# Data Mining – Homework 1

Madeleine Marangoz, Benedict Wolff – Group 2

November 7, 2025

## 1 Data

The data used for this assignment can be found on:
https://archive.ics.uci.edu/datasets/?skip=0take=10sort=descorderBy=NumHitssearch=Types=Text and downloading the "SMS Spam Collection". In this assignment the corpus size, meaning the number of SMS messages extracted from this dataset, was set to `num_docs` = 500.

## 2 Methods

The algorithm finds textually similar documents using the Shingling, MinHashing, and Locality-Sensitive Hashing (LSH). Running the program is straight forward since everything is in the same file. The parameters for this assignment are the shingle length, the signature length, the corpus size and the similarity filthering threshold. Below is table 1 showing each parameter setting. Each subsection later dives deeper into detail about every step of this project solution.

| **Parameter** | **Setting** | **Meaning** |
|---|---|---|
| Shingle length ($k$) | **5 characters** | The length of the $k$-shingles generated from each document. |
| MinHash signature length ($n$) | **100 hash functions** | Number of hash functions used to produce the MinHash signature of each document. |
| Corpus size ($N$) | **500 SMS messages** | Total number of documents processed in the experiment. |
| Similarity threshold ($s$) | **0.6** | Minimum similarity required for two documents to be considered a match. |

Table 1: Parameter settings and their meaning.

### 2.1 Shingling

Each document is lowercased and normalized, then converted into a set of character k-shingles. The shingles length was set to size $k = 5$, this is the number of characters in each shingle and this value was picked specifically because 5 is the average length of english words and SMS messages tend to be short.

To avoid hash collisions, each unique shingle is mapped to an integer ID (dictionary encoding). The document becomes a set of integers suitable for set operations which is needed for computing the jaccard similarity of two sets of integers. So for any two documents, similarity is

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

over their shingle ID sets.

### 2.2 MinHashing

Each document is represented using a MinHash signature. The parameter `num_hashes` was set to 100 which is the MinHash signature length. This signature of length n (100) is computed using n independent hash functions of the form h(x) = (a·x + b) mod p over shingle IDs. For each h, the minimum hashed value across the set is stored.

### 2.3 Locality-Sensitive Hashing (LSH)

To avoid comparing all MinHash signatures, each signature is split into bands. Tuples within each band are hashed into buckets and documents that fall into the same bucket in at least one band are treated as similar pairs. Only these pairs are verified using either Jaccard similarity or the MinHash estimate. The similarity threshold was set to 0.6.

## 3 Results

Lets first start with looking at the actual results, meaning the messages that were the most similar from the corpus shown in table 2, and then diving deeper into the LSH, Jaccard etc.

| **Document A (SMS content)** | **Document B (SMS content)** | **Jaccard** |
|---|---|---|
| *"As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your Callertune. Press *9 to copy your friend's Callertune."* | Same message as to the left (exact duplicate). | 1.0000 |
| *"Congrats! 1 year special cinema pass for 2 is yours. Call 09061209465 now!"* | Same message as to the left (exact duplicate). | 1.0000 |
| *"URGENT! We are trying to contact you. Last weekend's draw shows that you won a prize. Call 09061790121 now!"* | *"URGENT! We are trying to contact you. Last weekend's draw shows that you have won. Call 09061790121 now!"* | 0.6344 |
| *"That's cool. I am a gentleman and will treat you with dignity and respect."* | *"Ok. I am a gentleman and will treat you with dignity and respect."* | 0.8286 |

Table 2: Most similar SMS message pairs in the dataset, showing full SMS text and exact Jaccard similarity based on hashed shingle sets.

The first two rows in table 2 show perfect matches (Jaccard = 1.0), meaning that the messages are exact duplicates. These occurred multiple times in the dataset since they are spam messages. The third row shows a pair with a Jaccard similarity of 0.6344, indicating that although not identical, the messages are highly similar. The messages differ only in a few words but the jaccard similarity is at 0.6344, since the shingle overlap remains large despite minor wording differences. The final row has a jaccard similarity of 0.8286 and are almost duplicates.

### 3.1 Confusion matrix & LSH

Below is a confusion matrix showing the number of true positives (TP), which is the jaccard, the false positives (FP), the false negatives (FN) and the true negatives (TN). These values are the number of document pairs compared.

$$\textbf{Confusion Matrix:} \quad \begin{bmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{bmatrix} = \begin{bmatrix} 28 & 4 \\ 0 & 124718 \end{bmatrix}$$

After running the program, the LSH found 32 candidate pairs in 0.0050 seconds where 28 of them were true positives and 4 of them were false positives. However, there were a lot of true negatives. This is because most SMS messages in the dataset are not similar at all, so almost every pair is a "negative". Only 28 are actually similar (duplicates or near duplicates). This makes sense since most SMS messages are rarely duplicates unless it is spam. Furthermore, there were 0 false negatives which means LSH successfully manage to not miss any true similar documents.

### 3.2 Execution time

Table 3 shows the execution time of each processing stage. The most time consuming steps are the Exact Jaccard computation and MinHash signature comparison, each taking roughly 1 second. This is expected since Jaccard requires comparing all pairs of documents at the shingle level which grows quadratically (time complexity of $O(N^2)$) with the corpus size[1]. In contrast, the MinHashing step takes less than a second to compute compact signature vectors that summarize each document. The final stage, Locality Sensitive Hashing (LSH), is almost instantaneous (0.005s), because instead of comparing every pair of signatures, LSH only looks for band collisions to identify candidate similar documents. This means that LSH avoids comparing all pairs and only checks likely matches.

| **Stage** | **Execution time (seconds)** |
|---|---|
| Shingling | 0.0171 |
| Exact Jaccard computation | 1.0483 |
| MinHash signature generation | 0.7863 |
| Signature similarity comparison | 1.0374 |
| Locality-Sensitive Hashing (LSH) | 0.0050 |
| **Total processing time** | **2.8942** |

Table 3: Execution time of each processing stage for a corpus size of 500 SMS messages.

The total execution time for all steps on a corpus of 500 SMS messages is under 3 seconds which shows that the MinHash + LSH approach is highly efficient. Although, as the dataset size increases, the exact Jaccard computation would take too long while MinHash and LSH would not increase in computation time as much, making it more suitable for larger datasets.

## 4 Conclusion

In this assignment we implemented a program that finds similarities between documents using Shingling, MinHashing, and Locality-Sensitive Hashing to detect duplicate and near-duplicate SMS messages. The results show that although exact Jaccard computation is accurate, it is time consuming for large document collections since it requires comparing every document pair. However, the MinHashing and LSH together further reduced the time it took to compute.

The experiment successfully identified exact duplicate spam messages and messages with very small wording differences. Overall, this study demonstrates that MinHash + LSH can detect textual similarity with high accuracy and drastically reduce the execution time, making it suitable for large datasets.

## References

[1] B. Butcher, "Jaccard Coefficients," University of Notre Dame, Kernel Methods, 2018. [Online]. Available: https://www3.nd.edu/ kogge/courses/cse60742-Fall2018/Public/StudentWork/KernelPaperFinal/jaccard-butcher3.pdf