Spell Checker

Aditya Kumar

March 20, 2019

Task is to design a spell checker which tries to provide most likely correct spelling of a given word. Suppose user has typed date then To be more precise look for words that are 1 or 2 edit distance away. How we will know the correctness of the spell checker? Obviously there is no definite way so we can take here is the probabilistic approach to evaluate which word can be a potential answer.

1. **Naive Approach:** Take all unique words in dictionary and get edit distance of each word. But complexity of this approach will be very much. To be very precise, for each word in corpus we need to calculate edit distance and then need to parse through all distances to show words that are 1 or 2 edit distance away. Algorithmic complexity for edit distance algorithm for word of length m and n is $O(n * m)$.

$$d_{[i][j]} = \begin{cases} d_{[i-1][j-1]} & \text{for } a_{[j]} = b_{[i]} \\ min\begin{cases} d_{[i-1][j]} \\ d_{[i][j-1]} \\ d_{[i-1][j-1]} \end{cases} & \text{for } a_{[j]} \neq b_{[i]} \end{cases} \tag{1}$$

```
def edit_distance(source, target):
    """
    Function to calculate edit distance between two strings using
↪   replace, delete and transform operations
    :param source: Source string
    :param target: Target string
    :return: Integer denoting minimum distance
```

```
    """
    dp = [[0 for _ in range(len(source)+1)] for _ in
    ↪  range(len(target)+1)]
    for i in range(len(source)+1):
        dp[0][i]=i
    for j in range(len(target)+1):
        dp[j][0]=j
    for i in range(1, len(target)+1):
        for j in range(1, len(source)+1):
            if source[j-1]==target[i-1]:
                dp[i][j]=dp[i-1][j-1]
            else:
                dp[i][j] = min(dp[i][j-1], dp[i-1][j],
                ↪  dp[i-1][j-1])+1
    return dp[-1][-1]
```

Now lets suppose there are K words in dictionary, then we need to find edit distance with each word, which result into $O(K * n * m)$.

Also then there will be scan for finding edit distance of either 1 or 2, which will account for $O(K)$.

So final complexity of this method will result in $O(K) + O(K * n * m)$.

2. **Probabilistic Approach:** In this approach, we can take use of large corpus which will give the dictionary and also the frequency of each word. So, we can make use of the information about occurrence of each word provided in recommending correct spelling.

   (a) First generate all words which are 1 and 2 edit distance away from given word.

   (b) Check in dictionary that generated words are present. And suggest words with maximum frequency.

   Complexity Analysis: First step generates words that are 1 or 2 edit distance away from given word using delete, update, insert and transpose operation.

References:

1. Spell Checker by Norvig, https://norvig.com/spell-correct.html

2. Using the Web for Language Independent Spellchecking and Autocorrection

3. https://people.cs.umass.edu/ brenocon/inlp2015/12-editdist.pdf