NAME – RISHABH MALIK

ROLL NO. - 2020201074

Given – Index of vocabulary, Query entered by user

There are 2 basic approaches -
1. Isolated-term correction – Correcting each word in the query string individually irrespective of context of query
2. Context-sensitive correction – Correcting the query by considering the context of entire words together

Requirements -
- Measure of difference or distance between 2 words, a.k.a., **Edit Distance**
- An approach to shortlist a subset of words from the Index (vocabulary) which are potential corrections for the query. This will save time in getting closest similar words / corrections as we will have to make less comparisons instead of checking with entire index.

Calculating Edit distance between 2 strings/words
Edit distance is the minimum numbers od letters to be inserted to deleted to convert word A to word B.
INPUT - words A (length n), words B (length m)
Time Complexity – O(nm)
Algo – This can be done using dynamic programming by making a 2D array(say X) where X[i][j] stores the edit distance between A(till ith index) and B(till jth index).

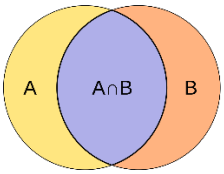
Function Edit_Distance(A, B):

1. **Initialize** matrix X[n][m] with all 0
2. **FOR** i = 0 to length(A)-1 :
   DO X[i][0] = i
3. **FOR** j = 0 to length(B)-1 :
   DO X[0][j] = j
4. **FOR** i = 0 to length(A)-1 :
   DO FOR j = 0 to length(B)-1 :
   DO tmp = 0 if A[i]==B[j] else 1
   X[i][j] = MIN (X[i-1][j-1] + tmp,
   X[i-1][j] + 1,
   X[i][j-1] + 1)
5. **RETURN** X[n-1][m-1]

Now that we have approach to get distance between 2 words, we need to get subset from index (potential correct words) with whom to check this distance from input query word.

Getting Potential Corrections Subset using k-grams
TO KNOW – Jaccard coefficient
For 2 sets A and B, Jaccard coefficient = $|A \cap B| / |A \cup B|$
This enables to get maximum overlap between the 2 sets. See the figure below to visualize how this gets maximum overlap. The larger the overlapping region in proportionate to the total region, greater the chances of both sets being similar are. The goes for words as well. Consider the sets of k-

grams of each word.



Algorithm :

1. K = Number of grams considered for finding overlaps

2. N = Number of words in index
Q = input query word
Th = Threshold for considering a word in index which exceeds jaccard index threshold

3. A = SET (K-grams of Q)
CloseWords = {} // Set of all words which exceeds jaccard index threshold with input query

4. FOR i = 0 **to** N-1 :
DO
B = SET (K-grams of Index[i])
Jaccard_index = | A ∩ B | / | A ∪ B |
IF Jaccard_index >= Th **:**
THEN
CloseWords.insert(Index[i])

5. Shortest_distance = INT_MAX

6. FOR i = 0 **to** Length(CloseWords)**:**
DO
dist = Edit_Distance (Q, CloseWords[i])
IF dist < Shortest_distance **:**
THEN
Shortest_distance = dist
Correct_word = CloseWords[i]

7. OUTPUT : Correct_word

NOTE : In case if multiple words with equal shortest distance exist then we output the word which has been mostly searched by pther users