countries_lib Documentation

Release 1.0

Aleksey Pauls

Jul 18, 2017

Оглавление

ГЛАВА 1

Описание

Основные возможности

Данный модуль предназначен для нормализации (нахождения корректного) названия страны посредством работы с прилагающейся простейшей базой данных.

На данный момент модуль поддерживает следующие операции:

- Нормализация названия страны
- Добавление в базу данных новой пары “возможное название”-“корректное название”
- Удаление из базы данных возможного названия

Подробнее о нормализации

На вход принимаются:

- Название страны (официальное, общее, полное или национальное, на любом из 9 (в среднем, зависит от страны) популярных языков (deu, eng, fra, hrv, ita, jpn, nld, rus, spa))
- Альтернативное название страны (транслит, неофициальное, сокращенное)
- 2-х или 3-х буквенный код страны
- Название столицы страны
- Название региона страны (поддерживается до 14 (в зависимости от страны) переводов для 3700 регионов)
- Объединения приведенных выше вариантов с различными знаками препинания (например, “London, UK”)
- “Странные” варианты, которые сложно классифицировать (добавляются в базу вручную)
- А также все выше приведенные варианты с опечатками и в любом регистре

В результате обработки и поиска по базе данных получаем корректное (по умолчанию - общее) название страны.

База данных

Эта база данных (б/д) создавалась на основе следующих б/д: https://github.com/mledoze/countries и https://github.com/x88/i18nGeoNamesDB, распространяемых с лицензией MIT. Она организована в виде словаря - каждому "ключу" (возможному названию) соответствует "значене" (корректное название страны). Это упрощает поиск, добавление и удаление элементов. Данная бд реализована с помощью модуля shelve стандартной библиотеки языка Python. В базе содержится более 9700 возможных вариантов названий стран. Для каждого ключа определяется приоритет: целое число 1 или 2, и записывается перед ключем. Приоритет 1 означает, что ключ является названием или индексом страны, приоритету 2 соответствуют названия столиц и регионов. Также, все ключи приводятся к нижнему регистру, а приоритет записывается вместе со значением Поэтому запись в б/д имеет вид: ключ - "some name", значение - "1Correct Name". Подробнее об этом в описании функции нормализации.

Тесты

В модуль встроены тесты, позволяющие проверить его функциональность при внесении изменений. Далее идут тесты и их описание:

1. test_simple_name - проверяет работу функции normalize_country_name на простых входных данных
2. test_punctuation_sensitivity - проверяет удаление пунктуации в функции normalize_country_name
3. test_upper_register - проверяет работу функции normalize_country_name на входной строке в верхнем регистре
4. test_low_register - проверяет работу функции normalize_country_name на входной строке в нижнем регистре
5. test_missed_letter - проверяет исправление опечатки типа "пропущенная буква" в функции normalize_country_name
6. test_excess_letter - проверяет исправление опечатки типа "лишняя буква" в функции normalize_country_name
7. test_another_letter - проверяет исправление опечатки типа "неправильная буква" в функции normalize_country_name
8. test_simple_two_words_name - проверяет работу функции normalize_country_name с входной строкой из 2-х слов (разделитель - пробел)
9. test_excess_word_name - проверяет работу функции normalize_country_name с входной строкой из 2-х слов, одно из которых - лишнее
10. test_american_paris_like_construction - проверяет работу приоритета в функции normalize_country_name
11. test_standard_accuracy_result - проверяет вывод функции normalize_country_name для несуществующего имени при стандартной точности
12. test_correct_accuracy_type - проверяет ввозможность ввода корректного необязательного аргумента dif_acc в функции normalize_country_name

13. test_incorrect_accuracy_type - проверяет ввозможность ввода некорректного (тип) необязательного аргумента dif_acc в функции normalize_country_name
14. test_incorrect_accuracy_value - проверяет ввозможность ввода некорректного (значение) необязательного аргумента dif_acc в функции normalize_country_name
15. test_non_existing_object_delete - проверяет удаление несуществующего ключа в функции del_country_name
16. test_match - проверяет добавление нового ключа и значения в функции match_country_name
17. test_existing_object_delete - проверяет удаление существующего ключа в функции del_country_name
18. test_correct_priority_match - проверяет добавление нового ключа и значения в функции match_country_name, причем необязательный аргумент priority корректен
19. test_incorrect_priority_match - проверяет добавление нового ключа и значения в функции match_country_name, причем необязательный аргумент priority некорректен
20. test_incorrect_match - проверяет добавление некорректного нового ключа и некорректного значения в функции match_country_name
21. test_incorrect_delete - проверяет удаление некорректного ключа в функции del_country_name

ГЛАВА 2

Функции

Нормализация страны - normalize_country_name

Аргументы и возвращаемое значение

Функция имеет вид: normalize_country_name(posname, dif_acc=0.7)

Принимает на вход один обязательный аргумент posname (от “possible name”) типа string - нормализуемое название, и один необязательный - dif_acc (от differency accurate) типа float - параметр точности при поиске подходящего ключа в библиотеке, принимающий значения от 0.0 до 1.0 (по умолчанию - 0.7).

