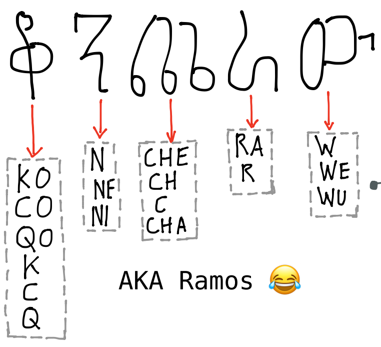
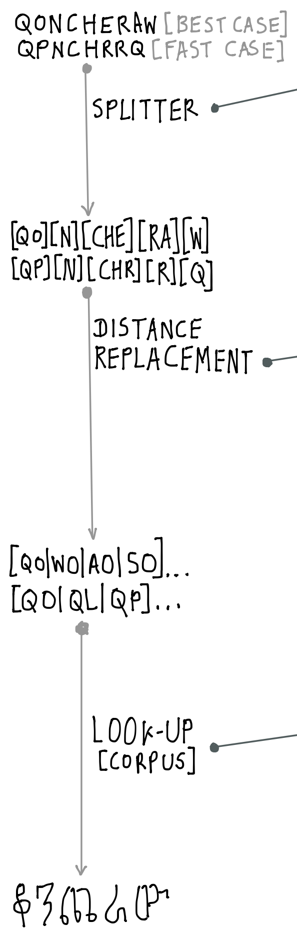


AKA Ramos 😂

This is the initial phase where using the corpus each KAL is mapped Fidel-by-Fidel bases. **But** this doesn't mean the user has to type one of the matches in order to get a match. Huge task awaits on the look-up algorithm.

Each KAL is mapped to all possible *sounds*; sticking with the main motto of eKeyboard ***if you can sound it out, you can type it.***



The splitter will chop each word into a group of [1,2,3]. This will result into huge list to look up but the indexing will allow us to handle the load.

Distance replacement will be applied on each group before the final matching. Using Levensthtein distance wouldn't be fair as 'qo' -> 'qp' will have the same edit distance as 'qo' -> 'qg' which shouldn't be in our case as **speed** is the name of the game. Distances computations must account for the proximity of a character.

Investigate (ANC) on available options, such as Smith-Waterman algorithm.

Whole process should be iterative, else waiting for each possible match will cost time. Going through each transformer function until (if) a threshold is hit will allow us to hit the goal of having responses under the 25ms mark.

The flow can be done on either direction. Tho the reverse lookup seems to be more efficient as it will have significantly smaller lookup.