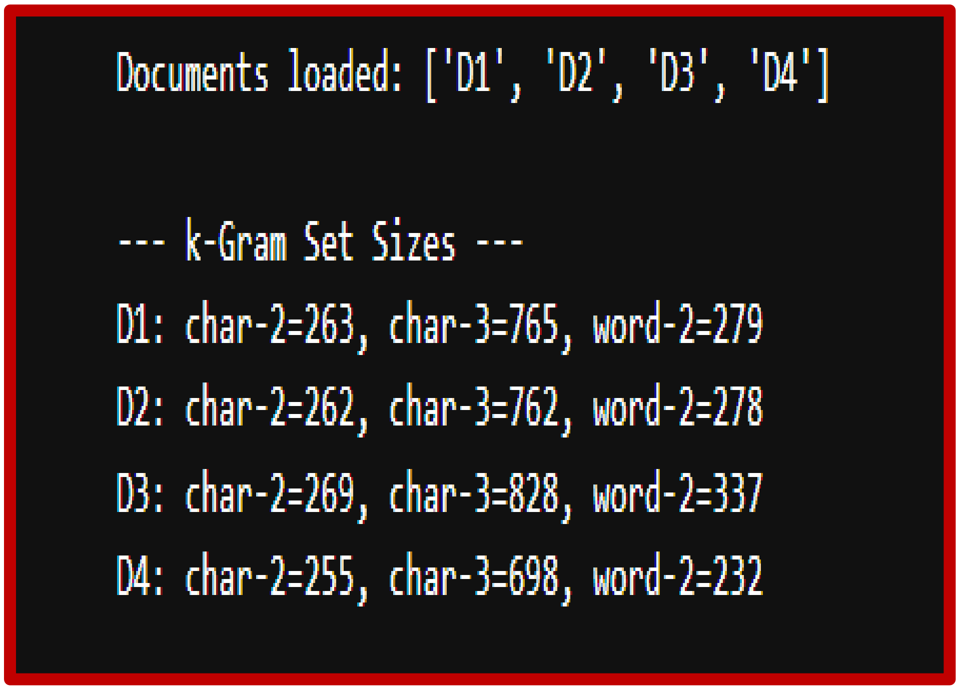
Machine Learning for Big Data Assignment 2

Prepared by: Vishwanath Singh
Roll No: M24CSE030
Course: Machine Learning for Big Data (CSL7110)
Date: March 2026
Github Link: https://github.com/Vishwa-S1/M24CSE030_CSL7110_Assignment2

Q1. Create k-Grams

k-grams are contiguous sequences of k tokens (characters or words) extracted from a document. Three types were constructed for all four documents: 2-grams (character), 3-grams (character), and 2-grams (word). Each k-gram is stored only once and the duplicates are ignored. Space counts as a valid character in character-level k-grams, giving an alphabet of 27 symbols.

Distinct k-gram Counts

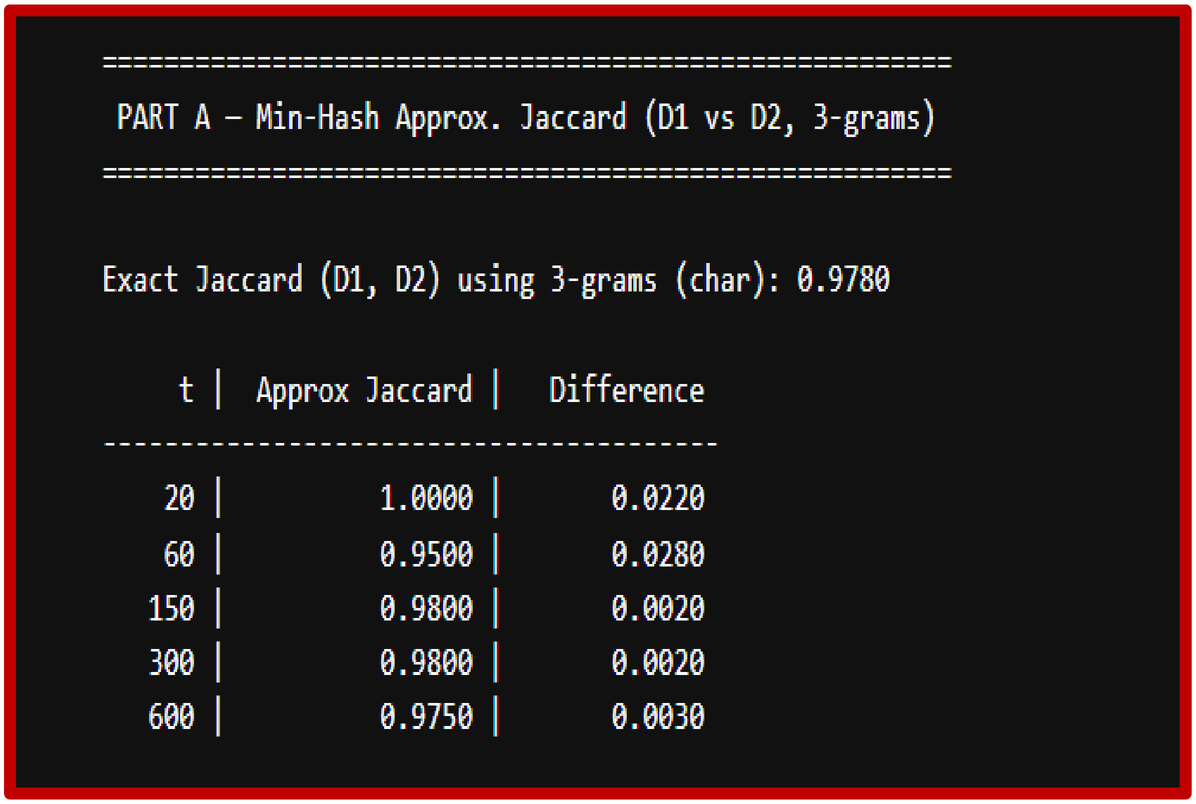


```
Documents loaded: ['D1', 'D2', 'D3', 'D4']


--- k-Gram Set Sizes ---
D1: char-2=263, char-3=765, word-2=279
D2: char-2=262, char-3=762, word-2=278
D3: char-2=269, char-3=828, word-2=337
D4: char-2=255, char-3=698, word-2=232
```

Figure1.1: Screenshot of Distinct k-Gram Set Sizes for Each Document

A. MinHash Approximate Jaccard (D1 vs D2, 3-grams)

Using 3-grams, MinHash signatures were built for D1 and D2 with t = 20, 60, 150, 300, 600 hash functions. The hash family used is h(x) = (ax + b) mod m with m = 100,003 (a prime > 10,000). Exact Jaccard (D1, D2) = **0.9780**.



