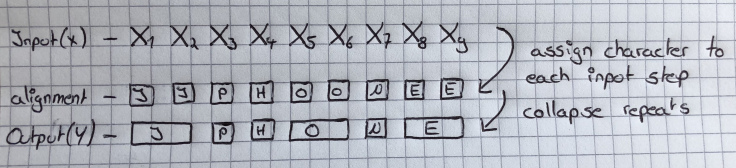
**Connectionist Temporal Classification:**

Connectionist Temporal Classification (CTC) is a valuable operation to tackle sequence problems where timing is variable, like Speech and Handwriting recognition. Without CTC, you would need an aligned dataset, which in the case of Speech Recognition, would mean that every character of a transcription, would need to be aligned to its exact location in the audio file. Therefore, CTC makes training such a system a lot easier.

CTC is “alignment-free“. It works by **summing over the probability of all possible alignments between the input and the label**.  To understand that, take a look at this naive approach:



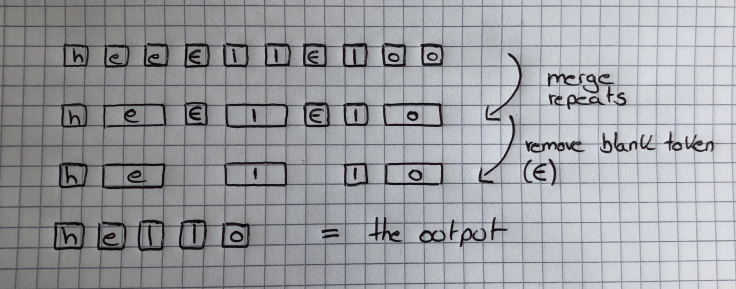
Here we have an input of size 9 and the correct transcription of it is „Iphone“. We force our system to assign an output character to each input step and then we collapse the repeats, which results in the output. But this approach has two problems:

It does not make sense to force every input step to be aligned to some output, because we also need to account for silence within the input. There is no way to produce words as output that have two characters in a row, like a word “Kaggle“. If we use this approach, we could only produce “Kagle” as output.

Blank Token

There is a way around that, called the “blank token“. It does not mean anything and it simply gets removed before the final output is produced. Now, the algorithm can assign a character or the blank token to each input step. There is one rule: To allow double ‘’characters in the output, a blank token must be between them. With this simple rule, we can also produce output with two characters in a row.

Here is an illustration of how this works:



1) The CTC network assigns the character with the highest probability (or no character) to every input sequence.

2) Repeats without a blank token in between get merged.

3) And lastly, the blank token gets removed.

The CTC network can then give you the probability of a label for a given input, by summing over all the probabilities of the character for each time-step.

**CTC LOSS:**

Minimize the log probability of property work

CTC loss = -ln(pw)

**CTC LOSS Components:**

* **Forward variable :** Calculates the total probability from the first timestep till timestep **t** and token **s**

**αt (s)**

* **Backward variable:** Calculate the total probability from timestep **t** and token **s** till last timestep

**βt (s)**

**Conclusion :**

* **CTC** Loss allows training models on sequences whose number of inputs is different than the number of labels.
* It makes use of dynamic programming to calculate path probabilities efficiently.
* **CTC** treats every timestep independently.