Comp 480/580: Assignment #4

Rice University — Due Date: Thursday, 12/05/2025

1 Build your own Approximate Search Engine (or plagiarism detector) with MinHash

In class, we learned MinHash (or Minwise Hashing), where we used different hash-functions and converts each set S into m values of $h_{min}(S)$ for these m functions $h_1, h_2, ..., h_m$. Recall that the probability that $h_{min}(A) = h_{min}(B)$ is true is equal to the Jaccard similarity $J(A, B)$ of set A and B.

1.1 Task 0

- 1. Implement simple MinHash generator. Given an input string and m, return m hashcodes.
- 2. Generate 100 MinHash values (i.e. m = 100 in the function above) of the following two strings: $S1$ = { *The mission statement of the WCSCC and area employers recognize the importance of good attendance on the job. Any student whose absences exceed 18 days is jeopardizing their opportunity for advanced placement as well as hindering his/her likelihood for successfully completing their program.*}

  $S1$ = { *The WCSCC's mission statement and surrounding employers recognize the importance of great attendance. Any student who is absent more than 18 days will loose the opportunity for successfully completing their trade program.*}

  Verify that the estimate of Jaccard similarity from MinHash is close to the actual Jaccard similarity. (similarity based on 3-gram representation of string).
- 3. Implement MinHash hashtable class which has insert and look-up function. The constructor takes in four parameters, K, number of hash functions and L, number of hash tables, B the size of hash tables (or the number of the buckets in hash tables). The function "insert" will take in hashcodes and an id. The role of the function is to insert this id into a different bucket in each hash table according to its hashcodes (as explained in class). Then "lookup" will retrieve all the items in the hash tables according to the supplied hashcodes (as explained in class).

1.1.1 Deliverables

MinHash.py

```
def MinHash(A, k):
...

class HashTable():
    def __init__(self, K, L, B, R):
        ...

    def insert(self, hashcodes, id):
        ...

    def lookup(self, hashcodes):
        ...
```

1.2 The Main Component (LSH Retreival)

- Download Aol dataset which includes anonymized user ids and click data. (url)
- **Data Preprocessing**: get the unique urls from the data file.

```
import csv
import pandas as pd

data = pd.read_csv('user-ct-test-collection-01.txt', sep="\t")
urllist = data.ClickURL.dropna().unique()
```

- Insert all URLs to your MinHash Hashtables. Use $K = 2, L = 50, B = 64, R = 2^{20}$ You can treat each URL as a string and use 3-gram in your MinHash function.
- **Task 1:** Sample 200 URLs from urllist as the query set. For each URL, compute the MinHash codes and retrieve all the items in the hash tables. Evaluate the quality of retrieved items by reporting the average Jaccard similarity of the query URL and retrieved URLs.
  1. Report the mean Jaccard similarity of the URL retrieved.
  2. Now for every retrieved set (candidate set), filter it to find the top-10 URLs only (based on actual Jaccard Similarity) . Report the mean Jaccard similarity of the top-10 URLs retrieved after filtering.
  3. Report the query time.
- **Task 2:** Write a function to compute the pairwise Jaccard similarity of those 200 URLs in the query set with all the other elements using a brute-force way. Use the computational time for this task to estimate the expected total time of computing the pairwise Jaccard similarity of all URLs.
  1. Report the total time and time per query, compare it with the one from the previous step.
- **Task 3:** Tuning K (K = $2, 3, 4, 5, 6$) and L ($L = 20, 50, 100$) observe how the average jaccard similarities of the query url and retrieved urls change accordingly.
  1. Report the mean Jaccard similarity of the URL retrieved and also the time per query for each of the combination.
- **Task 4:** Plot the S-curves. We all know that the probability of retrieving (P_x) and element x, having Jaccard similarity with query J_x, is given by $P_x = 1 - (1 - J_x^K)^L$. We will plot this curve for different values of K and L. We will only need two plots. On the X-axis, we have J_x varying from 0-1. On the Y-axis, we have the value of P_x. Make sure to plot different K and L, with a different color.
  1. **Plot 1:** Fix $L = 50$, vary $K = \{1, 2, 3, 4, 5, 6, 7\}$
  2. **Plot 2:** Fix $K = 4$, vary $L = \{5, 10, 20, 50, 100, 150, 200\}$

2 Adaptive Sampling and Random Sampling (Some Math)

We are trying to estimate the sum of the elements of a set, i.e. we have a set $S = w_1, w_2, w_3, ..., w_n$ and we want to calculate $T = \sum_{i=1}^{n} w_i$. However, the set is too large, i.e., n is big. Assume we have the capability to sample elements from the S such that w_i is sampled with probability p_i, which induces a distribution D on set S. Lets say, we sample k elements to form a sampled set $SS = x_1, x_2, ..., x_k$, where each $x_j \in SS$ is an i.i.d draw from D, i.e., the probability that x_j is w_a is given by $P(x_j) = p_a$, where $P(x_j)$ just denotes the probability that x_j. Note, when $p_i = \frac{1}{n}$, it a uniform sampling.

Please answer the following with mathematical proof

- Prove that $\hat{T} = \sum_{w_j \in SS} w_j / p_j$ is an unbiased estimator of T.
- Calculate the variance of $\hat{T}$ in terms of w_is and p_is
- Can we comment on how we can design p_is to get better (smaller) variance compared to Random sampling where $p_i = 1/n$