Fuzzy String Matching With Pandas and FuzzyWuzzy

Data Analysis Pandas

sammy ongaya Fuzzy, FuzzyWuzzy, string matching

One of the big challenges with data is that it is unstructured and incomplete. How do we tell that UK is the same as United Kingdom and Unite King also same as United Kingdom? Does Jpan mean Japan? Or is guugle same as google? Well, this is what we need to understand and correct before we can start analyzing and predicting our data else we risk ignoring very important data. The data that is being generated daily is unstructured and the objective of a

Recent posts

- Apriori Algorithm
- Singular Value Decomposition
- Principal Component Analysis
- Hierarchical Agglomerative Clustering
- K-Means Clustering Algorithm
- Unsupervised Machine Learning
- Multi-Layer Perceptron Model
- Logistic Regression Algorithm
- Linear Regression Algorithm
- Random Forest Algorithm

data analyst is to standardize it into a uniform format for easier and accurate analysis. In this post we are going to learn about fuzzy string matching with Pandas and FuzzyWuzzy.

Fuzzy String Matching With Pandas and FuzzyWuzzy

Fuzzy string matching or searching is a process of approximating strings that matches a particular pattern. Note that it gives an approximate and there is no guarantee that the string can be exact, however, sometimes the string accurately matches the pattern. How close the string is to a given match is measured by the *edit distance*. FuzzyWuzzy uses *Levenshtein Distance* to calculate the *edit distance.*

String Matching Problem Definition

From Wikipedia here is how string matching problem is defined. *Given a pattern string P = p 1 p 2 . . . p m {\displaystyle P=p_{1}p_{2}...p_{m}} and a text string T = t 1 t 2 ... t n {\displaystyle T=t_{1}t_{2}\dots t_{n}}, find a substring T j ′ , j = t j ′ ... t j {\displaystyle T_{j',j}=t_{j'}\dots t_{j}} in T, which, of all substrings of T, has the smallest edit distance to the pattern P. A brute-force approach would be to compute the edit distance to P for all substrings of T, and then choose the substring with the minimum distance.*

Application of String Matching

String matching has a wide range of applications from spell checking, spam filtering, and plagiarism detection among other uses.

Installing FuzzyWuzzy

FuzzyWuzzy is an open-source library developed by the **SeatGeek.** SeatGeek is the Web's largest event ticket search engine. The library was created due to the problem faced by having many different ways of referring to the same event, adding and hiding location on events and concerts. Nowadays, the problem is being experienced in different domains.

To install FuzzyWuzzy you can use the *pip* command as follows;

```
Pip install fuzzywuzzy
```

You can also use the conda to install FuzzyWuzzy. The following command will install the library;

```
conda install -c conda-forge fuzzywuzzy
```

Importing The Library

To use FuzzyWuzzy we need to import the required libraries as follows.

```
from fuzzywuzzy import fuzz
from fuzzywuzzy import process
```

Calculating Simple Ratio

```
from fuzzywuzzy import fuzz

simple_ratio=fuzz.ratio("Hello world", "Hello world!")

print(simple_ratio)
```

```
Output

96
```

Calculating Partial Ratio

```
from fuzzywuzzy import fuzz

partial=fuzz.partial_ratio("Hello world", "Hello world!")

print(partial)
```

```
Output

100
```

Calculating Token Sort Ration

Tag Cloud

Downloads

```
from fuzzywuzzy import fuzz

sort_ratio=fuzz.token_sort_ratio("Hello world", "World Hello!")

print(sort_ratio)
```

```
Output

100
```

Calculating Token Set Ratio

```
from fuzzywuzzy import fuzz

set_ratio=fuzz.token_set_ratio("Hello", "World Hello!")

print(set_ratio)
```

```
Output

100
```

Extracting Strings From List of Choices

When we have a list of options and we want to match with our string we can use the *extract* or *extractOne* methods. The *limit* argument in extract method specifies the number of matches to return. The extractOne method returns exactly one match with the highest ratio.

Extract

```
from fuzzywuzzy import process

choices = ["Hello world is an introductory phrase in programming",
           "Game of throne is a movie",
           "Albert Einstein developed the theory of relativity",
           "Asia is the largest continent in the world"]

extract_all=process.extract("Hello world", choices, limit=2)

print(extract_all)
```

```
Output
```

NumPy sample csv data (30 downloads)
births-and-deaths (30 downloads)
pima-indians-diabetes (30 downloads)
iris data set (31 downloads)
boston-house-price-dataset (31 downloads)
live-graph-test (32 downloads)
American Housing Affordability (35 downloads)
FMEL-Dataset (36 downloads)
NumPy sample text data (40 downloads)
string matched country names (41 downloads)