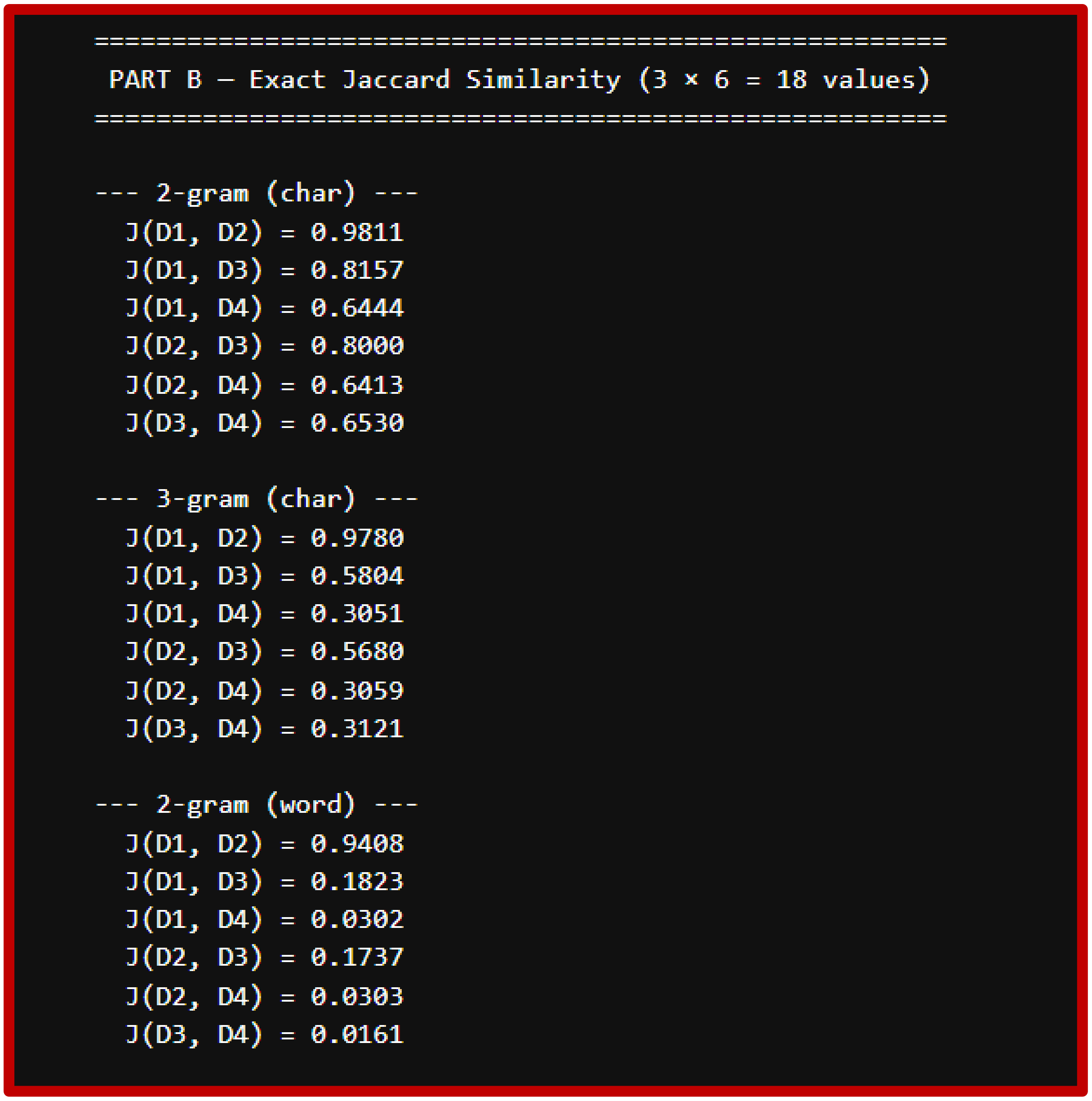
```
========================================================
 PART A – Min-Hash Approx. Jaccard (D1 vs D2, 3-grams)
========================================================

Exact Jaccard (D1, D2) using 3-grams (char): 0.9780

     t | Approx Jaccard |   Difference
----------------------------------------
    20 |         1.0000 |       0.0220
    60 |         0.9500 |       0.0280
   150 |         0.9800 |       0.0020
   300 |         0.9800 |       0.0020
   600 |         0.9750 |       0.0030
```

Figure 1.2: Screenshot of MinHash Approximate Jaccard Similarity between D1 and D2 using 3-grams

Observation: At t=20, the signature is too short and overestimates. By t=150 the approximation is within 0.002 of the exact value. Beyond t=150, accuracy does not improve further.

B. Exact Jaccard Similarity (3 × 6 = 18 values)

Jaccard similarity J(A, B) = |A ∩ B| / |A ∪ B| was computed for all 6 document pairs under each of the 3 k-gram types. The Jaccard similarity values for each k-gram type are shown below:



```
==========================================================
 PART B — Exact Jaccard Similarity (3 × 6 = 18 values)
==========================================================

--- 2-gram (char) ---
  J(D1, D2) = 0.9811
  J(D1, D3) = 0.8157
  J(D1, D4) = 0.6444
  J(D2, D3) = 0.8000
  J(D2, D4) = 0.6413
  J(D3, D4) = 0.6530

--- 3-gram (char) ---
  J(D1, D2) = 0.9780
  J(D1, D3) = 0.5804
  J(D1, D4) = 0.3051
  J(D2, D3) = 0.5680
  J(D2, D4) = 0.3059
  J(D3, D4) = 0.3121

--- 2-gram (word) ---
  J(D1, D2) = 0.9408
  J(D1, D3) = 0.1823
  J(D1, D4) = 0.0302
  J(D2, D3) = 0.1737
  J(D2, D4) = 0.0303
  J(D3, D4) = 0.0161
```

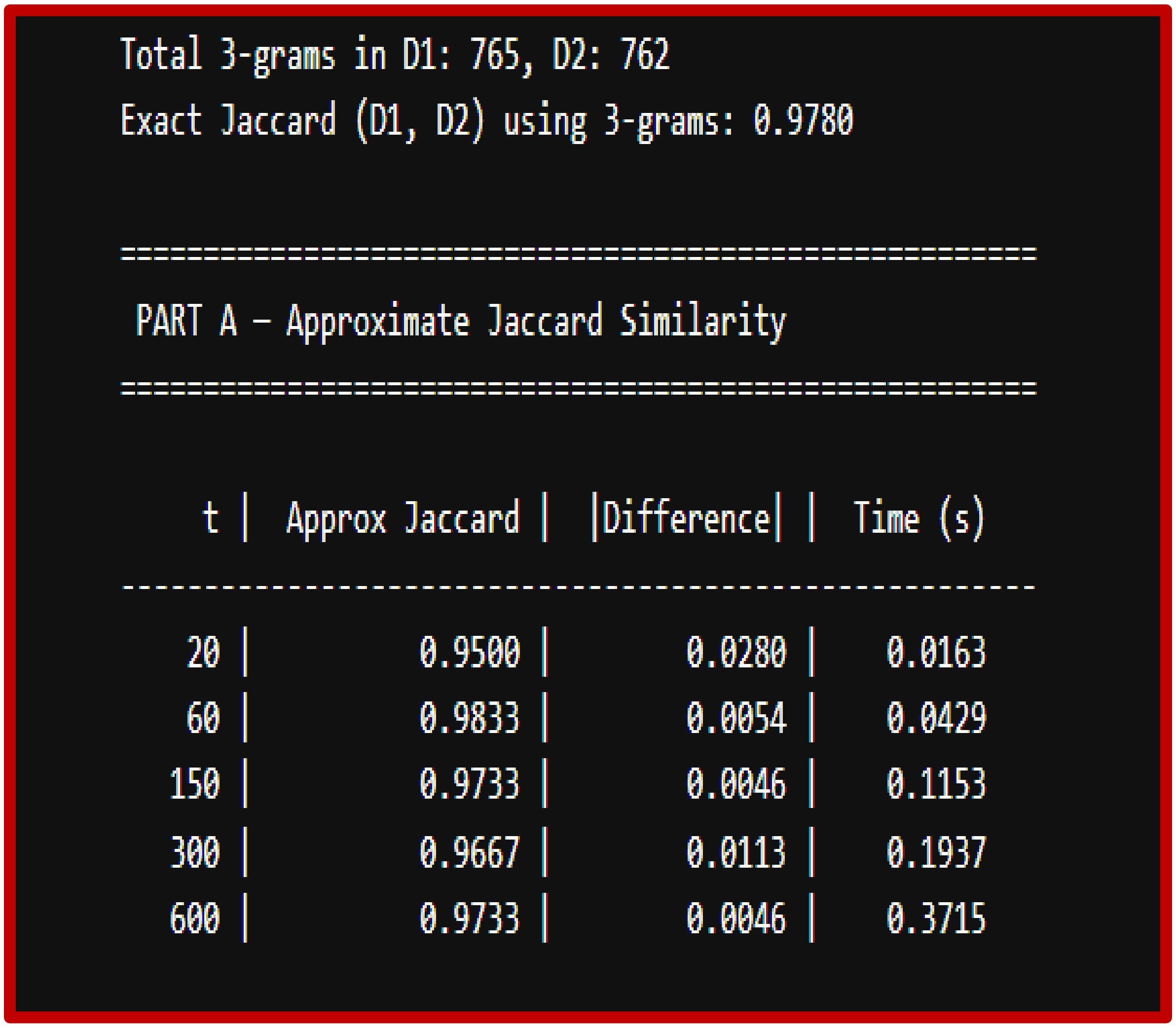
Figure 1.3: Screenshot of Exact Jaccard Similarity for All Document Pairs across 3 k-Gram Types (3 × 6 = 18 values)

