Data Mining Homework- 1

Finding Similar Items: Textually SimilarDocuments

Group – 7

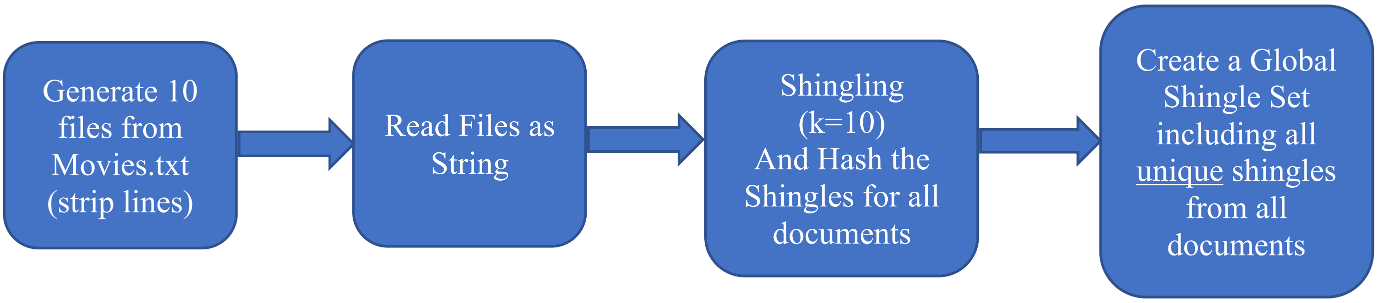
Xing Zeng <xingzeng@kth.se>   Sevket Melih Zenciroglu <smzen@kth.se>

1. Dataset

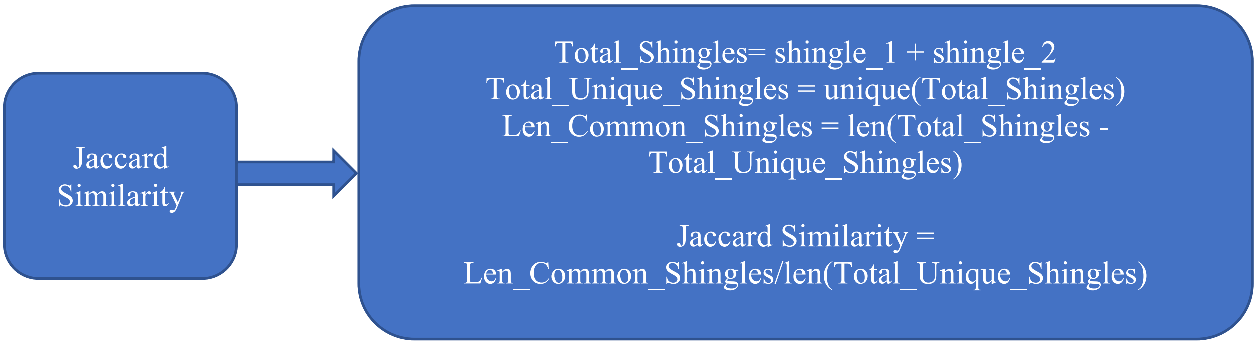
As the dataset we generated files from Amazon movie reviews file [1]. Since the original file was too big (9,33 GB), we generated 10 files by parsing it. Once we did the comparison between the files, there was a big gap in terms of the similarity, so we picked one of the files [2] and removed some text from it and created 2 more files [3, 4] in that way. Then we observed >85% similarity for those new files with the original file.

2. Workflow

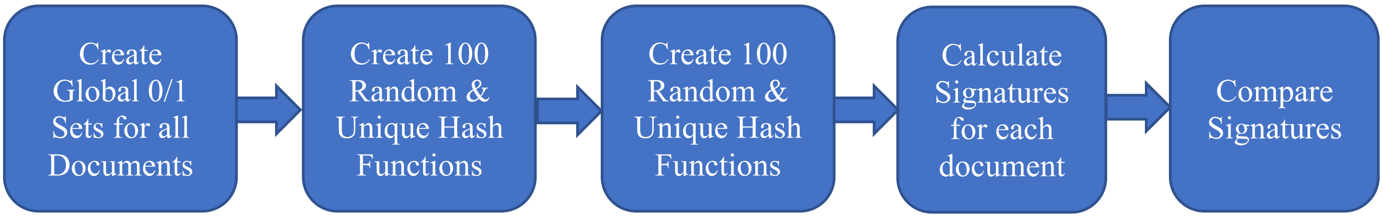
While generating the files, we strip the lines that are read from Movies.txt. So, basically, each file we generated has 1 huge line only.