Archives

July 2018 (5)
June 2018 (10)
May 2018 (11)
April 2018 (8)
March 2018 (16)
February 2018 (5)
January 2018 (5)
December 2017 (12)
November 2017 (8)
October 2017 (2)

```
[('Hello world is an introductory phrase in programming', 90), ('Asia is the largest continent in
```

extractOne

```
from fuzzywuzzy import process

choices = ["Hello world is an introductory phrase in programming",
           "Game of throne is a movie",
           "Albert Einstein developed the theory of relativity",
           "Asia is the largest continent in the world"]

extract_one=process.extractOne("Hello world", choices)

print(extract_one)
```

```
Output

('Hello world is an introductory phrase in programming', 90)
```

Fuzzy String Matching With Pandas.

We have seen how to install FuzzyWuzzy and learned how to use it. Now we are going to apply what we have learned in both Pandas and string matching to clean our sample data. If you are not familiar with Pandas I recommend to visit my previous posts on Pandas. In this project we have one data set which contains a list of countries, their country codes, area code, and region among other details in the wrong country names csv file. The country names are not correctly typed. The objective is to match the wrong country names and return the correct name for each wrongly spelled country name.

To solve the above problem we need a list of correct country names. This is provided in a country names csv file. We will string match the wrong country names with the correct country names and return a match with a ratio for each match. In our Pandas DataFrame we will create two new columns one for correct_country_name and another for country_names_ratio. We will save the output to a string_matched_country_names csv file.

Here are the csv files we need; wrong country names (64 downloads) , country names (59 downloads)

```
from fuzzywuzzy import process
import pandas as pd


names_array=[]
ratio_array=[]
def match_names(wrong_names,correct_names):
    for row in wrong_names:
        x=process.extractOne(row, correct_names)
        names_array.append(x[0])
        ratio_array.append(x[1])
    return names_array,ratio_array


#Wrong country names dataset
df=pd.read_csv("wrong-country-names.csv",encoding="ISO-8859-1")
wrong_names=df['name'].dropna().values

#Correct country names dataset
choices_df=pd.read_csv("country-names.csv",encoding="ISO-8859-1")
correct_names=choices_df['name'].values

name_match,ratio_match=match_names(wrong_names,correct_names)

df['correct_country_name']=pd.Series(name_match)
df['country_names_ratio']=pd.Series(ratio_match)

df.to_csv("string_matched_country_names.csv")

print(df[['name','correct_country_name','country_names_ratio']].head(10))
```

```
Output

                    name    correct_country_name  country_names_ratio
0           Ålend Islends           Åland Islands                   83
1                 elbenie                   Niue                   77
2                 endorre                Andorra                   71
3                 eustrie                Austria                   71
4                 Belerus                Belarus                   86
5                 Belgium                Belgium                  100
6  Bosnie end Herzegovine  Bosnia and Herzegovina                   86
7                Bulgerie               Bulgaria                   75
8                 Croetie                Croatia                   71
9          Czech Republic          Czech Republic                  100
```

Deixe uma

In the above example you notice something about the matching. The second name is not well matched, *elbenie* is supposed to be *albania.* However, our string matching code has managed to get most of the matching correct. A point to note also is about the ratio, the ratio is generated by the edit distance between pattern and the string to be matched. The lower ratio does not imply that the matching is inaccurate also the higher ratio does not mean that the matching is accurate, apart from the ratio being 100 which will always be accurate unless it's a partial ratio . This can be seen on the second country name whose ratio is high but yields inaccurate results and the third country name whose ratio is low but yields accurate matching.

When working with Pandas and FuzzyWuzzy ensure that the DataFrame of strings to be matched does not have NaN else the code will throw an exception. Here is the output csv after string matching; string matched country names (41 downloads)

Conclusion

In this post you have seen how to use string matching to compare and similar data. FuzzyWuzzy is a great library for string approximation. There are other techniques for string matching like NLP techniques that we'll see in other post. Fuzzy string approximation uses a brute-force approach in comparing string with a given pattern. The challenges of this approach is that it does not semantically work well with synonyms. In fuzzy string matching United Kingdom and England are two different strings yet semantically we know they refer to the same thing. Fuzzy string matching only applies to text data and not to other data such as sound or images However, fuzzy string matching remains to be a very important process for data scientist in matching similar data.

Optimizing Pandas Code

Introduction to NumPy

About Galaxy Data Technologies

Galaxy data technologies is an online learning platform that focuses on technology and data. We leverage data science, software engineering and computer science concepts to learn how to solve complex problems using data.

Galaxy Data Technologies

Menu

Home
About
Data Analysis
Data Visualization
Machine Learning
Deployment
Projects