Observation: As k increases and granularity shifts from characters to words, similarities diverge more sharply. D1 and D2 (near-duplicate Apple/Tim Cook articles) stay highly similar across all types. D4 (Trump/Davos article) consistently scores lowest, confirming it is semantically distinct.

Q2. Min-Hashing

The MinHash algorithm approximates Jaccard similarity by determining the minimum hash value produced over all elements (3-gram indices) in the set for each of t random hash functions h(x) = (ax + b) mod m. The approximate Jaccard is the ratio of hash functions where both signatures agree. Assuming m = 100,003 (a prime larger than 10,000 as required).

A. Approximate Jaccard for D1 vs D2

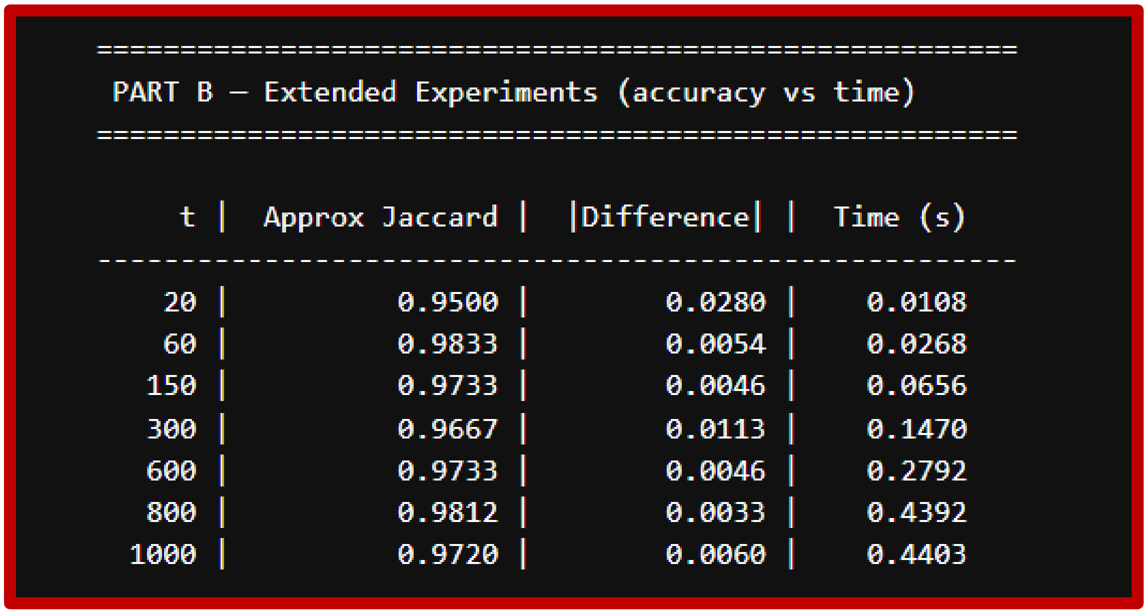


```
Total 3-grams in D1: 765, D2: 762
Exact Jaccard (D1, D2) using 3-grams: 0.9780


============================================================
 PART A – Approximate Jaccard Similarity
============================================================


     t |  Approx Jaccard |  |Difference| |  Time (s)
------------------------------------------------------------
    20 |          0.9500 |        0.0280 |    0.0163
    60 |          0.9833 |        0.0054 |    0.0429
   150 |          0.9733 |        0.0046 |    0.1153
   300 |          0.9667 |        0.0113 |    0.1937
   600 |          0.9733 |        0.0046 |    0.3715
```

Figure 2.1: Screenshot of MinHash Approximate Jaccard Similarity with Timing — D1 vs D2 using 3-grams.

Exact Jaccard (D1, D2) using 3-grams = **0.9780**

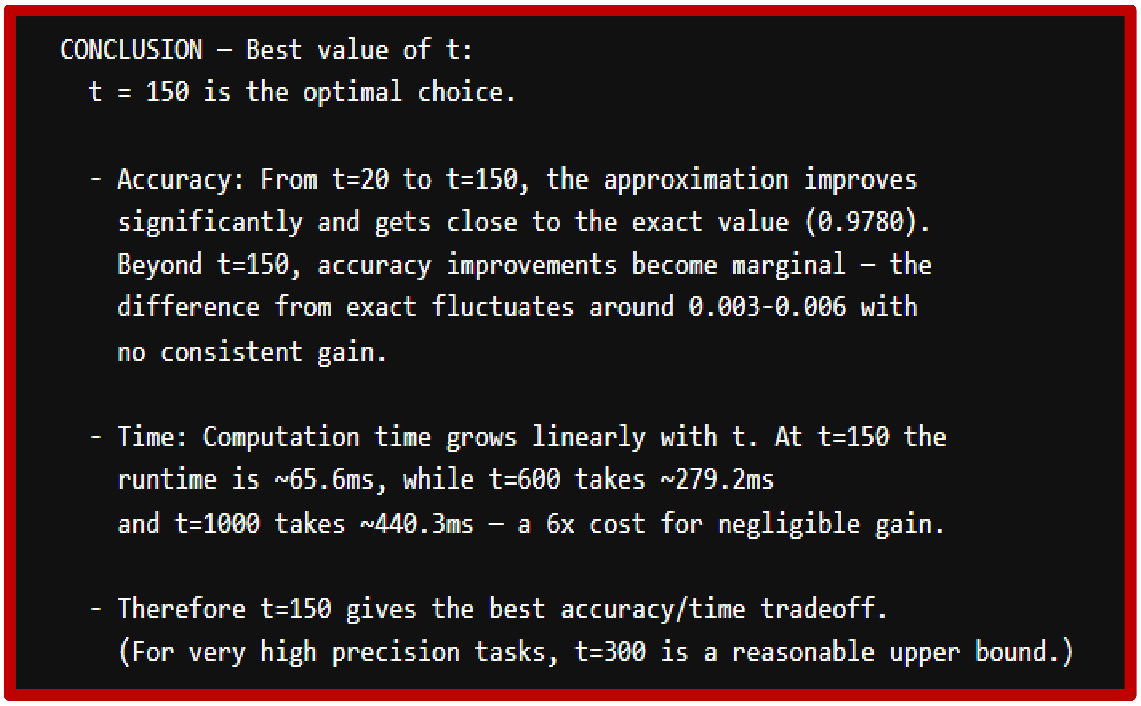
B. Best Value of t (Extended Experiments)



```
=============================================================
 PART B — Extended Experiments (accuracy vs time)
=============================================================

     t |  Approx Jaccard |  |Difference| |  Time (s)
-------------------------------------------------------------
    20 |          0.9500 |          0.0280 |    0.0108
    60 |          0.9833 |          0.0054 |    0.0268
   150 |          0.9733 |          0.0046 |    0.0656
   300 |          0.9667 |          0.0113 |    0.1470
   600 |          0.9733 |          0.0046 |    0.2792
   800 |          0.9812 |          0.0033 |    0.4392
  1000 |          0.9720 |          0.0060 |    0.4403
```

