Heidelberg University
Winter semester 2024/25

Group Artificial Intelligence for Programming (AIP)
Prof. Dr. Artur Andrzejak, Kim Tuyen Le

Problem Set 6 for lecture Mining Massive Datasets

Due December 09, 2024, 23:59 CET

Exercise 1 (1 point)

The Jaccard similarity can be applied to sets of elements. Sometimes, documents (or other objects) may be represented as multi-sets/bags rather than sets. In a multi-set, an element can be a member more than once, whereas a set can only hold each element at most once. Try to define a similarity metric for multi-sets. This metric should take exactly the same values as Jaccard similarity in the special case where both multi-sets are in fact sets.

Exercise 2 (6 points)

Recall the concept of shingling documents which makes it possible to represent text documents as sets. Develop a scalable implementation in Spark (using DataFrames and not RDDs) which computes for a collection of documents the corresponding sets of shingles.

Here, we mean shingles of characters (as opposed to e.g., shingles of words). Be careful to take into consideration both line breaks and the hyphenation at the end of lines. I.e., in the following example, with $k = 9$, the shingles should include: "uished bu", "is possib", "hen he st", "mpossible", "robably w". Other special cases needed for proper shingling may be ignored.

> When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.

Run your implementation on each of 10 different documents each with size of at least 10 KByte. In order for the solutions to be comparable, one of the documents should be the *Grundgesetz für die Bundesrepublik Deutschland* in the version 64 from 03.04.2019[1]. Output the number of distinct shingles per document for a run with $k = 5$ and $k = 9$, respectively. Indicate in your solution which results are for the document *Grundgesetz.*

Exercise 3 (3 points)

Figure 1 shows a table (or matrix) representing four sets S_1, S_2, S_3 and S_4 (subsets of $\{0, 1, 2, 3, 4, 5\}$).

a) Compute the MinHash signature for each set using the following three hash functions:
$h_1(x) = 2x + 1 \mod 6$
$h_2(x) = 3x + 2 \mod 6$
$h_3(x) = 5x + 2 \mod 6$

b) Which of these hash functions are true permutations? What collisions do occur in the other hash functions? Name the corresponding inputs and outputs.

[1] You can find it at https://gg.docpatch.org/. To get correct German umlaut signs, you should download RTF version and convert it to plain text.

c) Compare the similarity of the MinHash signatures against the corresponding Jaccard similarities, for each of the $\binom{4}{2} = 6$ pairs of columns.

Element	S_1	S_2	S_3	S_4
0	0	1	0	1
1	0	1	0	0
2	1	0	0	1
3	0	0	1	0
4	0	0	1	1
5	1	0	0	0

Figure 1: An example matrix

Exercise 4 (5 points)

Recall the concepts of Shingling and MinHash signatures to perform the following tasks. Submit as your solution your source code, results, and logs of runs. You do *not* need to take care of the scalability of your code, e.g., assume that all input/output data fit into RAM.

a) Implement a routine in Python to compute a representation of a string of decimal digits (0...9) as a set of k-shingles. The input of your routine is a string of digits and k. The output is an ordered list of positions of 1's in a (virtual) Boolean representation of a set of k-shingles as outlined in Lecture 7 (see slide "From Sets to Boolean Matrices"). The position of a k-shingle x (of digits) in the Boolean vector is x interpreted as an integer. For example, shingles "0...00" and "0...2024" would map to (decimal) positions 0 and 2024, respectively. Moreover, for a string "1234567" and $k = 4$ your routine should output the list [1234, 2345, 3456, 4567].

Hint: You can use Python's data structure `set()` (or as alternative `dict()`) to need just one pass through the input string plus outputting the positions in an ordered fashion.

b) Run your implementation from a) on the first 10000 digits of π after comma using $k = 12$. Save the output list as a text file with one position (list element) per line, and submit it as a part of your solution.

Hint: There are multiple sources to get digits of π. You can also use Python libraries such as SymPy and mpmath.

b) Implement (in Python) the algorithm for MinHash signatures as described in the slides "Implementation /*" of Lecture 7. We simplify here and assume only one column C representing one document/string. Thus, your algorithm shall use as input a single list of positions of 1s in a (virtual) Boolean vector described in a).

Run your implementation on the list of positions obtained in b) using 5 hash functions, specified as follows:

- First hash function uses $a = 37$, $b = 126$ and $p = 10^{15} + 223$.
- Generate the remaining 4 functions by sampling a and b as uniform random integers from the range $U = [0, 10^{12}]$ and using the following prime numbers p, respectively: $p = 10^{15} + i$ with $i \in [37, 91, 159, 187]$.