1793434 Livia Biggi
1396786 Adriano Fragomeni

DMT Report – Homework 1

Part 1.1

The given data on the Ground Truth and the results of the Search Engines have 1,249 and 44,400 entries, respectively. Although the query IDs are numbered (ending with query 225), they are not representative of the total amount of the queries, as they add up to 222, indicating three missing IDs, i.e. query 31, 119 and 215.

The measures of precisions have provided the following results:

Precision at K:

Search Engine	Mean(P@1)	Mean(P@3)	Mean(P@5)	Mean(P@10)
SE_1	0.03153	0.03003	0.02793	0.02568
SE_2	0.3018	0.2958	0.26306	0.18559
SE_3	0.23874	0.20571	0.18649	0.14324

Mean Reciprocal Rank:

Search Engine	MRR
SE_1	0.08141
SE_2	0.4863
SE_3	0.39512

R-Precision:

Search Engine	Mean (R-Precision Distribution)	min(R-Precision Distribution)	1°_quartile (R-Precision Distribution)	MEDIAN(R-Precision Distribution)	3°_quartile (R-Precision Distribution)	MAX(R-Precision Distribution)
SE_1	0.02256	0	0	0	0	0.66667
SE_2	0.25494	0	0	0.25	0.42857	1
SE_3	0.17933	0	0	0.14286	0.33333	1

Normalised Discounted Cumulative Gain:

Search Engine	Mean(nDCG@1)	Mean(nDCG@3)	Mean(nDCG@5)	Mean(nDCG@10)
SE_1	0.031532	0.030451	0.028749	0.027052
SE_2	0.301802	0.296849	0.27541	0.22221
SE_3	0.238739	0.208655	0.195767	0.165577

The tables above clearly identify the second Search Engine as the best-performing out of the three, as it exhibits the highest results for each measure of precision.

1793434 Livia Biggi
1396786 Adriano Fragomeni

Part 1.2

Although the number of queries (222) is the same as in the previous part, including the missing query IDs (31, 119 and 215), here the length of the Search Engines is unchanged, whilst the length of the Ground Truth is 198, meaning that not only are queries 31, 119 and 215 missing, but the queries from 201 onwards are also missing.

Given that the app provides in output only four results, and that these are displayed in random positions, we reasoned that, in order to assess which Search Engine identifies the most relevant documents, the implementation of the functions we wrote above had to be slightly changed. Since the outputs of the Search Engines are ordered lists, we took the first four elements (i.e. the most relevant results of each SE), and we randomly permuted their orders, such that their positions on the screen do not reflect their true rank.

Hence, out of the measures of precision used in the previous part, the only one that does not take into account the position of the relevant results of the SEs is the Precision at K, which we set equal to 4, to account for the four results displayed. Although we could have chosen to compute it for $K \in \{1, 2, 3, 4\}$, we figured it would be better to use 4 as it considers more elements.

Search Engine	Mean(P@4)
SE_1	0.28409
SE_2	0.20581
SE_3	0.28914

The table above clearly shows that the first and third Search Engines are better performing than the second one, though their values of Precision at K do not differ much.
The most accurate Search Engine turns out to be the third one, as it is the best performing of the three, and it outperforms the others for each value of K.

Although the third Search Engine appears to be the best performing one, it is important to notice that Precision at K is not a very efficient way of quantifying the performance of a given SE. The reason for this is that it fails to account for the importance of the order of the results, meaning that each result will have the same weight, no matter the position, as long as it is within the first K positions. Even though R-Precision (and other measures of precision, such as the NDCG) does a better job in accounting for this problem, it relies on the information on the position of each result in the output ordered list, and was hence pointless to use in this kind of analysis.

1793434 Livia Biggi
1396786 Adriano Fragomeni

Part 2.1

The given dataset comprises 87,041 files, which we parsed and cleaned in order to examine (and tokenise) only the lyrics of the songs, to make sure they actually contain any text. We found 1,452 empty songs, that were thus excluded from our analysis.

In order to find the values of the rows and the bands to use when implementing the Java command that calculated the LSH and Min Hashing, we took into account the given constraints, and we solved the following system of an equation and an inequality:

$$\begin{cases} b \cdot r = k \\ 1-(1-s^r)^b \geq p \end{cases}$$

Where $k = 300$ (number of hash functions), $s = 0.85$ (the Jaccard Similarity), and $p = 0.97$ (the probability of finding near duplicate candidates with a Jaccard Similarity s)

The system found several values for r and b, so we looked for the combination of the two that maximises the approximation to the threshold, in order to limit the number of false positives:

$$\left(\frac{1}{b}\right)^{\frac{1}{r}}$$

The resulting values that satisfy the three constraints (including the value of the Jaccard Similarity and that of the probability), are 12 and 25 for rows and bands respectively.

The resulting number of False Positives of our analysis is 202, that of the Near Duplicate Candidates is 2,846, and the number of real Near Duplicates is 2,644.

For a full list of each of the three categories, refer to our code.

The following is the command that we used in the Java command line when implementing the LSH and Min-Hashing.

```
java -Xmx1G tools.NearDuplicatesDetector lsh_plus_min_hashing 0.85 12 25
.\input_data\300_hash_functions_file.tsv .\input_data\all_lyrics.tsv
.\output_data\near_duplicate_candidates.tsv
```

1793434 Livia Biggi
1396786 Adriano Fragomeni

Part 2.2 (a)

In this part we were given a tsv file containing 50 Min-Hash sketches, and we were asked to calculate the length of the ith sketch of the matrix.

Since the available data were that of the sketches and the size of the Universe, we were able to derive the length of the ith set by computing the Jaccard Similarity between each sketch and the sketch of the Universe, which we reasoned would be composed of 0s (i.e. the first relevant position of each element). Given that each set is contained in the Universe, the intersection of the ith set with the Universe (i.e. the numerator of the Jaccard Similarity) is just the set itself, while the union (i.e. the denominator of the JS), is the cardinality of the Universe.

In order to calculate the length of the ith set, we computed the inverse of the Jaccard Similarity formula and found that the cardinality of the ith set is equal to the cardinality of the Universe, multiplied by the Jaccard Similarity.

$$JS = \frac{A \cap B}{A \cup B} = \frac{|setX|}{|U|} \qquad\qquad |setX| = JS \cdot |U|$$

Even though we could not compute the true value of the Jaccard Similarity, we approximated it by referring to the concept of Min-Hash sketching, which finds the minimum values in common over the length of the sketches, i.e. the number of permutations of the ith set. Once we obtained the 50 values of the Jaccard Similarity between each sketch and the sketch of the Universe, we substituted them in the formula above and found the length of each set.

Part 2.2 (b)

In order to estimate the size of the unions of different sketches, we referred back to the tsv file containing the Min-Hash sketches used in the previous part.
For each element in a given union set, we checked its min-hash sketch and computed the sketch of the union by taking the minimum of the values in each position, for all vectors of sketches in the union set.

After obtaining the sketch of each union set, we applied a function that estimated its length in the same way as in the previous part, that is, by multiplying it by the size of the Universe.

Since the size of the Union is such a large number (over a billion in length), the sizes of the sets (calculated in Part 2.2 (a)) and the sizes of the union sets will also be quite significant.