COMPSCI 753: Algorithms for Massive Data
Assignment 1: Locality-sensitive Hashing (Worth 5 Pts in Total)
Due date: 23:59 14 August 2022

Learning Objectives:
The goal of this assignment is to investigate Locality-sensitive Hashing (LSH) framework for near neighbor search on real-world datasets. We have learned how to construct hash tables using multiple hash functions in LSH family for near neighbor retrieval in sublinear time during weeks 2-3. By accomplishing this assignment, you will be familiar with the following concepts:

- Hash Table Construction
- Hash Table Lookup
- Search Quality Evaluation

General Instruction:
This core component in this assignment is to construct a document retrieval system upon the LSH framework. This assignment consists of three parts. Please write a **python program** to complete the following components:

- Part I: Construct LSH Hash Tables for All News Articles
- Part II: Perfor
- Part III: Invest rch Quality in F1 score.

Datasets:
Let's consider two classes of BBC news articles rtainment news. In `bitvector_all.csv` and `bitvector_query.c` is a tab-separated line with three columns: `<news_id\tword_features\tnews class>`, where `news_id` is a unique string ID, `word_features` is a sequence of tab-separated binary values. Each entry in the `word_features` refers to the occurrence of a token. You can find their original new articles in `text_all.csv` and `text_query.csv`, accordingly in `bbc.zip` on Canvas.

Submission:
Please submit a single **report** (`.pdf`) and the **source code with detailed comments**.(`.py or .ipynb or .html`) on Canvas by **23:59, Sunday 14 August 2022**. 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 medical certificate, family/personal emergencies). Penalties will be calculated as follows as a percentage of the mark for the assignment.

- By 23:59, Sunday 14 August 2022 (No penalty)
- By 23:59, Monday 15 August 2022 (25% penalty)
- By 23:59, Tuesday 16 August 2022 (50% penalty)

Part I: Construct LSH Hash Tables for All News Articles [40 pts]

(a) Load `bitvector_all.csv` and `bitvector_query.csv`. Construct a feature vector for each news article in the dataset. Please report the number of articles, and the number of features (n) for these two sets of data. **[5 pts]**

(b) Construct a family of MinHash functions in the LSH family by taking a prime $p \geq n$ and for $0 < a < p$, $0 \leq b < p$ with the number of tables (l=10) and a tunable choice of hash size (k). Please report the family of MinHash functions you have generated with l=10 and k=2. **[15 pts]**

(c) Construct LSH hash tables using your hash functions with the number of tables (l=10) and bucket size of your choice (m). Please report the collision distribution of the l hash tables with all documents hashed into m buckets using heatmap plot, where x-axis is m, y-axis is l=10, and the values at (m,l) refers to the number of colliding articles). **[20 pts]**

Part II: Nearest Neighbor Search [35 pts]

(a) Query the LSH tables and return the top-10 articles that have the highest Jaccard similarities as the answer. For each query document **q** in our queries dataset Q, firstly, find the set of articles $\boldsymbol{D_q}$ that collide with **q** in at least one hash table. Compute Jaccard similarity between **q** and each article in $\boldsymbol{D_q}$. Please report the list of top-10 articles with highest Jaccard similarity in descending order for each query **q** (i.e., four lists in total). The article with the highest Jaccard similarity is ranked id> <Jaccard_sim> <class_label> fo

(b) Compute Jaccard . Please report the list of top-10 articles with highest Jaccard similarity in de h query **q** (i.e., four lists in total). **[10 pts]**

(c) Compare the query time in Part II(a) and Part II(b ds and comment on their differences if any. **[5 pts]**

Part III: Search Quality Evaluation [25 pts]

(a) Investigate the impact of the hash size (k). Given l=10, for each value of hash size k compute the F1-score for each query **q** ($F1_q$) using the reported result from query **q** in Part II(a) as search results and Part II(b) as ground-truth. Take the average of F1-score across all queries at k. Please report:

1. the F1-score plot with a varying k=[2,4,8]. (Note: $F1 = \frac{1}{|Q|}\sum_{q \in Q} F1_q$, $F1_q = \frac{TP}{TP + 1/2\,(FP+FN)}$, where TP (FP/FN) refers to the number of true positives (false positives / false negatives) of the top-K similar articles in the ground-truth for the query q (K is capped at 10)
2. the average query time in milliseconds with a varying k=[2,4,8]. **[20 pts]**

(b) Explain what you have observed from Part III(a) and suggest how you would tune the number of hash size (k) in terms of higher F1-score and lesser query time, respectively? **[5 pts]**