# Task 1 README

1155192350 Tse Wing Yan

11551954253 Huang Tsz Wing

## Overview

This project identifies pairs of similar articles using **Locality Sensitive Hashing (LSH)** based on cosine similarity. The methodology is designed for efficient processing of large datasets, using MinHash and shingling to reduce dimensionality and group similar items effectively.

## Features

1. **Text Preprocessing**: Removes punctuation, short words, and numbers, converts to lowercase, and splits text into article IDs and content.
2. **Shingling**: Divides articles into overlapping sequences of words (shingles) and hashes them using the MD5 algorithm.
3. **MinHash Signature Matrix**: Reduces the dimensionality of shingles while preserving similarity relationships using random hash functions.
4. **Locality Sensitive Hashing (LSH)**: Divides the MinHash matrix into bands, hashes each band, and groups articles with similar signatures.
5. **Candidate Filtering**: Uses cosine similarity to filter and retain highly similar document pairs with cosine similarity > 0.8.
6. **Output**: Outputs pairs of similar articles into a results file (`result.txt`).

## Requirements

### Python Version

- Python 3.12.7 or higher

## Installation

1. Clone the repository.

2. Install required Python packages:

```
pip install numpy scipy scikit-learn
```

# Usage

## 1. Input File

The program expects an input text file (e.g., `all_articles.txt`) containing articles in the following format:

```
t132 <article_content>
t212 <article_content>
...
```

## 2. Configuration

Set the parameters in the `main()` function within `LSH.py`:

- `n_hashes`: Number of hash functions (default: 100).
- `band_size`: Size of each band (default: 20).
- `shingle_size`: Size of shingles (default: 2).
- `cosine_similarity_threshold`: Similarity threshold (default: 0.8).

## 3. Run the Program

Run the script using:

```
python LSH.p
```

## 4. Output

The output will be saved to a file (`result.txt`) containing pairs of similar articles:

```
t1768 t5248
t3600 t644
...
```

## Results and Performance

- **Output**: The program identified 80 similar article pairs.
- **Runtime**: Average runtime across 5 executions was **4.825 seconds**.

## File Structure

- **LSH.py**: Python implementation of the LSH-based article similarity pipeline.
- **Task_1_Report.pdf**: Detailed report explaining the methodology and results.
- **result.txt**: Output file containing similar article pairs.

## Key Functions in `LSH.py`

- `clean_text(text)`: Cleans and pre-processes the input text.
- `generate_shingles(doc, shingle_size)`: Generates and hashes shingles from a document.
- `compute_minhash_matrix(shingles_in_all_docs, ...)`: Computes the MinHash matrix for documents.
- `get_band_hashes(minhash_matrix, band_size)`: Divides the MinHash matrix into bands and hashes them.
- `filter_similar_pairs(docs, candidate_pairs, ...)`: Filters candidate pairs using cosine similarity.

## Conclusion

This project demonstrates the use of Locality Sensitive Hashing (LSH) in efficiently finding similar articles. By using shingling, MinHash, and cosine similarity, the method achieves both scalability and precision for large datasets.