When to use which fuzz function to compare 2 strings

Ask Question

Asked 4 years, 8 months ago Active 11 months ago Viewed 23k times

32

25

I am learning `fuzzywuzzy` in Python.

I understand the concept of `fuzz.ratio`, `fuzz.partial_ratio`, `fuzz.token_sort_ratio` and `fuzz.token_set_ratio`. My question is when to use which function?

- Do I check the 2 strings' length first, say if not similar, then rule out `fuzz.partial_ratio`?
- If the 2 strings' length are similar, I'll use `fuzz.token_sort_ratio`?
- Should I always use `fuzz.token_set_ratio`?

Anyone knows what criteria SeatGeek uses?

I am trying to build a real estate website, thinking to use `fuzzywuzzy` to compare addresses.

python string-comparison fuzzywuzzy

share edit follow

edited May 11 '19 at 22:48
daaawx
1,383 1 10 12

asked Aug 4 '15 at 10:21
Pot
353 1 4 8

add a comment

2 Answers

Active Oldest Votes

69

Great question.

I'm an engineer at SeatGeek, so I think I can help here. We have a great blog post that explains the differences quite well, but I can summarize and offer some insight into how we use the different types.

Overview

Under the hood each of the four methods calculate the edit distance between some ordering of the tokens in both input strings. This is done using the `difflib.ratio` function which will:

> Return a measure of the sequences' similarity (float in [0,1]).
>
> Where T is the total number of elements in both sequences, and M is the number of matches, this is 2.0*M / T. Note that this is 1 if the sequences are identical, and 0 if they have nothing in common.

The four fuzzywuzzy methods call `difflib.ratio` on different combinations of the input strings.

fuzz.ratio

Simple. Just calls `difflib.ratio` on the two input strings (code).

```
fuzz.ratio("NEW YORK METS", "NEW YORK MEATS")
> 96
```

fuzz.partial_ratio

Attempts to account for partial string matches better. Calls `ratio` using the shortest string (length n) against all n-length substrings of the larger string and returns the highest score (code).

Notice here that "YANKEES" is the shortest string (length 7), and we run the ratio with "YANKEES" against all substrings of length 7 of "NEW YORK YANKEES" (which would include checking against "YANKEES", a 100% match):

```
fuzz.ratio("YANKEES", "NEW YORK YANKEES")
> 60
fuzz.partial_ratio("YANKEES", "NEW YORK YANKEES")
> 100
```

fuzz.token_sort_ratio

Attempts to account for similar strings out of order. Calls `ratio` on both strings after sorting the tokens in each string (code). Notice here `fuzz.ratio` and `fuzz.partial_ratio` both fail, but once you sort the tokens it's a 100% match:

```
fuzz.ratio("New York Mets vs Atlanta Braves", "Atlanta Braves vs New York Mets")
> 45
fuzz.partial_ratio("New York Mets vs Atlanta Braves", "Atlanta Braves vs New York Mets")
> 45
fuzz.token_sort_ratio("New York Mets vs Atlanta Braves", "Atlanta Braves vs New York Met
> 100
```

fuzz.token_set_ratio

Attempts to rule out differences in the strings. Calls ratio on three particular substring sets and returns the max (code):

1. intersection-only and the intersection with remainder of string one
2. intersection-only and the intersection with remainder of string two
3. intersection with remainder of one and intersection with remainder of two

Notice that by splitting up the intersection and remainders of the two strings, we're accounting for both how similar and different the two strings are:

```
fuzz.ratio("mariners vs angels", "los angeles angels of anaheim at seattle mariners")
> 36
fuzz.partial_ratio("mariners vs angels", "los angeles angels of anaheim at seattle marin
> 61
fuzz.token_sort_ratio("mariners vs angels", "los angeles angels of anaheim at seattle ma
> 51
fuzz.token_set_ratio("mariners vs angels", "los angeles angels of anaheim at seattle mar
> 91
```

Application

This is where the magic happens. At SeatGeek, essentially we create a vector score with each ratio for each data point (venue, event name, etc) and use that to inform programatic decisions of similarity that are specific to our problem domain.

That being said, truth by told it doesn't sound like FuzzyWuzzy is useful for your use case. It will be tremendiously bad at determining if two addresses are similar. Consider two possible addresses for SeatGeek HQ: "235 Park Ave Floor 12" and "235 Park Ave S. Floor 12":

```
fuzz.ratio("235 Park Ave Floor 12", "235 Park Ave S. Floor 12")
> 93
fuzz.partial_ratio("235 Park Ave Floor 12", "235 Park Ave S. Floor 12")
> 85
fuzz.token_sort_ratio("235 Park Ave Floor 12", "235 Park Ave S. Floor 12")
> 95
fuzz.token_set_ratio("235 Park Ave Floor 12", "235 Park Ave S. Floor 12")
> 100
```

FuzzyWuzzy gives these strings a high match score, but one address is our actual office near Union Square and the other is on the other side of Grand Central.