Generate 10 files from Movies.txt (strip lines)
Read Files as String
Shingling (k=10) And Hash the Shingles for all documents
Create a Global Shingle Set including all unique shingles from all documents

Here, it is possible to calculate the Jaccard similarity for given 2 documents easily:


Jaccard Similarity
Total_Shingles= shingle_1 + shingle_2
Total_Unique_Shingles = unique(Total_Shingles)
Len_Common_Shingles = len(Total_Shingles - Total_Unique_Shingles)
Jaccard Similarity = Len_Common_Shingles/len(Total_Unique_Shingles)

But we will continue applygin the MinHash Signatures by using random Hash Functions in the form of (ax+b)%c where c = # of unique shingles in the global set


Create Global 0/1 Sets for all Documents

Create 100 Random & Unique Hash Functions

Create 100 Random & Unique Hash Functions

Calculate Signatures for each document

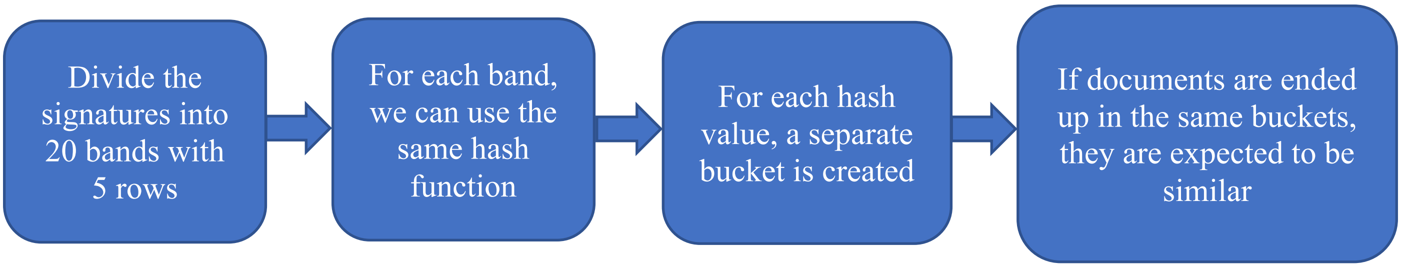
Compare Signatures

We display the Jaccard Similarity Matrix and the Signature Similarity Matrix at this stage:

```
JACCARD SIMILARITY MATRIX:
[[1.    0.03  0.043 0.021 0.045 0.896 0.035 0.934 0.033 0.018 0.017 0.013]
 [0.03  1.    0.032 0.02  0.024 0.03  0.035 0.03  0.019 0.023 0.017 0.013]
 [0.043 0.032 1.    0.02  0.029 0.043 0.041 0.043 0.025 0.025 0.016 0.018]
 [0.021 0.02  0.02  1.    0.025 0.021 0.022 0.021 0.017 0.016 0.018 0.015]
 [0.045 0.024 0.029 0.025 1.    0.045 0.03  0.045 0.029 0.018 0.02  0.014]
 [0.896 0.03  0.043 0.021 0.045 1.    0.037 0.956 0.033 0.02  0.017 0.013]
 [0.035 0.035 0.041 0.022 0.03  0.037 1.    0.035 0.02  0.024 0.019 0.015]
 [0.934 0.03  0.043 0.021 0.045 0.956 0.035 1.    0.033 0.018 0.017 0.013]
 [0.033 0.019 0.025 0.017 0.029 0.033 0.02  0.033 1.    0.026 0.03  0.014]
 [0.018 0.023 0.025 0.016 0.018 0.02  0.024 0.018 0.026 1.    0.035 0.016]
 [0.017 0.017 0.016 0.018 0.02  0.017 0.019 0.017 0.03  0.035 1.    0.015]
 [0.013 0.013 0.018 0.015 0.014 0.013 0.015 0.013 0.014 0.016 0.015 1.   ]]

SIGNATURE SIMILARITY MATRIX:
[[1.   0.07 0.08 0.07 0.06 0.86 0.12 0.92 0.09 0.1  0.1  0.07]
 [0.07 1.   0.13 0.09 0.06 0.07 0.12 0.07 0.05 0.09 0.11 0.13]
 [0.08 0.13 1.   0.12 0.1  0.09 0.14 0.09 0.06 0.09 0.07 0.09]
 [0.07 0.09 0.12 1.   0.09 0.06 0.13 0.07 0.09 0.09 0.14 0.11]
 [0.06 0.06 0.1  0.09 1.   0.07 0.13 0.07 0.1  0.09 0.15 0.16]
 [0.86 0.07 0.09 0.06 0.07 1.   0.13 0.93 0.09 0.1  0.09 0.07]
 [0.12 0.12 0.14 0.13 0.13 0.13 1.   0.12 0.12 0.1  0.15 0.15]
 [0.92 0.07 0.09 0.07 0.07 0.93 0.12 1.   0.09 0.1  0.1  0.07]
 [0.09 0.05 0.06 0.09 0.1  0.09 0.12 0.09 1.   0.12 0.14 0.12]
 [0.1  0.09 0.09 0.09 0.09 0.1  0.1  0.1  0.12 1.   0.19 0.13]
 [0.1  0.11 0.07 0.14 0.15 0.09 0.15 0.1  0.14 0.19 1.   0.17]
 [0.07 0.13 0.09 0.11 0.16 0.07 0.15 0.07 0.12 0.13 0.17 1.  ]]
```