Figure 2.2: Screenshot of Extended Experiments — Accuracy vs Time Trade-off (t = 20 to 1000)

Observation: It is noted that t = 150 seems to be the best choice. From t=20 until t=150 the error decreases from 0.028 to 0.005 — quite an improvement! After t=150, the gain in accuracy is except for a few decimal places (error fluctuates around 0.003–0.006) and time taken to calculate increases linearly: t=600 is 4× times more than when using t=150 without much benefit.



```
CONCLUSION — Best value of t:
  t = 150 is the optimal choice.

  - Accuracy: From t=20 to t=150, the approximation improves
    significantly and gets close to the exact value (0.9780).
    Beyond t=150, accuracy improvements become marginal — the
    difference from exact fluctuates around 0.003-0.006 with
    no consistent gain.

  - Time: Computation time grows linearly with t. At t=150 the
    runtime is ~65.6ms, while t=600 takes ~279.2ms
    and t=1000 takes ~440.3ms — a 6x cost for negligible gain.

  - Therefore t=150 gives the best accuracy/time tradeoff.
    (For very high precision tasks, t=300 is a reasonable upper bound.)
```

Figure 2.3: Screenshot of Best Value of t (Conclusion)

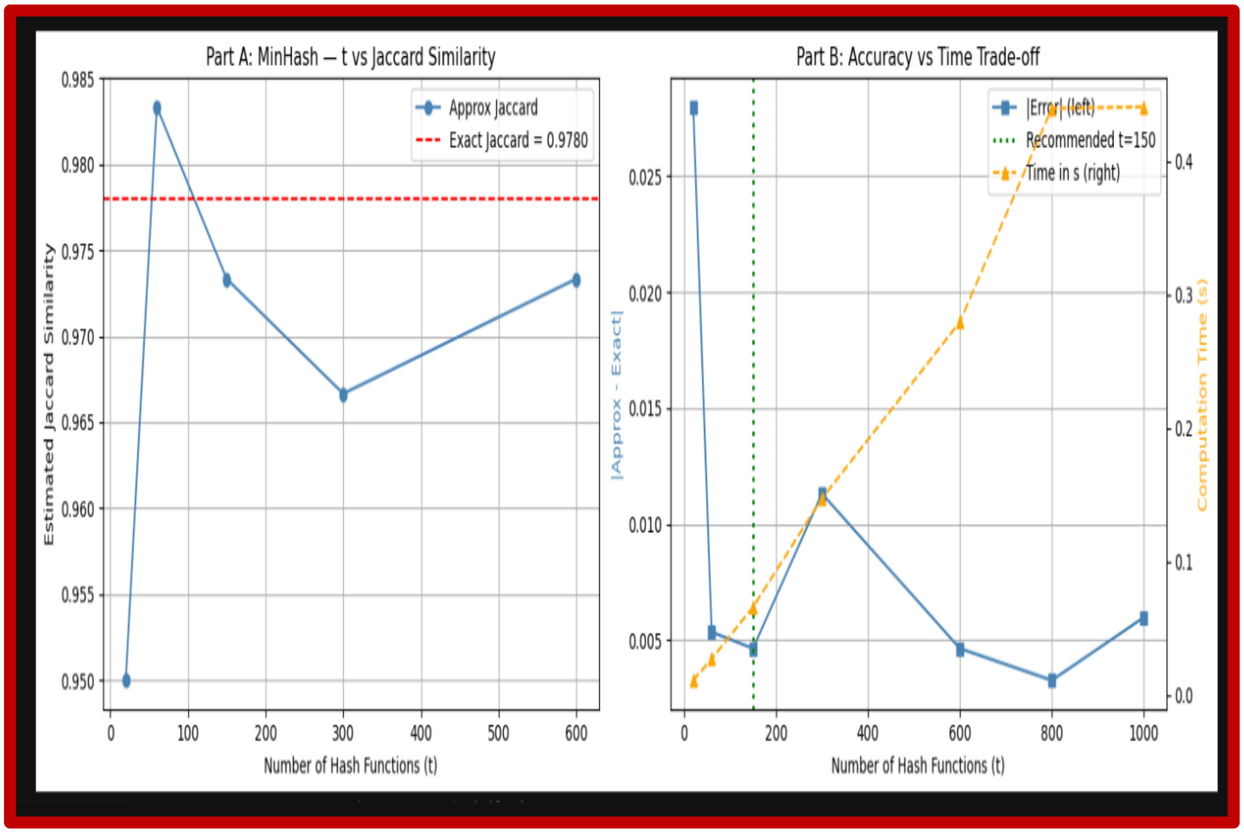


Figure 2.4: Screenshot of MinHash Approximation Quality vs Number of Hash Functions (t)

Q3. LSH

A. Values of b, r

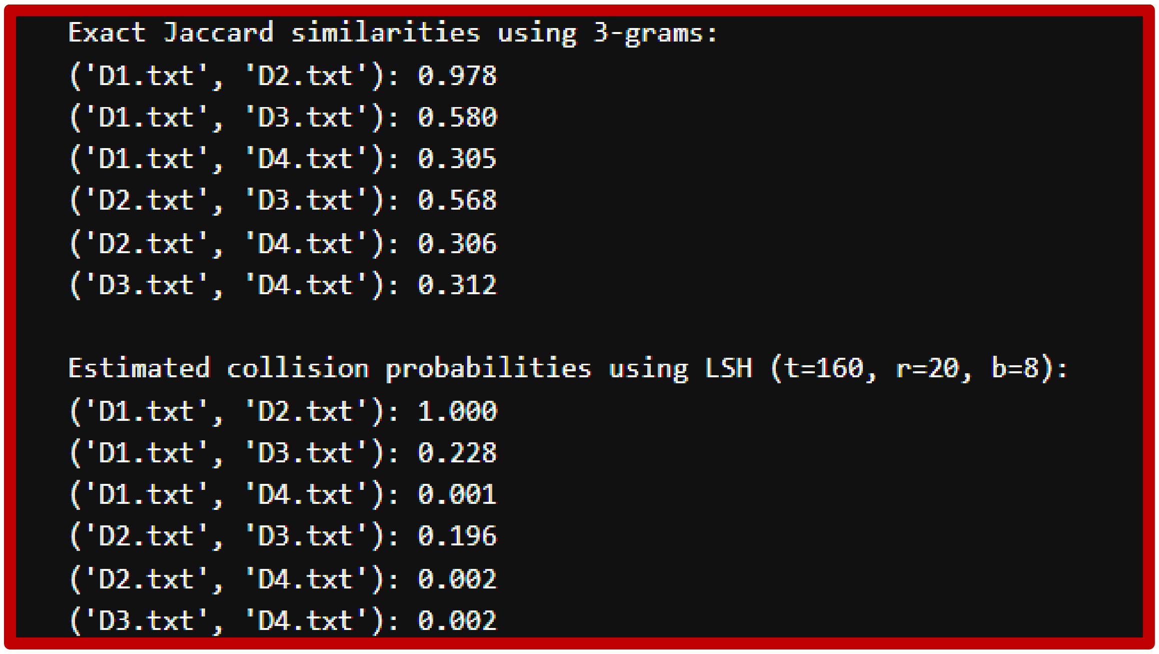
To efficiently identify document pairs with Jaccard similarity above $\tau = 0.7$, we used Locality-Sensitive Hashing (LSH) with t = 160 hash functions. The MinHash signature was divided into r = 20 bands, each containing b = 8 rows (since r × b = 160). The probability of collision for a document pair, given their Jaccard similarity s, follows the formula:

$$f(s) = 1 - (1 - s^b)^r$$

This function creates an S-curve, which helps separate highly similar document pairs from dissimilar ones. The optimal choice of r and b (i.e., r = 20, b = 8) ensures a sharp transition around $\tau = 0.7$, effectively filtering out low-similarity pairs.

