1. Introduction

This report provides a detailed explanation of the implementation of Task1, which involves finding similar articles using Locality Sensitive Hashing (LSH). The goal is to efficiently identify pairs of articles with a cosine similarity greater than 0.8 from a dataset containing 10,000 articles.

2. Methodology

2.1. Data Preprocessing

The preprocessing step begins by reading the article dataset from 'all_articles.txt', which contains one article per line, with each line containing an article ID followed by its content. The article content is then tokenized into individual words. During this process, many common English words and punctuation marks, which carry very little useful information, are removed. After tokenization, a bag-of-words (BoW) vector is created by counting the frequency of the remaining and useful words. Finally, a structured data frame is formed, containing article IDs paired with their corresponding BoW vectors.

2.2. SimHash Calculation

For the next step, the SimHash algorithm is implemented to generate hash signatures for each article based on the BoW vector. It is important to notice that MinHash techniques, which are typically used for calculating Jaccard similarity, should not be used for this task since it focuses on cosine similarity.

A data structure called 'HashListWeight' is designed to store both the hash vector and its corresponding weight for each unique word in the BoW vector. For each word, a MD5 hash is computed, the result hash value is then converted into a binary vector. In the binary vector, bits set to 1 indicate a positive contribution to the specific feature of the word, while bits set to 0 are transformed into -1 to indicate no contribution. The final SimHash value is derived by aggregating contributions from all words, with each contribution weighted according to the frequency of that word in the article. This allows more frequently occurring words to have a greater impact on the result hash signature. The accumulated contributions are then converted into binary values to represent the final SimHash value.

2.3. LSH Bucketing

To reduce the number of comparisons needed for similarity checks, articles are grouped into buckets based on their SimHash values. A specified number of bits from the end of each SimHash value is used as the bucket key. After several trial and error, it was determined that using a bucket size of 9 bits yields better results compared to starting from the beginning of the SimHash value, which did not provide satisfactory results. This ensure that similar items are mapped to the same bucket with high probability.

2.4. Similarity Computation

For articles within the same bucket, cosine similarity is computed to determine if they are similar. Notice that if there is only one article in the bucket, that bucket can be skipped since no comparisons are needed. The SimHash values of the articles in each bucket are converted into vectors suitable for easier calculation.

3. Experimental Evaluations

The results are shown as follows. It is important to note that the runtime can vary significantly based on the machine used for execution. For instance, running the program on Kaggle result in different performance compared to a local machine due to difference in hardware.

```
[nltk_data]   Package punkt_tab is already up-to-date!
(ftec4005_proj) (base) raymondli@RaymonddeAir FTEC4005_group_11_project_1 % python3 FTEC4005_group_11_project_1.py
Data Processing time: 31.659904291969724
Search time: 23.323098041990306
t797 t3088
t4910 t5780
t269 t8413
t1513 t764
t6205 t4467
t5239 t2001
t969 t6244
t2839 t9303
t104 t4172
t1726 t9170
t3889 t538
t4969 t2390
t2957 t7111
t4591 t1206
t1295 t6680
t6613 t9385
t3495 t1952
t1768 t5248
```

```
Data Processing time: 73.54679008
Search time: 68.67761141300002
t9363 t1012
t4022 t3358
t9724 t8861
t9445 t6370
t6092 t3783
t3889 t538
t269 t8413
t6205 t4467
t980 t2023
t4530 t7907
t1782 t7716
t9549 t1488
t5239 t2001
```

4.Results

The implementation successfully identifies similar article pairs, which are saved in the output file ‘result.txt’.