BONUS: LSH - Locality Sensitive Hashing using b = 20, r = 5
Hash function used is the python regular hash(). E.g. hash(str(list_5_rows_of_1_document))

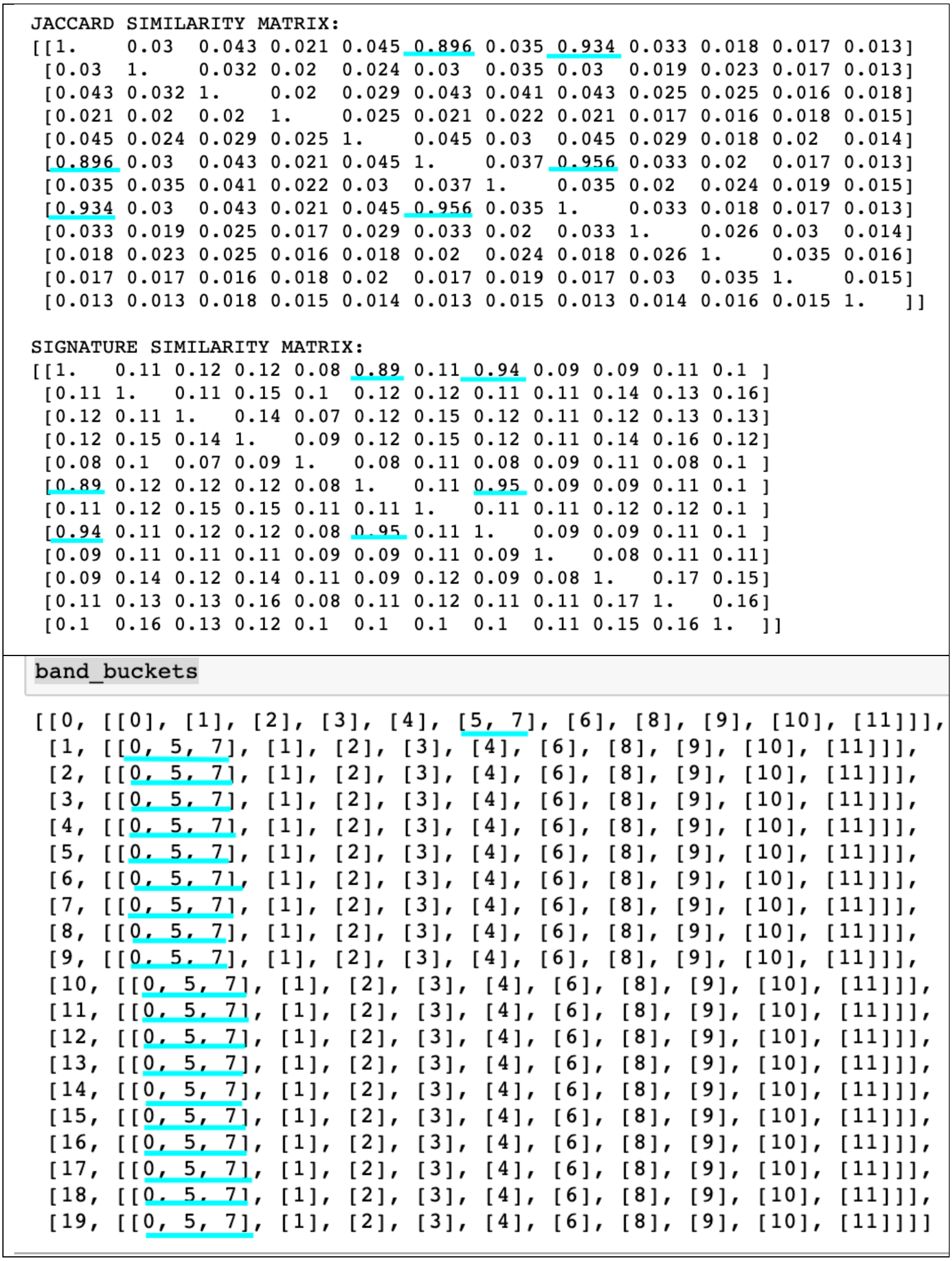

Divide the signatures into 20 bands with 5 rows

For each band, we can use the same hash function

For each hash value, a separate bucket is created

If documents are ended up in the same buckets, they are expected to be similar

The Jaccard similarity, MinHash Signature Comparisons (80%) and LSH Buckets provided the similar results. Converting documents to 0/1 sets is taking too much time (around 4 minutes for 12 documents). Other parts are quite fast, only a few seconds.