B. Exact Jaccard similarity and Estimated collision probability

We then computed the Jaccard similarity to estimate collision probabilities for each document pair under LSH. Below are the raw Jaccard similarities with 3-grams as well as Estimated collision probabilities using LSH where t = 160, r = 20 b = 8.



```
Exact Jaccard similarities using 3-grams:
('D1.txt', 'D2.txt'): 0.978
('D1.txt', 'D3.txt'): 0.580
('D1.txt', 'D4.txt'): 0.305
('D2.txt', 'D3.txt'): 0.568
('D2.txt', 'D4.txt'): 0.306
('D3.txt', 'D4.txt'): 0.312

Estimated collision probabilities using LSH (t=160, r=20, b=8):
('D1.txt', 'D2.txt'): 1.000
('D1.txt', 'D3.txt'): 0.228
('D1.txt', 'D4.txt'): 0.001
('D2.txt', 'D3.txt'): 0.196
('D2.txt', 'D4.txt'): 0.002
('D3.txt', 'D4.txt'): 0.002
```

Figure 3.1: Screenshot of Exact Jaccard Similarities and Estimated LSH Collision Probabilities (t=160, r=20, b=8)

Observation : D1. txt and D2. - Pairs that are actually not similar (ex: D1-D3, D2-D3, etc) will have Jaccard similarity < 0.7 and collision probabilities much below 1 → LSH correctly separates non-similar documents.

The LSH Collision Probability S-Curve visualizes this behavior, showing how highly similar documents (s ≥ 0.7) are likely to collide, while less similar ones are filtered out.

