Fuzzy String Matching in Python

Authors: Ivan Morhaliuk, Artur Pinkevych

Co to jest Fuzzy String Matching?

Technika znajdowania ciągów, które pasują do danego ciągu częściowo, a nie dokładnie.

```
Str1 = "Apple Inc."
Str2 = "apple Inc."
```

Zastosowanie

- Spelling check software
- Database mapping
- Search engines

```
Str1 = "Apple Inc."
Str2 = "apple Inc."
Result = Str1 == Str2
print(Result)

False
```

Problem

Gdy użytkownik źle wpisze słowo lub wpisze je częściowo, rozmyte dopasowywanie ciągów znaków pomaga w znalezieniu właściwego słowa – co widzimy w wyszukiwarkach

```
Str1 = "Apple Inc."
Str2 = "apple Inc."
Result = Str1.lower() == Str2.lower()
print(Result)

True
```

The Levenshtein Distance

Odległość Levenshteina jest miarą mierzącą odległość od siebie dwóch sekwencji słów.

Metryka ta została nazwana na cześć Vladimira Levenshteina, który pierwotnie rozważał ją w 1965 roku.

```
Str1 = "Apple Inc."
Str2 = "apple Inc"
Distance = levenshtein_ratio_and_distance(Str1,Str2)
print(Distance)
Ratio = levenshtein_ratio_and_distance(Str1,Str2,ratio_calc = True)
print(Ratio)

The strings are 2 edits away
0.8421052631578947
```

Partial Ratio

Współczynnik częściowy pomaga nam w dopasowywaniu podłańcuchów. Bierze najkrótszy ciąg i porównuje go ze wszystkimi podciągami o tej samej długości.

Dane wyjściowe kodu dają 100, ponieważ partial_ratio() sprawdza tylko, czy jeden ciąg jest podłańcuchem drugiego.

```
Str1 = "Los Angeles Lakers"
Str2 = "Lakers"
Ratio = fuzz.ratio(Str1.lower(),Str2.lower())
Partial_Ratio = fuzz.partial_ratio(Str1.lower(),Str2.lower())
print(Ratio)
print(Partial_Ratio)
```

```
50
100
```

Token Sort Ratio

W przypadku współczynnika sortowania tokenów łańcuchy są tokenizowane i wstępnie przetwarzane przez konwersję na małe litery i pozbycie się interpunkcji. Ciągi są następnie sortowane alfabetycznie i łączone ze sobą. Na końcu, współczynnik podobieństwa odległości Levenshteina jest obliczany między stringami.

```
Str1 = "united states v. nixon"
Str2 = "Nixon v. United States"
Ratio = fuzz.ratio(Str1.lower(),Str2.lower())
Partial_Ratio = fuzz.partial_ratio(Str1.lower(),Str2.lower())
Token_Sort_Ratio = fuzz.token_sort_ratio(Str1,Str2)
print(Ratio)
print(Partial_Ratio)
print(Token_Sort_Ratio)

59
74
100
```

Token Set Ratio

Współczynnik zestawu tokenów wykonuje operację na zestawie, która usuwa wspólne tokeny zamiast tylko tokenizacji ciągów, sortowania, a następnie wklejania tokenów z powrotem. Dodatkowe lub te same powtórzone słowa nie mają znaczenia.

```
Str1 = "The supreme court case of Nixon vs The United States"
Str2 = "Nixon v. United States"
Ratio = fuzz.ratio(Str1.lower(),Str2.lower())
Partial_Ratio = fuzz.partial_ratio(Str1.lower(),Str2.lower())
Token_Sort_Ratio = fuzz.token_sort_ratio(Str1,Str2)
Token_Set_Ratio = fuzz.token_set_ratio(Str1,Str2)
print(Ratio)
print(Partial_Ratio)
print(Token_Sort_Ratio)
print(Token_Set_Ratio)

57
77
58
95
```

Process Module

Jeśli mamy listę ciągów i chcemy znaleźć najbardziej pasujący ciąg z listy z danym ciągiem, możemy wykorzystać moduł „proces”.

```
from fuzzywuzzy import process
str2Match = "apple inc"
strOptions = ["Apple Inc.","apple park","apple incorporated","iphone"]
Ratios = process.extract(str2Match,strOptions)
print(Ratios)
# You can also select the string with the highest matching percentage
highest = process.extractOne(str2Match,strOptions)
print(highest)
```

```
[('Apple Inc.', 100), ('apple incorporated', 90), ('apple park', 67), ('iphone', 30)]
('Apple Inc.', 100)
```