# COMPSCI 753

## Algorithms for Massive Data

Semester 2, 2020

## Assignment 1: Locality-sensitive Hashing

Ninh Pham

## Submission:

Please submit a single pdf file & the source code on CANVAS by **11:59pm, Sunday 23 August 2020**. The answer file must contain your studentID, UPI and name.

## Penalty Dates:

The assignment will not be accepted after the last penalty date unless there are special circumstances (e.g., sickness with certificate). Penalties will be calculated as follows as a percentage of the mark for the assignment.

- **By 11:59pm, Sunday 23 August 2020 – No penalty**
- **By 11:59pm, Monday 24 August 2020 – 25% penalty**
- **By 11:59pm, Tuesday 25 August 2020 – 50% penalty**

# 1 Assignment problem (50 pts)

The assignment aims at investigating MinHash and Locality-sensitive Hashing (LSH) framework on real-world data sets. In class, we have seen how to construct signature matrices using random permutations. However, in practice, random permutation on very large matrix is prohibitive. Section 3.3.5 of your textbook (Chapter 3, *Mining of Massive Datasets*[1] by J. Leskovec, A. Rajaraman, J. Ullman) introduces a simple but fast method to "simulate" this randomness using different hash functions. We encourage you to read through that section before attempting the assignment.

In the assignment, you write a program[2] to compute *all pairs similarity* on the "bag of words" data set from the UCI Repository[3] using the *Jaccard* similarity. This problem is the core component for detecting plagiarism and finding similar documents in information retrieval.

The bag of words data set contains 5 text datasets which share the same pre-processing procedure. That is, after tokenization and removal of stopwords, the vocabulary of unique words was truncated by only keeping important words that occurred more than 10 times for large data sets. For small data sets, there was not truncation.

It has the format: $docID$ $wordID$ $count$, where $docID$ is the document ID, $wordID$ is the word ID in the vocabulary, and *count* is the word frequency. Since the Jaccard similarity does not take into account the word frequency, we simply ignore this information in the assignment. This means that we consider $count = 1$ for each pair ($docID$, $wordID$). We consider a document as a set and each word as a set element, and make use the Jaccard similarity.

We only make use the ***KOS blog entries*** data set for this assignment since we still need to run bruteforce algorithm to measure the accuracy of MinHash and LSH. However, students are encouraged to try on larger data sets, such as *NYTimes news article* and *PubMed abstracts*. Note that the data set is very sparse, so you might think of the relevant data structures for fast processing.

The assignment tasks and its point are as follows.

1. **Execute bruteforce computation (10 pts):** Compute all pairs similarity with Jaccard and save the result into file (since you have to use the bruteforce result for the next tasks). You need to report:

   (a) **The running time of your bruteforce algorithm (5 pts).**

[1] `http://www.mmds.org/`

[2] no restriction on programming languages used but preferred Python.

[3] `https://archive.ics.uci.edu/ml/datasets/Bag+of+Words`

(b) **The average Jaccard similarity of all pairs except identical pairs i.e. $J(d_i, d_j)$ where $i \neq j$ (5 pts).**

2. **Compute the MinHash signatures for all documents (10 pts):** Compute the MinHash signatures (number of hash functions $d = 10$) for all documents. You need to report:

   (a) **The running time of this step (10 pts)**.

3. **Measure the accuracy of MinHash estimators (10 pts):** Compute all pairs similarity estimators based on MinHash. Repeat the procedure 2) and 3) with the number of hash functions $d$ ranging from $\{10, 20, \ldots, 100\}$ and plot the *mean absolute error (MAE)* of MinHash estimators on different values of $d$.

   $MAE = \sum_{i,j=1,i\neq j}^{n} |J(d_i, d_j) - \hat{J}(d_i, d_j)|/(n^2 - n)$ where $J(d_i, d_j)$ is the actual Jaccard similarity between $d_i$ and $d_j$ and $\hat{J}(d_i, d_j)$ is their MinHash-based estimator. You need to report:

   (a) **The running time of estimating all pairs similarity based on MinHash with different values of $d$ (5 pts).**

   (b) **The figure of MAEs with different values of $d$ on $x$-axis and MAE values on $y$-axis (5 pts).**

4. **Exploit LSH (20 pts):** Implement LSH framework to solve the subproblem: "Finding all pairs similar documents with Jaccard $\geq 0.6$". In particular, using $d = 100$ hash functions, you need to explain:

   (a) **How to tune the parameter $b$ (number of bands) and $r$ (number of rows in one band) so that we achieve the false negatives of 60%-similar pairs at most 10% (5 pts)**.

   (b) **The space usage affected by these parameters (5 pts).**

   Given your chosen setting, **from your experiment**, you need to report

   (a) **The false candidate ratio (5 pts).**

   $$\frac{\texttt{the number of candidate pairs with exact Jaccard} < 0.6}{\texttt{number of candidate pairs}}$$

   (b) **The probability that a dissimilar pair with Jaccard $\leq 0.3$ is a candidate pair (5 pts).**

   $$\frac{\texttt{the number of candidate pairs with exact Jaccard} \leq 0.3}{\texttt{number of pairs with exact Jaccard} \leq 0.3}$$

# 2 What to submit?

An `answer.pdf` file reports the requested values and explanation of each task.

A `source code` file contains detailed comments.

*Note: When taking the screenshots, make sure that you do not reveal any additional content you do not wish to share with us ;-).*