Выдает строку типа string, содержащую либо общее название страны, либо ‘None’, если выполнение прошло успешно. Если было вызвано исключение, то строка содержит ‘DatabaseError’ (это означает, что не найдена корректная база данных по пути из переменной DB_PATH) или ‘Invalid argument type’, если хотя бы один из аргументов задан неправильно (имеет некорректный тип или значение).

Логика работы

В функции преобразуется входная строка posname и находится ближайший по расстоянию между строками ключ (dif_acc определяет минимальное расстояние, при котором строки считаются похожими). Для этого предпринимаются следующие шаги:

1. Удаление пунктуации и приведение к нижнему регистру:

   Из входной строки posname удаляются символы, обозначающие пунктуацию, не используемые в названиях пунктуации, чтобы избежать конфликтов с национальными символами для различных языков. Иной возможный способ реализации - регулярное выражение. Также posname приводится к нижнему регистру, что позволяет сравнивать его с ключами (которые все записаны в нижнем регистре). Таким образом достигается нечувствительность к регистру входной строки.

2. Поиск совпадения всей строки:

   2.1. И ключа с приоритетом “1”:

С помощью функции get_close_matches из модуля difflib стандартной библиотеки языка Python находи
между строками ключ из нашей базы данных. При этом проверяем разность
длин совпавшего ключа и входной строки. Если эта разность по модулю
не больше 1, то есть длины примерно равны, то значение по полученному
ключу возвращается функцией и ее работа прекращается.

2.2. И ключа с приоритетом “2”:

Аналогично, различие только в приоритете.

3. Разбиение входной строки на слова (разделитель - ‘ ‘)
4. Поиск совпадения слов из входной строки:

4.1. И ключа с приоритетом “1”. При этом:

4.1.1. Длины ключа и слова примерно равны

Если найден такой ключ, то функция возвращает значение по этому
ключу и завершает работу. Если нет, то выполнение продолжается по
указанному порядку.

4.1.2. Длины ключа и слова не равны

Аналогично пункту 4.1.1, но без сравнения длины строк.

4.2. И ключа с приоритетом “2”. При этом:

4.2.1. Длины ключа и слова примерно равны

Аналогично 4.1.1, разница в приоритете ключей.

4.2.2. Длины ключа и слова не равны

Аналогично 4.1.2, разница в приоритете ключей.

5. Завершение работы - совпадения не найдено

Если функция дошла до этого этапа, значит для входной строки не существует похожего ключа из базы да
какая страна имелась в виду, и функция возвращает строку ‘None’. Если тип пере-
данных аргументов некорректен, функция возвращает строку ‘Invalid argument type’.
Если же в процессе выполнения функции было вызвано исключение, то функция
возвращает строку ‘DatabaseError’ (с огромной вероятностью это означает, что не
найдена корректная база данных по пути из константы DB_PATH)

Такая структура позволяет охватить многие возможные опечатки и странные варианты. Например, удаление символов пунктуации и приведение к нижнему регистру позволяют корректно определить страну в случаях “America!!!” и “uNITED kINGDOM”, а приоритет ключей разрешает ситуации типа “американского Парижа” - по запросу “Paris, US” функция вернет “Unated States”. Сравнение длины ключа и строки (слова) нужно для того, чтобы сначала исключить варианты, когда совпало только начало или другая часть строки (слова).

А также, так как значение записывается вместе с приоритетом перед ним в одну строку, то нулевой элемент строки значения не учитывается при возвращении строки. То есть, в базе данных по ключу ‘ru’ лежит значение ‘1Russia’, где 1 - приоритет ключа ‘ru’, и возвращается только часть строки значения: ‘Russia’

Добавление возможного названия страны - match_country_name

Аргументы и возвращаемое значение

Функция имеет вид: match_country_name(key, value, priority=2)

Принимает на вход два обязательных аргумента key и value типа string - возможное и корректное названия соответственно, и один необязательный - priority типа int - приоритет ключа, принимающий значения 1 или 2 (по умолчанию - 2) и определяющий, что содержится в ключе: название, сокращение, индекс или перевод названия страны, если приоритет равен 1, и все остальное, если приоритет равен 2. Так как большинство ключей, подходящих под приоритет 1, уже в базе, то возможно задать приоритет по умолчанию равный 2.

