# Data Mining

## Homework 5

**Due:** 4/6/2013, 23:59.

You must hand in the homeworks electronically and before the due date and time. Check the web page for instructions about collaboration, about being late, and about handing in the homework.

**Hand in:**

- The `.py` files with the source code.
- Instruction to execute them.
- The output files generated, and output at the screen.

**Problem 1.** Here we are asking to implement nearest-neighbor search for text documents. You have to implement shingling, minhashing, and locally 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 minhashing based signature for each set.
3. Implement a class that implements the locally sensitive hashing (LSH) technique, so that, given a collection of minhash 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 it 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. We will work on the announcements for job positions of Homework 1, Problem 5, but here we need the full text. So for every announcement you need to visit its page and retrieve the full text. Also, note that there has been a slight change in the html code created by kijiji so you may also need to change the code that retrieves the list of the announcements. **Note, that because you will make a large number of visits to kijiji, you need to have delays between them, or you might be blocked.**

After you retrieve the full text of each announcement find all the near duplicates using the code you have implemented.

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. Download the full text of all the announcements from `http://www.kijiji.it` that are in Rome and about *Informatica/Grafica/Web* with a *Contratto* (as in Problem 5 in Homework 1). Save the announacements in a TSV file, one line per announcement.
2. Find the near-duplicates among all the announcements using LSH.

3. Find the near-duplicates among all the announcements by comparing them with each other.
4. Report the number of duplicates found in both cases, and the size of the intersection.
5. 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.