**[All about RapidFuzz — String Similarity and Matching](https://medium.com/mlearning-ai/all-about-rapidfuzz-string-similarity-and-matching-cd26fdc963d8)**

What’s all the fuzz about? Another Fuzzy String Matching Technique

[Parthvi Shah](https://medium.com/@shahparthvi22?source=post_page-----cd26fdc963d8--------------------------------)

RapidFuzz is another string matching library which calculate the differences between strings & it also has a lot more to offer. This is mostly written in C++ to make the string matching work ever faster. There are 3 main modules:

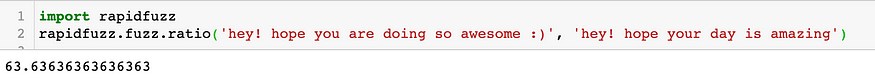
1. Fuzz Module
2. String Metric Module
3. Process Module

GitHub Link: <https://github.com/maxbachmann/RapidFuzz>

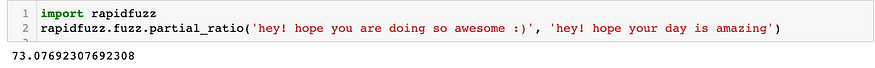
Fuzz Module

It offers all the unique ways in which you can compare two pieces of strings. There are a total of 10 ratios through which you can compare two strings. Some of them are:

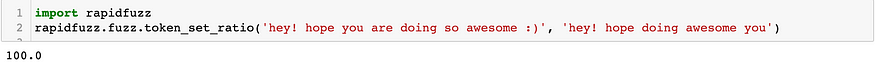
1. **Ratio:**It calculates the normalized distance.



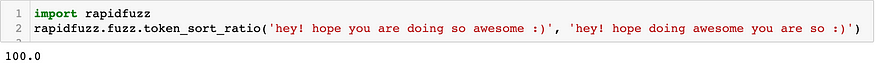
**2.** **Partial Ratio:**It finds the ratio similarity measure between the shorter string and every substring of length m of the longer string, and returns the maximum of those similarity measures. Basically, it searches for the optimal alignment of the shorter string in the longer string and returns the fuzz.ratio for this alignment.



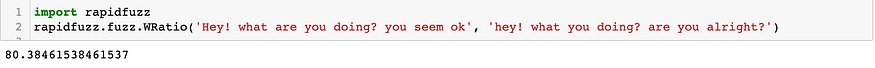
**3.** **Token Set Ratio:** Compares the words in the strings based on unique and common words between them. It takes a set of all the tokens in the string and then compares it. If String A has all the elements of String B, the ratio will be 100. But it’s not true the other way around.



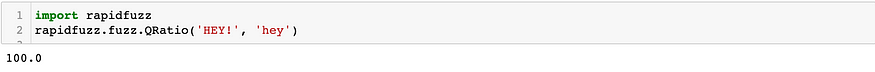
**4. Token Sort Ratio:**Sorts the words in the strings and calculates the fuzz.ratio between them



**5.** **W Ratio**: Calculates a weighted ratio based on the other ratio algorithms. It depends on the number of times a word occurs, order of the tokens, etc.



**6. Q Ratio:**It is very similar to fuzzy.ratio except this pre-processes the string before calculating the distance.



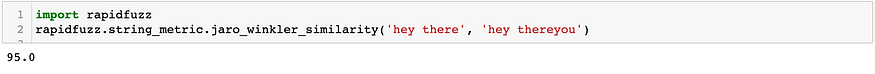
**String Metric Module:**

This module is responsible for the type of various edit distance, more similar the metric, greater the edit distance between two strings. The top similarity metrics are:

1. **Jaro Similarity:** The similarity between two strings is normalized in this case. 0 being not similar at all & 1 being most similar. It depends on the intersection of characters, length of the two strings and the number of transpositions to match the string.



**2. Jaro-Winkler Similarity:** It is almost the same as jaro similarity but in this case, it has a scaling factor p, which makes it more accurate.



**3. Hamming Distance:**It describes the minimum amount of substitutions required to transform s1 into s2. For hamming distance comparison, two strings need to be of the same length.



Which similarity metric or fuzzy ratio to use, really depends on the use-case of your problem. It depends on how strictly similar your strings have to be. If you are still confused, just be sure to start somewhere and iterate unless one metric works your your use case.

Thank you!

**References:**

1. <https://github.com/maxbachmann/RapidFuzz>