Locality Sensitive Hashing

CS69201: Computing Lab-1
Take Home Assignment
August 7, 2024

========== Instructions==========

1. In the case of user input assume only valid values will be passed as input.
2. You can use C or C++ as the programming language. **However, you are not allowed to use any STL libraries in C++**
3. Regarding Submission: create a C file with the name <rollno>_LSH.c. Create a zip file containing the C file and the input and output files with the name <rollno>_LSH.zip and submit it to Moodle. For example, if your roll number is 24CS60R15, then your file name will be 24CS60R15_LSH.c and your zip file name will be 24CS60R15_LSH.zip.
4. **Inputs should be taken from a file, and outputs should be written to a file.**

==========

You will be provided with 10 sentences. Your task is to find the edit distance between the sentences and the cosine similarity between the hashes(dense vectors of the sentences) for each pair of sentences. You need to use the steps mentioned in the LSH ppt to find the hashes of the sentences. Use k=3 for making shingles. Size of dense vector=10.

Input sentences:

1. There is a cat in the room.
2. The cat in the room is playing.
3. Cat and dog are playing.
4. There is a dog in the park.
5. Children are playing in the park.
6. Hari loves playing with dogs.
7. Hari has a dog and a cat.
8. The dog is playing with a ball.
9. The Lion roars.
10. The dog barks.

Note: You can ignore the case while calculating the edit distance (ie ‘D’=’d’).
Edit distance: Given two strings ‘str1’ and ‘str2’ of length M and N respectively and below operations that can be performed on str1. The minimum number of edits (operations) to convert ‘str1’ into ‘str2’ is the edit distance between the two strings.

1. Operation 1 (INSERT): Insert any character before or after any index of str1
2. Operation 2 (REMOVE): Remove a character of str1
3. Operation 3 (REPLACE): Replace a character at any index of str1 with some other Character.

```
EDITDISTANCE(s1, s2)
 1  int m[i, j] = 0
 2  for i ← 1 to |s1|
 3  do m[i, 0] = i
 4  for j ← 1 to |s2|
 5  do m[0, j] = j
 6  for i ← 1 to |s1|
 7  do for j ← 1 to |s2|
 8     do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
 9                      m[i − 1, j] + 1,
10                      m[i, j − 1] + 1}
11  return m[|s1|, |s2|]
```

► **Figure 3.5** Dynamic programming algorithm for computing the edit distance between strings s_1 and s_2.

Cosine Similarity:
This is a metric helpful in determining how similar the sentences are irrespective of their size. In cosine similarity, data objects are treated as vectors. The formula to find the cosine similarity between two vectors is

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$