For your problem you would be better to use the Google Geocoding API.

share edit follow

edited May 20 '18 at 18:27
Dennis Golomazov
10.6k • 4 • 61 • 60

answered Aug 5 '15 at 5:08
Rick Hanlon II
14.7k • 6 • 37 • 51

Hi Rick, thanks for your willingness to help. I got the point about using Google Geocoding API, I'll spend more time on it. Since I got this far learning seatGeek, I want to get a better understanding of the "Application" where the magic happens. Does seatGeek system keep the upcoming events, venues, performers in separate lists (in python) / array? So when I type giants, it checks against these lists, then performs all 4 ratio function calls, it rules out those with low scores, keep those high scores items on dropdown box. You would have preset the low scrore and high score thresholds? – Pot Aug 6 '15 at 8:14

We create a canonical source of each event, venue, and performer and compare new inputs against the canonical sources to pair them so that by the time the user begins searching for "giants" we do a search on the canonical source, rather than all the potential inputs we ingest. I hope that makes it more clear. – Rick Hanlon II Aug 6 '15 at 18:02

If I understand it right, you standardise and normalise those events, performers and venues. Any source you find will map to these canonical lists, unless you don't find a good match in your canonical lists, then

decorators?

3599 Does Python have a string 'contains' substring method?

Hot Network Questions

The ultimate inequality challenge
Can you use a phone as grey/white card?
Russian word for female teacher
Why do some viruses cease being a problem even though no vaccine or cure is found?
How to make an infinite blend in illustrator?
Intercept the missile
Intuitive/combinatorial explanation of Delannoy summand
What "unusual" syntax assembly languages are/were there?
Why is a doubling of frequency called an octave?
Layering unicast addressing over multicasts
LetsEncrypt Certificate for Internal Site
How can I explain a device that blurs vision within a localized area?
How can I frighten my enemies besides using spells?
Why do we learn the genitive singular of each Latin noun?
Breath Marks (different symbols)
Should I convert my family pictures from JPEG to PNG?
Missed April 15th deadline to accept grad school offer. What can I do?
what is this weird mechanical thing I found in the woods?
Unexplained Error "StringJoin: String expected at position 1 in #1<>
Why is "asdf".replace(/.*/g, "x") == "xx"?
What is the meaning of this airport marking?
pure Python Bézier curve implementation
Why would an adventurer use a sword frog?
Why is this airplane flying in circles randomly?

Question feed

you find will map to these canonical lists, unless you don't find a good match in your canonical lists, then you will create a new entry and store them. So when I type giants, seatgeek just search against these canonical lists. No hard feeling if it's too sensitive to share, I'm not trying to build a seatgeek in asia, just out of interest. :) Thanks so much for your insight, I have learned alot chatting with you. I'm sure this fuzzywuzzy concept will help my development one day. – Pot Aug 7 '15 at 5:52

I'm surprised you're not training a NLP model, it seems like it would be easy and perform way better. – pguardiario Aug 15 '19 at 2:41

add a comment

14

As of June 2017, `fuzzywuzzy` also includes some other comparison functions. Here is an overview of the ones missing from the accepted answer (taken from the source code):

fuzz.partial_token_sort_ratio

Same algorithm as in `token_sort_ratio`, but instead of applying `ratio` after sorting the tokens, uses `partial_ratio`.

```
fuzz.token_sort_ratio("New York Mets vs Braves", "Atlanta Braves vs New York Mets")
> 85
fuzz.partial_token_sort_ratio("New York Mets vs Braves", "Atlanta Braves vs New York Met
> 100
fuzz.token_sort_ratio("React.js framework", "React.js")
> 62
fuzz.partial_token_sort_ratio("React.js framework", "React.js")
> 100
```

fuzz.partial_token_set_ratio

Same algorithm as in `token_set_ratio`, but instead of applying `ratio` to the sets of tokens, uses `partial_ratio`.

```
fuzz.token_set_ratio("New York Mets vs Braves", "Atlanta vs New York Mets")
> 82
fuzz.partial_token_set_ratio("New York Mets vs Braves", "Atlanta vs New York Mets")
> 100
fuzz.token_set_ratio("React.js framework", "Reactjs")
> 40
fuzz.partial_token_set_ratio("React.js framework", "Reactjs")
> 71
```

fuzz.QRatio, fuzz.UQRatio

Just wrappers around `fuzz.ratio` with some validation and short-circuiting, included here for completeness. `UQRatio` is a unicode version of `QRatio`.

fuzz.WRatio

An attempt to weight (the name stands for 'Weighted Ratio') results from different algorithms to calculate the 'best' score. Description from the source code:

```
1. Take the ratio of the two processed strings (fuzz.ratio)
2. Run checks to compare the length of the strings
    * If one of the strings is more than 1.5 times as long as the other
      use partial_ratio comparisons - scale partial results by 0.9
      (this makes sure only full results can return 100)
    * If one of the strings is over 8 times as long as the other
      instead scale by 0.6
3. Run the other ratio functions
    * if using partial ratio functions call partial_ratio,
      partial_token_sort_ratio and partial_token_set_ratio
      scale all of these by the ratio based on length
    * otherwise call token_sort_ratio and token_set_ratio
    * all token based comparisons are scaled by 0.95
      (on top of any partial scalars)
4. Take the highest value from these results
   round it and return it as an integer.
```

fuzz.UWRatio

Unicode version of `WRatio`.

share edit follow

answered Jun 13 '17 at 19:43

Dennis Golomazov
10.6k • 4 • 61 • 60

add a comment

Your Answer

Links Images Styling/Headers Lists Blockquotes Code HTML Advanced help

Post Your Answer

Not the answer you're looking for? Browse other questions tagged python string-comparison fuzzywuzzy or ask your own question.