Introduction

We want to use the concept of “similarity" versus “edit distance". This is a more general way to write these problems, and will allow us to easily write extensions to the code base itself.

We define some terms:

Definition (Alphabet) Let Σ' be alphabet (including space ‘_’)
Definition (Score) Let $x,\ y\ \in \Sigma'$ be two characters from the alphabet, then $s(x, y)$ is the score for aligning x and y

With these two terms, we can define the total "cost" or "value" of aligning two strings:

Definition (Total Alignment Value) Fix an alignment of strings S_1 and S_2. Let S_1' and S_2' be strings that are the same length constructed by inserting all the spaces ’_’, meaing $|S_1| = |S_2| = L$. Then the total alignment value, $\bar{A}$, is the sum of scores:

$$\bar{A} = \sum_{i=1}^{L} s\big(S_1'(i), S_2'(i)\big)$$

Definition (Similarity) Given pairwise scoring function (or matrix) s over vocabulary Σ', similarity of two strings S_1 and S_2 is alignment A^* that maximizes total alignment value

Dynamic Programming Problem

Definition (Value Function) $V(i, j)$ is the value of the optimal alignment of prefixes $S_1[1, .., i]$ and $S_2[1, .., j]$.[1]

Let’s write down the problem, including the initial conditions. I will then show how to map this to the Wagner Fischer algorithm (or an edit distance formulation).

[1]Note, prefix is just a substring, starting at the beginning of a string going to some index, say i. This is in contrast to suffixes which are substrings starting from the last character going to some index, say $L - i$. These terms will pop up every so often.

Initial Conditions

$$V(0,j) = \sum_{1 \leq k \leq j} s('\_', S_2(k))$$
$$V(i,0) = \sum_{1 \leq k \leq i} s(S_1(k), '\_')$$

Thus, we sum up the costs of inserting spaces (or deleting characters).

Recurrence Relation

$$V(i,j) = \max \Big\{ V(i-1,j) + s(S_1(i), '\_') \\ , \ V(i,j-1) + s('\_', S_2(j)) \\ , \ V(i-1,j-1) + s(S_1(i), S_2(j)) \Big\}$$

Mapping to Edit Distance

Definition (Edit Distance) Let $\bar{V}(i,j)$ be the edit distance for prefixes $S_1[1,..,i]$ and $S_2[1,..,j]$. We defined the edit distance as the following dynamic program:

Initial Conditions

$$\bar{V}(0,j) = j$$
$$\bar{V}(i,0) = i$$

Recurrence Relation

$$\bar{V}(i,j) = \min \Big\{ \bar{V}(i-1,j) + 1 \\ , \ \bar{V}(i,j-1) + 1 \\ , \ \bar{V}(i-1,j-1) + t(S_1(i), S_2(j)) \Big\}$$

It turns out these are equivalent problems with the right Score functions $s(x,y)$. If we use the negative of the costs of our edit distance then we are done:

$$s(S_1(i), '\_') = 1$$
$$s('\_', S_2(j)) = 1$$
$$s(S_1(i), S_2(j)) = \begin{cases} 2 \text{ if } S_1(i) \neq S_2(j)) \\ 0 \text{ otherwise} \end{cases}$$

With this socre function, then we have the following:

$$\underbrace{V(i,j)}_{\text{Similarity}} = -\underbrace{\bar{V}(i,j)}_{\text{Edit}}$$

Extension : End of Space Variant

Let the cost of inserting spaces at beginning and ending of strings be costless. This leads to two changes:

- Base conditions are now value 0: $V(i,0) = V(0,j) = 0$
- $V(n,m)$ is not value in last cell (n,m) but maximum value in last row/column

Extension : Local Alignment