MARASIGAN, SANTIAGO

Minimum Edit Distance - Write Up

Source	Target	Distance	Alignment
naruto's son	borutos's dad	10	naruto's son **SSMMMMMMMSSS** boruto's dad
kumakain	kumain	2	kumakain **MMMDDMMM** kum--ain
levinstien	levenshtein	5	levins-tie-n **MMMSMMIMDMIM** levensht-ein
leviathan	levenshtein	10	lev--iathan **MMMIISSMSSM** levenshtein
ATGCATCCCATGAC	TCTATATCCG T	11	ATGC-ATCCCAT--GAC **DMDMIMMDDDMMIIMDS** -T-CTAT---ATCCG-T
AGGCTATCACCTGA CCTCCAGGCCGATG CCCACCTGG	TAGCTATCAC GACCGCGGTC GATTTGCCCG ACGGTCC	18	-AGGCTATCACCTGACCTCCAGGC CGA--TGCCC-ACCTGG--- **IMDMMMMMMMDMDMMMMDSMDMMS** **MMMIIMMMMMIMDMDMMIII** TA-GCTATCA-C-GACC-GC-GGT CGATTTGCCCGA-C-GGTCC

S: Substitution, M: Match, I: Insertion, D: Deletion

Algorithm

Our algorithm is an application of the existing algorithm to calculate for the Levenshtein distance between two Strings. This results in the generation of a matrix showing the Levenshtein distance of each substring. The value in the corner of the matrix that encompasses both Strings is used to calculate the Minimum Edit Distance. Insertions and Deletions count as one (1) operation, and since substitution could be seen as a Deletion and then an Insertion, it counts as two (2) operations. To show the backtrace, we executed the operations to determine the Levenshtein distance again, but in reverse. We then stored the operation in an array to show the sequence of operations.

Modifications

We first ran the backtrace algorithm with no bias towards any operation, meaning that none of the operations were prioritized and was taken at face value. The problem we found was that the distance that would be returned by the backtrace was unnecessarily long, therefore not being the minimum edit distance. After our encounter with this problem, we found that it would be best to use an algorithm which provided prioritization for each operation. Through implementation of this algorithm, we reached the correct distance with all cases rather than some or most.

Biases

The algorithm we used prioritizes substitutions and matches. This is because a substitution is essentially an insertion and a deletion, meaning that two operations are being executed through a single move rather than executing an insertion then a deletion or vice-versa becoming two moves.

Experimentation

We also tried making Substitutions count as one (1) operation instead of two (2), and results were quite surprising. Most of the test cases dropped in number of operations, but they were not as aligned as they were previously. Most of the words with a small amount of operations actually didn't change much, but the ones with a lot of operations such as the last example had drastic changes.