# of Data Mining

Homework 2

Due: 7/11/2021, 23:59

Instructions

You must hand in the homework electronically and before the due date and time.

This homework has to be done by each **person individually**.

Handing in: You must hand in the homework by the due date and time by an email to Andrea (`mastropietro.1652886@studenti.uniroma1.it`) that will contain as attachment **(not links to some file-uploading server!)** a .zip file with your answers. The filename of the attachment should be
`DM_Homework_1__StudentID_StudentName_StudentLastname.zip`;
for example:
`DM_Homework_1__1235711_Robert_Anthony_De_Niro.zip`.
The email subject should be
`[Data Mining] Homework_1 StudentID StudentName StudentLastname`;
For example:
`[Data Mining] Homework_1 1235711 Robert Anthony De Niro`.
After you submit, you will receive an acknowledgement email that your project has been received and at what date and time. If you have not received an acknowledgement email within 2 days after you submit then contact Andrea.

The solutions for the theoretical exercises must contain your answers either typed up or hand written clearly and scanned.

For information about collaboration, and about being late check the web page.

Problem 1. For this question you have to implement a search engine for the kijiji announcement that you downloaded for Homework 1, Problem 6. It has several parts that you need to implement. For the linguistic analysis you can use the NLTK Python library.

1. Firt you have to preprocess them. For each announcement use the textual fields (e.g., *title*, *full description*). You can do whatever preprocessing you think is essential (e.g., stopword removal, normalization, stemming).

2. The next step is to build a search-engine index. First, you need to build an inverted index, and store it in a file. Build an index that allows to perform proximity queries using the cosine-similarity measure. Then build also a query-processing part, which, given some terms it will bring the most related announcements.

Hand in the code, along with some examples of queries and screenshots of the results. Try short queries, such as a few terms, as well as long queries, which can be other announcements.

Problem 2. Here we are asking to implement nearest-neighbor search for text documents. You have to implement shingling, minwise hashing, and locality-sensitive hashing. We split it into several parts:

1. Implement a class that, given a document, creates its set of character shingles of some length k. Then represent the document as the set of the hashes of the shingles, for some hash function.

2. Implement a class, that given a collection of sets of objects (e.g., strings, or numbers), creates a minwise hashing based signature for each set.

3. Implement a class that implements the locally sensitive hashing (LSH) technique, so that, given a collection of minwise hash signatures of a set of documents, it finds the all the documents pairs that are near each other.

To test the LSH algorithm, also implement a class that given the shingles of each of the documents, finds the nearest neighbors by comparing all the shingle sets with each other.

We will apply the algorithm to solve the problem that companies such as kijiji face when companies or individuals post many copies of the same announcement—usully they want to block announcements that are near duplicates, otherwise users keep putting up announcements for the same job to show up in the top. We will work on the announcements for apartments of Problem 1.

We want to find announcements that are near duplicates. We will say that two announcements are near duplicates if the Jaccard coefficient of their shingle sets is at least 80%. We will use shingles of length 10 characters. Find values for r and b (see Section 3.4 in the book) that can give us the desired behavior. To plot the graph that gives the probability as a function of the similarity for different values of r and b you can use, for example, Wolfram Alpha.

To apply the algorithm you have the following tasks:

1. Find the near-duplicates among all the announcements of Homework 1 (Problem 6), using LSH. Use the *full description*, if you have it, otherwise the *short description*.

2. Find the near-duplicates among all the announcements of Homework 1 by comparing them with each other.

3. Report the number of duplicates found in both cases, and the size of the intersection.

4. Report the time required to compute the near duplicates in either case.

You will need a way to create a family of hash functions. One way is to use a hash function and a code similar to the following.

```
# Implement a family of hash functions. It hashes strings and takes an
# integer to define the member of the family.
# Return a hash function parametrized by i
import hashlib
def hashFamily(i):
  resultSize = 8        # how many bytes we want back
  maxLen = 20           # how long can our i be (in decimal)
  salt = str(i).zfill(maxLen)[-maxLen:]
  def hashMember(x):
    return hashlib.sha1(x + salt).digest()[-resultSize:]
  return hashMember
```

Note that this code is an overkill because we use a cryptographic hash function, which can be very slow, even though it is not needed to be as secure. However, for the necessities of the homework we will use it to avoid having to install some external hash library.

Problem 3. Implement Problem 2 using Apache Spark.