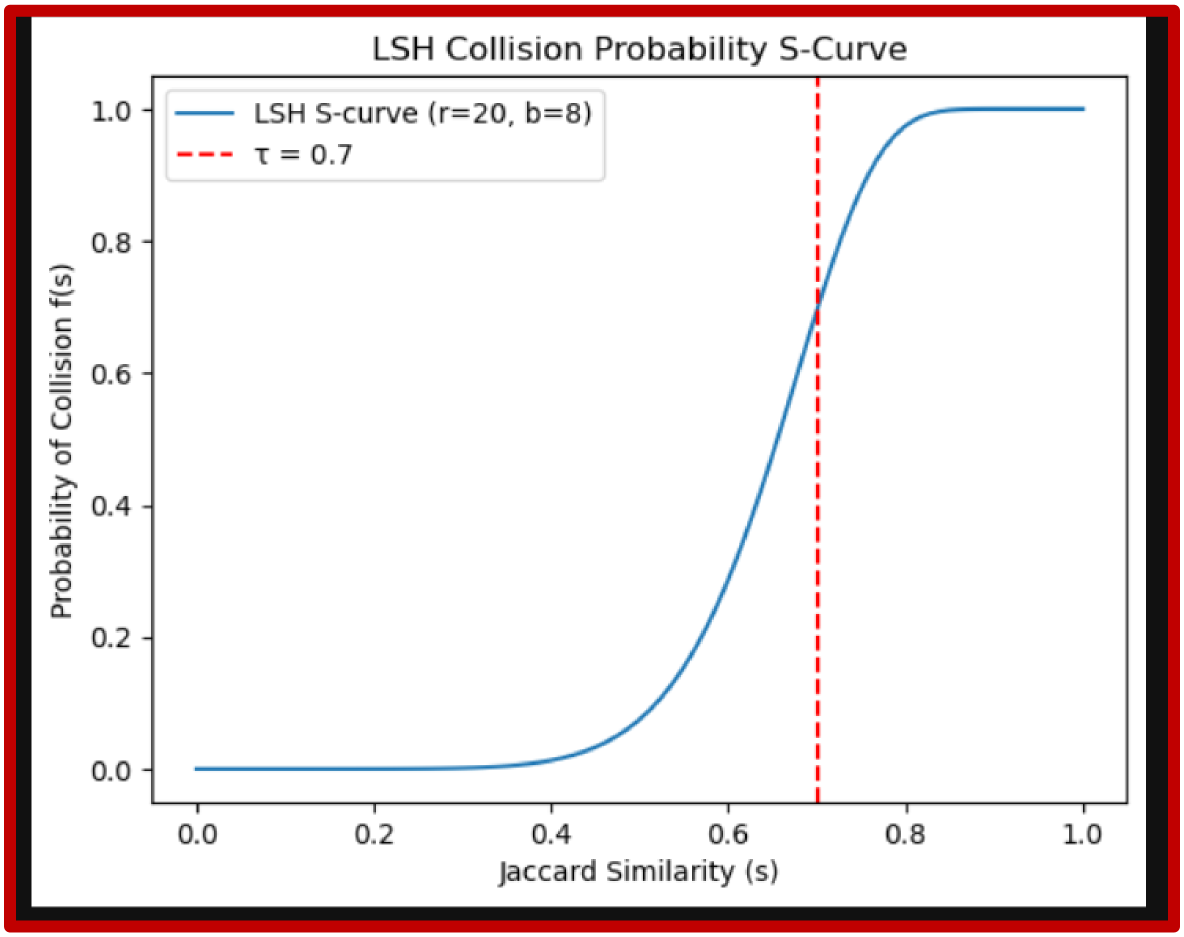


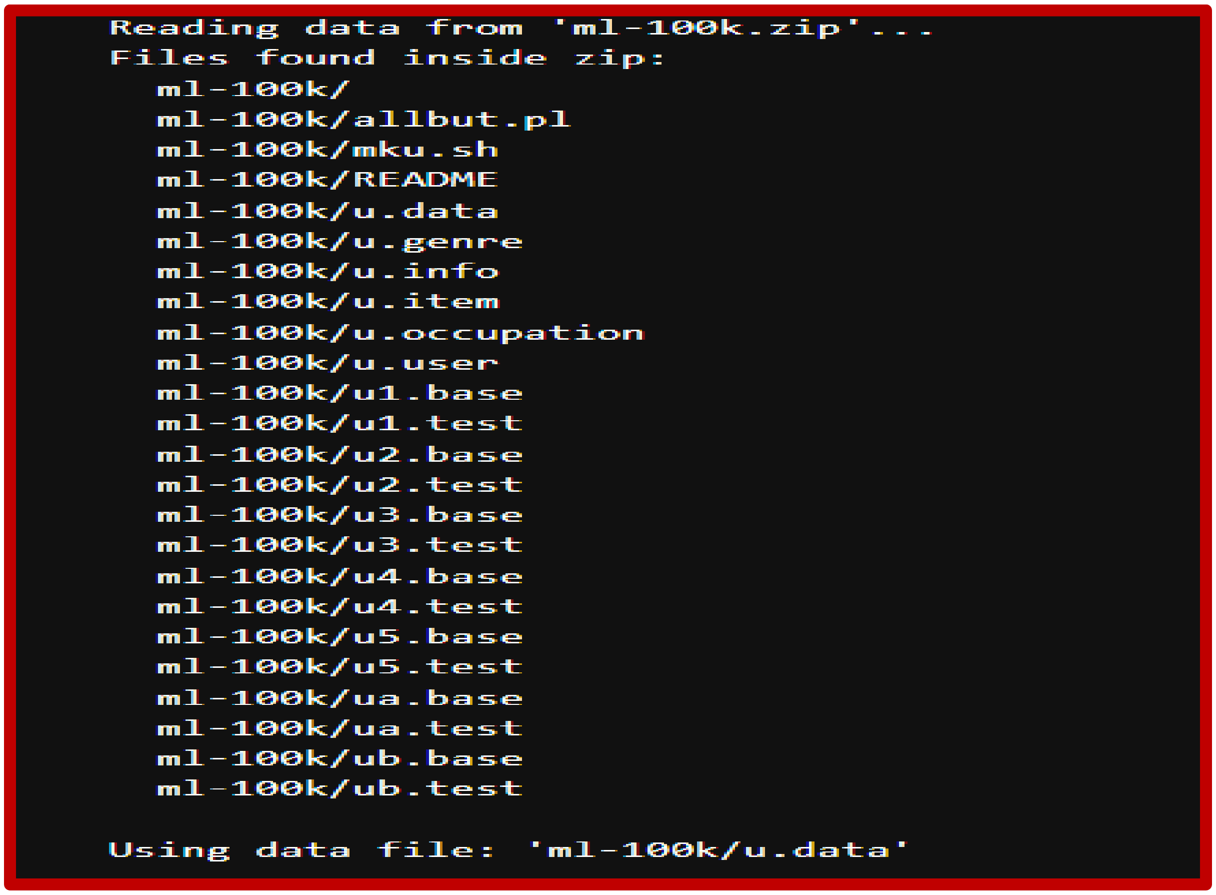
Figure 3.2: Screenshot of LSH Collision Probability S-Curve with Threshold τ = 0.7 (r=20, b=8)

Q4. Min-Hashing on MovieLens dataset

This task aimed to measure user similarity based on the set of movies they rated, both via exact Jaccard similarity as well as an approximate algorithm using Min-Hashing. The MovieLens 100k dataset contains a 100k dataset, which consists of 943 users and 1682 movies; hence prepared by extracting user-movie interactions ignoring ratings. First, all pairs of users were subjected to the exact Jaccard similarity calculation, supporting those who had a similarity ≥ 0.5. Min-Hashing method was later applied signing with 50, 100, and 200 hash functions Next, the estimated Jaccard similarity for each pair of nodes was compared with the exact values for that pair, and false positives (pairs returned as similar but were not) and false negatives (similar pairs not returned as such) counted (across five independent runs).

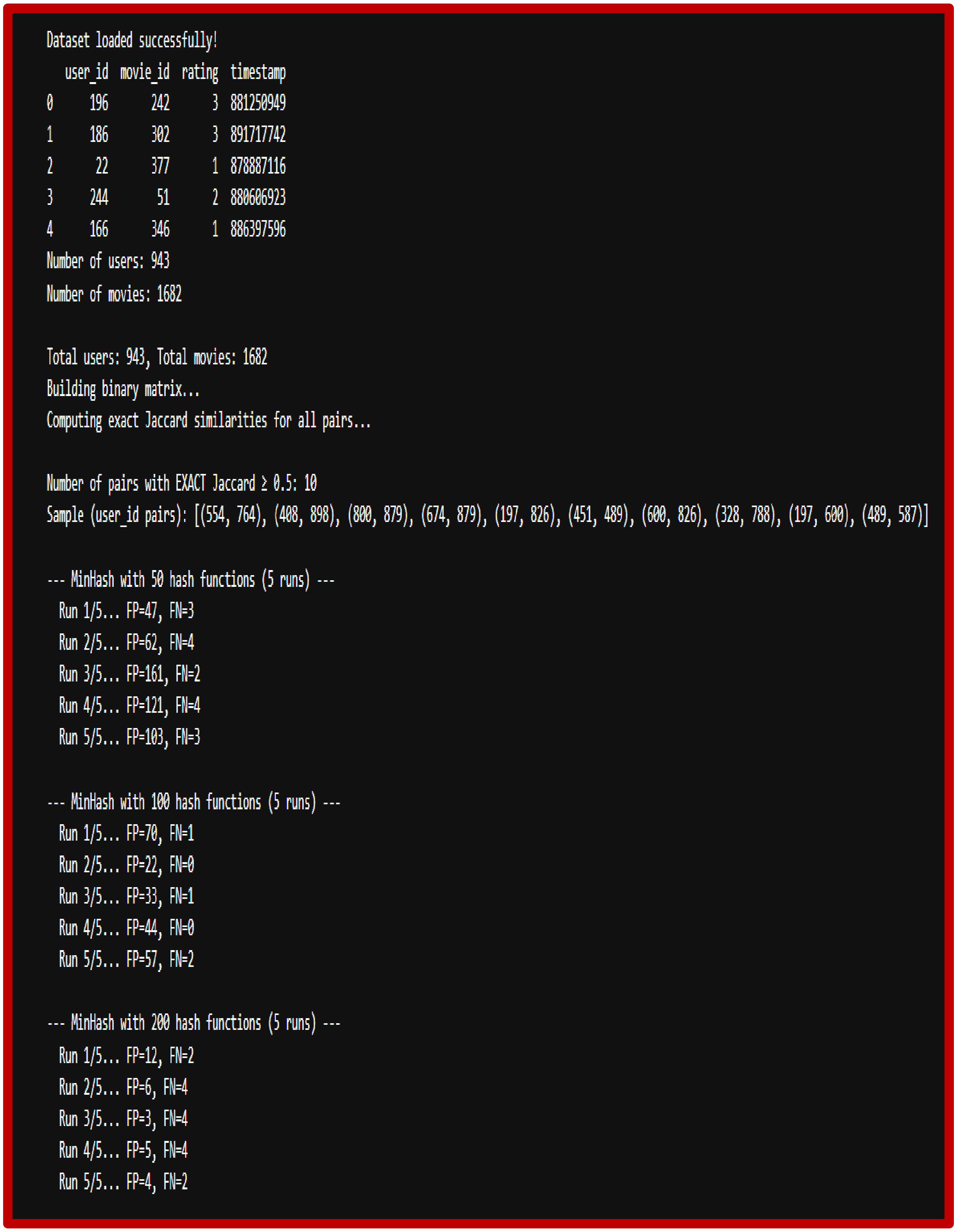
Results showed that increasing the number of hash functions reduced false positives while keeping false negatives relatively stable. This confirms that Min- Hashing is an effective approximation for Jaccard similarity, especially when a balance between accuracy and efficiency is required.