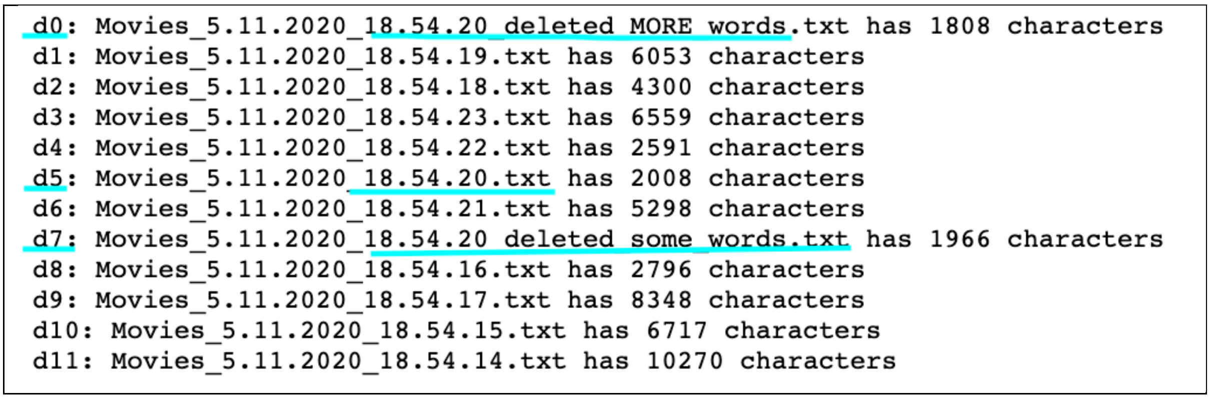


```
JACCARD SIMILARITY MATRIX:
[[1.    0.03  0.043 0.021 0.045 0.896 0.035 0.934 0.033 0.018 0.017 0.013]
 [0.03  1.    0.032 0.02  0.024 0.03  0.035 0.03  0.019 0.023 0.017 0.013]
 [0.043 0.032 1.    0.02  0.029 0.043 0.041 0.043 0.025 0.025 0.016 0.018]
 [0.021 0.02  0.02  1.    0.025 0.021 0.022 0.021 0.017 0.016 0.018 0.015]
 [0.045 0.024 0.029 0.025 1.    0.045 0.03  0.045 0.029 0.018 0.02  0.014]
 [0.896 0.03  0.043 0.021 0.045 1.    0.037 0.956 0.033 0.02  0.017 0.013]
 [0.035 0.035 0.041 0.022 0.03  0.037 1.    0.035 0.02  0.024 0.019 0.015]
 [0.934 0.03  0.043 0.021 0.045 0.956 0.035 1.    0.033 0.018 0.017 0.013]
 [0.033 0.019 0.025 0.017 0.029 0.033 0.02  0.033 1.    0.026 0.03  0.014]
 [0.018 0.023 0.025 0.016 0.018 0.02  0.024 0.018 0.026 1.    0.035 0.016]
 [0.017 0.017 0.016 0.018 0.02  0.017 0.019 0.017 0.03  0.035 1.    0.015]
 [0.013 0.013 0.018 0.015 0.014 0.013 0.015 0.013 0.014 0.016 0.015 1.   ]]

SIGNATURE SIMILARITY MATRIX:
[[1.   0.11 0.12 0.12 0.08 0.89 0.11 0.94 0.09 0.09 0.11 0.1 ]
 [0.11 1.   0.11 0.15 0.1  0.12 0.12 0.11 0.11 0.14 0.13 0.16]
 [0.12 0.11 1.   0.14 0.07 0.12 0.15 0.12 0.11 0.12 0.13 0.13]
 [0.12 0.15 0.14 1.   0.09 0.12 0.15 0.12 0.11 0.14 0.16 0.12]
 [0.08 0.1  0.07 0.09 1.   0.08 0.11 0.08 0.09 0.11 0.08 0.1 ]
 [0.89 0.12 0.12 0.12 0.08 1.   0.11 0.95 0.09 0.09 0.11 0.1 ]
 [0.11 0.12 0.15 0.15 0.11 0.11 1.   0.11 0.11 0.12 0.12 0.1 ]
 [0.94 0.11 0.12 0.12 0.08 0.95 0.11 1.   0.09 0.09 0.11 0.1 ]
 [0.09 0.11 0.11 0.11 0.09 0.09 0.11 0.09 1.   0.08 0.11 0.11]
 [0.09 0.14 0.12 0.14 0.11 0.09 0.12 0.09 0.08 1.   0.17 0.15]
 [0.11 0.13 0.13 0.16 0.08 0.11 0.12 0.11 0.11 0.17 1.   0.16]
 [0.1  0.16 0.13 0.12 0.1  0.1  0.1  0.1  0.11 0.15 0.16 1.  ]]
```

```
band_buckets
```

```
[[0, [[0], [1], [2], [3], [4], [5, 7], [6], [8], [9], [10], [11]]],
 [1, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [2, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [3, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [4, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [5, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [6, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [7, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [8, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [9, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [10, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [11, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [12, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [13, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [14, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [15, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [16, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [17, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [18, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]],
 [19, [[0, 5, 7], [1], [2], [3], [4], [6], [8], [9], [10], [11]]]]
```



```
d0: Movies_5.11.2020_18.54.20_deleted MORE words.txt has 1808 characters
d1: Movies_5.11.2020_18.54.19.txt has 6053 characters
d2: Movies_5.11.2020_18.54.18.txt has 4300 characters
d3: Movies_5.11.2020_18.54.23.txt has 6559 characters
d4: Movies_5.11.2020_18.54.22.txt has 2591 characters
d5: Movies_5.11.2020_18.54.20.txt has 2008 characters
d6: Movies_5.11.2020_18.54.21.txt has 5298 characters
d7: Movies_5.11.2020_18.54.20 deleted some words.txt has 1966 characters
d8: Movies_5.11.2020_18.54.16.txt has 2796 characters
d9: Movies_5.11.2020_18.54.17.txt has 8348 characters
d10: Movies_5.11.2020_18.54.15.txt has 6717 characters
d11: Movies_5.11.2020_18.54.14.txt has 10270 characters
```

3. Run the code:

You can basically run the entire notebook with the default settings on it.
Default settings used:

k_shingles = 10, s = 0.8 (similarity threshold), n_band = 20 (corresponds to b), r = 5
n_hash_functions = 100 (Random Hash Functions for MinHashing Signatures)

1. Generate files by using [5]
2. Run the code for the calculations [6]

References:

[1] https://snap.stanford.edu/data/web-Movies.html

[2] https://drive.google.com/drive/folders/1EG8wFmkHFg6_UZSejfMY75E0LT6dv3us?usp=sharing > Movies_5.11.2020_18.54.20.txt

[3] https://drive.google.com/drive/folders/1EG8wFmkHFg6_UZSejfMY75E0LT6dv3us?usp=sharing > Movies_5.11.2020_18.54.20_deleted_some_words.txt

[4] https://drive.google.com/drive/folders/1EG8wFmkHFg6_UZSejfMY75E0LT6dv3us?usp=sharing > Movies_5.11.2020_18.54.20_deleted_MORE_words.txt

[5] File_Generation_code@Google_Drive

[6] Finding_Similar_Items_code@Google_Drive