging field that brings together methods from statistics, mathematics, and computing.

Hand in your code and the output of your code for $k = 3$.

(b) Write a Java class with a `main` method that prints, for a given `String` array `documents`, the shingle-based Jaccard similarity of all pairs of Strings `documents[i]` and `documents[j]` with $i < j$ using your implementation from Task (a) to compute the shingles. The array `documents` to use as input is given in the class **Task1bSkeleton** provided in Moodle. The output of your program should be of the form
`1-Shingles:`
`sim(0,1)=0.1`
(for the first and second document from the array), but of course include all pairs. Hand in your code and the output of your program for 1-shingles, 3-shingles, and 5-shingles in one file.

(c) Implement MinHashing, following the algorithm on slide 4-76 and using your shingling implementation from Task (a). The number of hash functions to use, i.e., the size of the signature vector, should be a configurable parameter.
The class **GenericHashFunction** provided in Moodle implements a generic hash function in its method `hash` that takes a `String` object (i.e., a shingle) as a parameter. Using its static method `generate`, it is possible to create an array of k objects representing k random hash functions. Note that the number of distinct shingles in the document collection is a parameter to this method and thus needs to be computed beforehand. Using the provided implementation of random hash functions, you can directly apply them to shingles instead of their index (as it is done on slide 4-76).
The `main` method should again consider the `String` array from **Task1bSkeleton** and compute MinHash signatures for the documents included there.

Your program should, for each input document, output the MinHash signature. In addition, it should output for every pair of documents (as in Task (b)) the similarity of their signatures, as shown on slide 4-74.
Hand in your code and the output of your program for 1, 5, and 10 hash functions in one file, all using 5-shingles.

Task 2: Text Similarity in MapReduce (45) points

Write a MapReduce application that, given a collection of documents, uses MinHashing and LSH to identify pairs of documents that should be checked for their similarity. In more detail,

- Each input line that the Mapper consumes has the format
  `docid: text`
  The Mapper should first extract the docid and then proceed with processing the remainder of the line.
- The Mapper should, for each input line, generate the corresponding k-shingles and the MinHash signature with m hash functions. Both parameters should be set in the source code.
- **Creating the random hash function requires the number of distinct shingles, which cannot be computed in a single Mapper (since the collection is distributed). Assume in your program that there are 90,000 distinct shingles.**
- Given a MinHash signature with m entries, the mapper should emit subsequences of length p as key and the docid as the value. For example, if the signature is `[1,7,2,1]` and we are using subsequences of length $p = 2$, the Mapper should emit the keys `1:1,7`,`2:7,2`, `3:2,1`. The additional prefix encodes the position in the signature where the subsequence starts (see slide 4-83).
- The reducer checks if it received more than one value for a key. In this case, the reducer should output the key and the complete list of values received **if the list of values includes docid 1282**. We will not implement the actual comparison of the documents.

Hand in your code and the output of your program for 5-shingles, $m = 10$ hash functions, and values of $p = 1$, $p = 3$ **and** $p = 5$ (so three output files, with proper naming or other indication which parameters were used), when run on the input file `corpus_with_line_numbers.txt` provided in Moodle (and already used in Assignment 3). **This is an optional task.**

These tasks will be discussed in the meeting on June 30, 2021.

General remarks:

- The tutorial group takes place on Wednesdays in the regular Zoom meeting at 10:15, see StudIP (slides with general information on the lecture) for the registration link.
- To be admitted to the final exam, you need to acquire at least 50% of the points in the assignments.
- It is preferred to submit in groups of size 2 (but not larger); in that case, only one submission is sufficient for the whole group. Groups must be chosen in Moodle (see link on the course page in Moodle). Write the names of all group members on your solutions.

- Solutions must be handed in before the deadline in Moodle (`https://moodle.uni-trier.de/`, course `BDA-21`) as as a PDF or, if submitting multiple files, as an archive (.zip or comparable). Submissions that arrive after the deadline will not be considered.
- Graded versions of your submissions will be returned in Moodle until the following tutorial.
- Announcements regarding the lecture and the tutorial group will be done in the area of the lecture in StudIP.