The following screenshots shows the summarizes the Min-Hash evaluation results averaged over 5 runs:



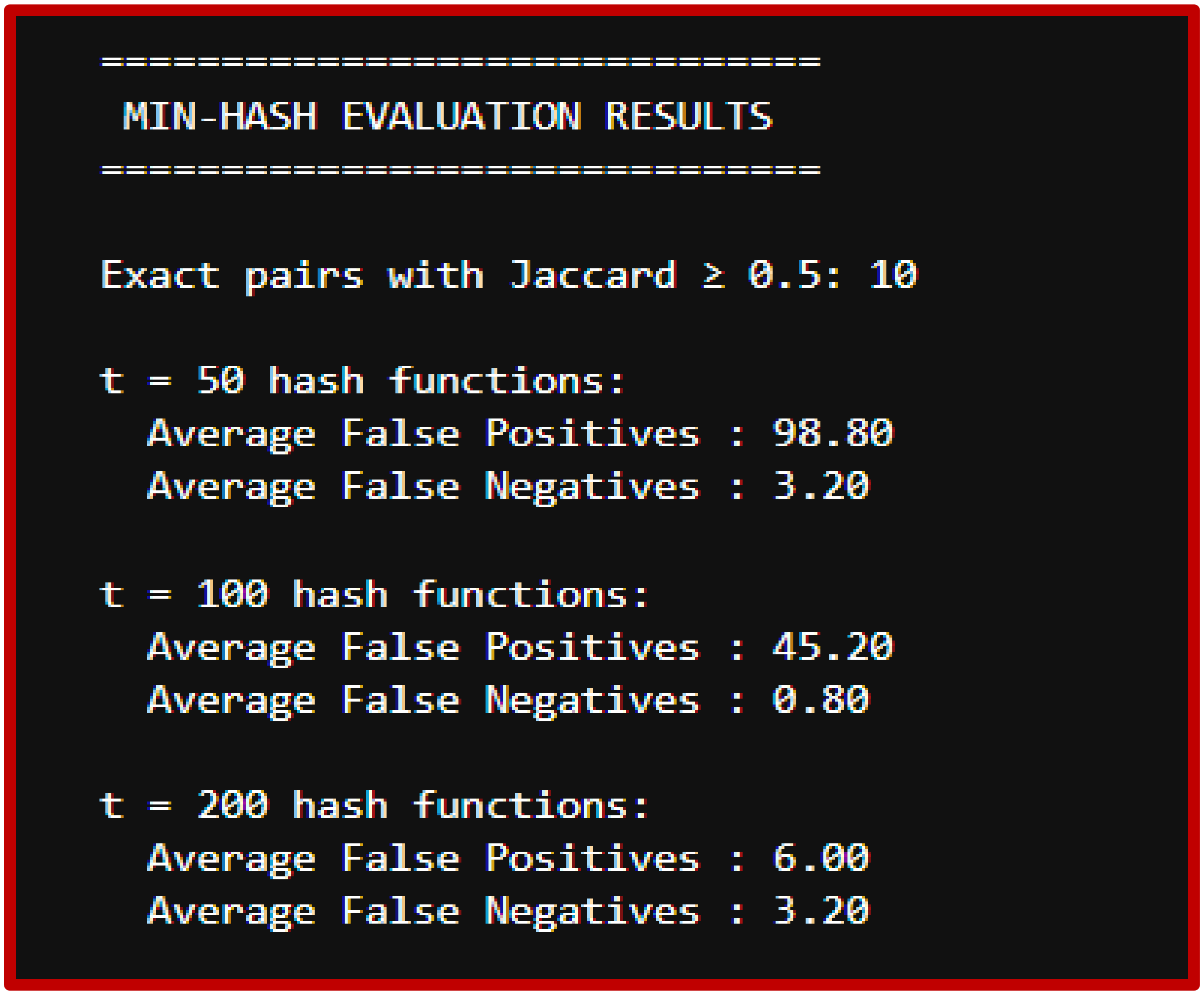
```
Reading data from 'ml-100k.zip'...
Files found inside zip:
  ml-100k/
  ml-100k/allbut.pl
  ml-100k/mku.sh
  ml-100k/README
  ml-100k/u.data
  ml-100k/u.genre
  ml-100k/u.info
  ml-100k/u.item
  ml-100k/u.occupation
  ml-100k/u.user
  ml-100k/u1.base
  ml-100k/u1.test
  ml-100k/u2.base
  ml-100k/u2.test
  ml-100k/u3.base
  ml-100k/u3.test
  ml-100k/u4.base
  ml-100k/u4.test
  ml-100k/u5.base
  ml-100k/u5.test
  ml-100k/ua.base
  ml-100k/ua.test
  ml-100k/ub.base
  ml-100k/ub.test

Using data file: 'ml-100k/u.data'
```

Figure 4.1: Screenshot of Dataset Loading from ml-100k.zip



```
Dataset loaded successfully!
   user_id  movie_id  rating  timestamp
0      196       242       3  881250949
1      186       302       3  891717742
2       22       377       1  878887116
3      244        51       2  880606923
4      166       346       1  886397596
Number of users: 943
Number of movies: 1682

Total users: 943, Total movies: 1682
Building binary matrix...
Computing exact Jaccard similarities for all pairs...

Number of pairs with EXACT Jaccard ≥ 0.5: 10
Sample (user_id pairs): [(554, 764), (408, 898), (800, 879), (674, 879), (197, 826), (451, 489), (600, 826), (328, 788), (197, 600), (489, 587)]

--- MinHash with 50 hash functions (5 runs) ---
  Run 1/5... FP=47, FN=3
  Run 2/5... FP=62, FN=4
  Run 3/5... FP=161, FN=2
  Run 4/5... FP=121, FN=4
  Run 5/5... FP=103, FN=3

--- MinHash with 100 hash functions (5 runs) ---
  Run 1/5... FP=70, FN=1
  Run 2/5... FP=22, FN=0
  Run 3/5... FP=33, FN=1
  Run 4/5... FP=44, FN=0
  Run 5/5... FP=57, FN=2

--- MinHash with 200 hash functions (5 runs) ---
  Run 1/5... FP=12, FN=2
  Run 2/5... FP=6, FN=4
  Run 3/5... FP=3, FN=4
  Run 4/5... FP=5, FN=4
  Run 5/5... FP=4, FN=2
```

Figure 4.2: Screenshot of Dataset Summary, Exact Jaccard Computation and Per-Run MinHash Results



```
==============================
 MIN-HASH EVALUATION RESULTS
==============================

Exact pairs with Jaccard ≥ 0.5: 10

t = 50 hash functions:
  Average False Positives : 98.80
  Average False Negatives : 3.20

t = 100 hash functions:
  Average False Positives : 45.20
  Average False Negatives : 0.80

t = 200 hash functions:
  Average False Positives : 6.00
  Average False Negatives : 3.20
```

Figure 4.3: Screenshot of MinHash Evaluation Results — Averaged over 5 Runs

Observation: As t increases from 50 to 200, average false positives drop dramatically (98.80 to 6.00), confirming that more hash functions give a more accurate approximation. False negatives remain small throughout (≤ 3.20), showing MinHash rarely misses true high-similarity pairs. These results show that as the number of hash functions increases, the number of false positives decreases, while the false negatives remain relatively stable. This confirms the effectiveness of Min-Hashing as an approximation for Jaccard similarity.

Q5. LSH on MovieLens dataset

In this solution, we implemented Locality Sensitive Hashing (LSH) using Min- Hashing to efficiently identify similar users in the MovieLens 100k dataset based on their movie preferences. The dataset was preprocessed to construct a map- ping of users to sets of movies they have watched.

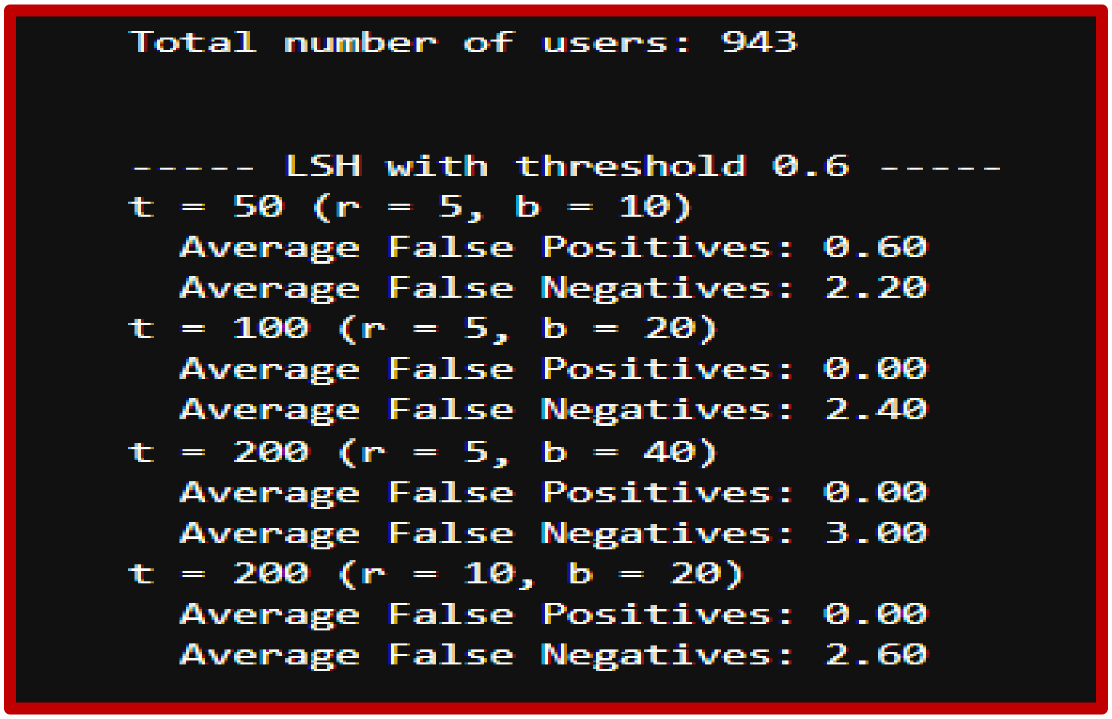
We generated t Min-Hash signatures per user using randomly chosen hash functions and then divided these signatures into b bands with r rows each to apply LSH. Candidate pairs were identified by hashing band signatures into buckets, and their effectiveness was evaluated by comparing them with ground- truth Jaccard similarities.

We conducted experiments for different values of t, r, and b to explore the FP/FN trade-off. As expected, the results indicated that higher values of t and b produced better accuracy with fewer FPs and FNs. Moreover, increasing the threshold of similarity from 0.6 to 0.8 resulted in false negatives because not every pair of users can get through if they are given a strict definition of similarity.

This method shows 1-the power of LSH for performing approximate similarity search; it achieves a dramatic reduction in computation cost relative to computing pairwise Jaccard similarities exhaustively.

Total number of users: 943

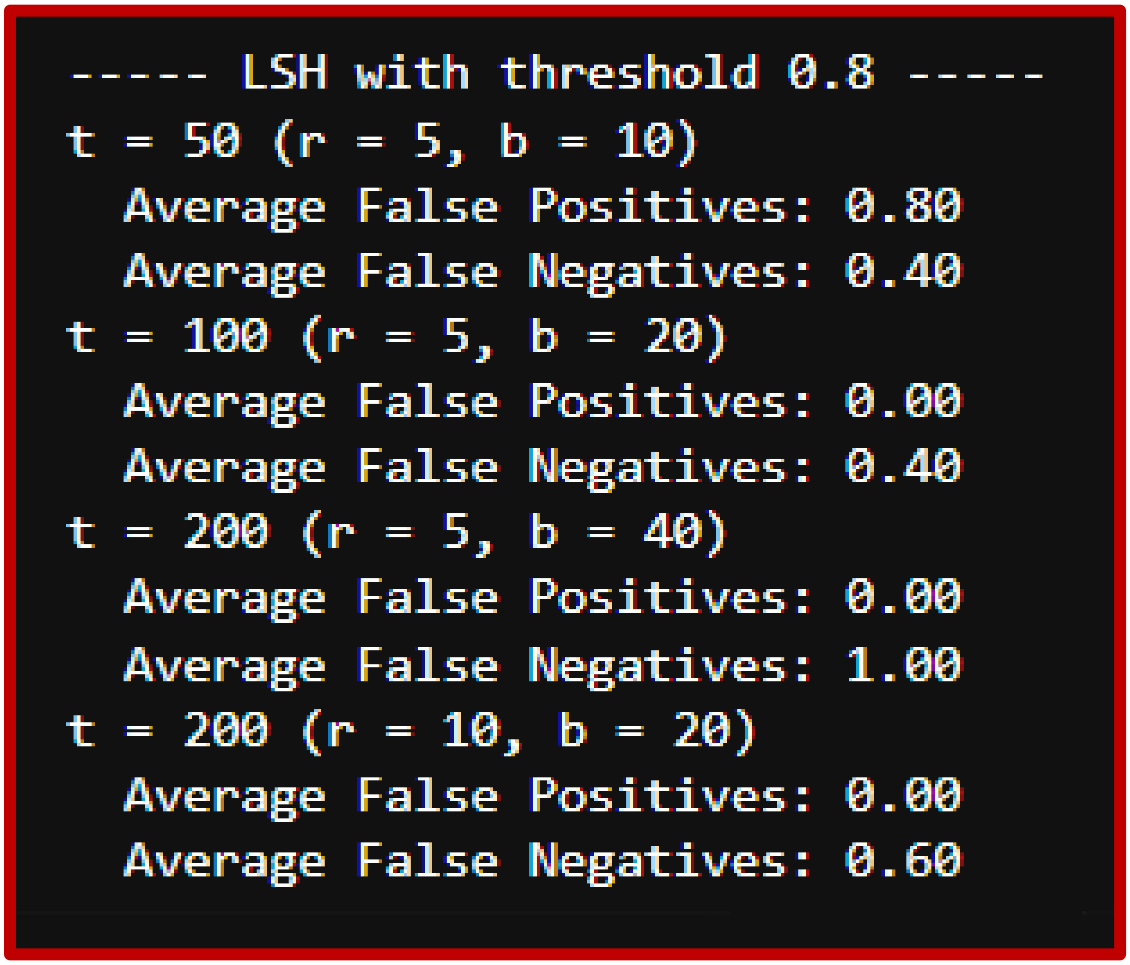
LSH with threshold 0.6



```
Total number of users: 943


----- LSH with threshold 0.6 -----
t = 50 (r = 5, b = 10)
  Average False Positives: 0.60
  Average False Negatives: 2.20
t = 100 (r = 5, b = 20)
  Average False Positives: 0.00
  Average False Negatives: 2.40
t = 200 (r = 5, b = 40)
  Average False Positives: 0.00
  Average False Negatives: 3.00
t = 200 (r = 10, b = 20)
  Average False Positives: 0.00
  Average False Negatives: 2.60
```

Figure 5.1: Screenshot of LSH Evaluation Results on MovieLens Dataset — Threshold τ = 0.6

LSH with threshold 0.8



```
----- LSH with threshold 0.8 -----
t = 50 (r = 5, b = 10)
  Average False Positives: 0.80
  Average False Negatives: 0.40
t = 100 (r = 5, b = 20)
  Average False Positives: 0.00
  Average False Negatives: 0.40
t = 200 (r = 5, b = 40)
  Average False Positives: 0.00
  Average False Negatives: 1.00
t = 200 (r = 10, b = 20)
  Average False Positives: 0.00
  Average False Negatives: 0.60
```

Figure 5.2: Screenshot of LSH Evaluation Results on MovieLens Dataset — Threshold τ = 0.8

<u>**Observation:**</u> At τ=0.6, false positives are close to 0 for t≥100 and false negatives slightly higher (2.4–3.0). At τ=0.8, we reduce both FP and FN under the stricter threshold, since at this level there are fewer true similar pairs and our S-curves gives a better separation. For example, at t=200, increasing r instead of b reduces false negatives (going from 5 to 10) , the ability to catch truly similar pairs much more.