Выдает строку 'Invalid argument type' типа string , если хотя бы один из аргументов задан неправильно (имеет некорректный тип или значение), строку 'DatabaseError', если произошла ошибка во время открытия базы данных по пути из переменной DB_PATH, и ничего не возвращает (None), если добавление прошло успешно.

Логика работы

Сначала функция проверяет типы и значения переменных. Если проверка прошла успешно, то делается попытка открыть базу данных. Если база данных успешно открыта, то ключ (key, возможное название) приводится к нижнему регистру, значение (value, корректное название) объединяется с приоритетом (priority) в одну строку, и пара (ключ, приоритет+значение) записывается в базу данных. Если ключ уже находится в базе, то его значение перезаписывается.

Удаление возможного названия страны - del_country_name

Аргументы и возвращаемое значение

Функция имеет вид: del_country_name(key)

Принимает на вход один обязательный аргумент key типа string - возможное название, которое нужно удалить из базы данных.

Выдает строку 'Invalid argument type' типа string , аргумент задан неправильно (имеет некорректный тип или значение), строку 'DatabaseError', если произошла ошибка во время открытия базы данных по пути из переменной DB_PATH, и ничего не возвращает (None), если удаление прошло успешно.

Логика работы

Сначала функция проверяет типы и значения переменных. Если проверка прошла успешно, то делается попытка открыть базу данных. Если база данных успешно открыта, то ключ (key, возможное название) приводится к нижнему регистру и проверяется на наличие в базе данных. Если ключ находится в базе, то он и его значение удаляются, и функция завершается. Если нет - то выполнение функции завершается сразу.

ГЛАВА 3

Применение

Общая демонстрация

Код, используемый для демонстрации возможностей модуля:

```
from countries_lib import country


def main():
""" Пример использования библиотеки для нормализации названия страны """

# Вывод корректные названия для вариантов из списка:
test_list = ['USA', 'US', 'Amurica!!!', 'NewYork', 'Untgd States of America', 'Paris, USA', 'agagagag']
for variant in test_list:
    print(variant, ' - ', country.normalize_country_name(variant))
print('------------------------------------------------------------')
# Добавление значения
print('Проверка "AddCountryTest" на существование: ', country.normalize_country_name('AddCountryTest
↪'))
country.match_country_name('AddCountryTest', 'AddCountryTest')
print('Проверка "AddCountryTest" на существование: ', country.normalize_country_name('AddCountryTest
↪'))
print('------------------------------------------------------------')
# Удаление значения
print('Проверка "AddCountryTest" на существование: ', country.normalize_country_name('AddCountryTest
↪'))
country.del_country_name('AddCountryTest')
print('Проверка "AddCountryTest" на существование: ', country.normalize_country_name('AddCountryTest
↪'))
print('------------------------------------------------------------')
# Демонстрация низкой и высокой точности
print('Testing variant: ', 'ololo')
print('0.3 (low) accurate: ', country.normalize_country_name('ololo', 0.3))
print('0.6 (standard) accurate: ', country.normalize_country_name('ololo'))
print('0.9 (high) accurate: ', country.normalize_country_name('ololo', 0.9))
```

```
print('Testing variant: ', 'Rasia')
print('0.3 (low) accurate: ', country.normalize_country_name('Rasia', 0.3))
print('0.6 (standard) accurate: ', country.normalize_country_name('Rasia'))
print('0.9 (high) accurate: ', country.normalize_country_name('Rasia', 0.9))


if __name__ == "__main__":
    main()
```

Вывод при выполнении данного кода:

```
USA  -  United States
US  -  United States
Amurica!!!  -  United States
NewYork  -  United States
Untgd States of America  -  United States
Paris, USA  -  United States
agagagag  -  None
------------------------------------------------------------
Проверка "AddCountryTest" на существование:  None
Проверка "AddCountryTest" на существование:  AddCountryTest
------------------------------------------------------------
Проверка "AddCountryTest" на существование:  AddCountryTest
Проверка "AddCountryTest" на существование:  None
------------------------------------------------------------
Testing variant:  ololo
0.3 (low) accurate:  Norway
0.6 (standard) accurate:  None
0.9 (high) accurate:  None
Testing variant:  Rasia
0.3 (low) accurate:  Russia
0.6 (standard) accurate:  Russia
0.9 (high) accurate:  None
```

Как видно из результатов, функции делают именно то, что заявлено в их описании (без учета ошибок, это рассматривается далее).

Возможна другая форма импорта:

```
from countries_lib.country import normalize_country_name, match_country_name, del_country_name
```

Такая форма позволяет обращаться к функциям напрямую.

Необязательные аргументы

В функциях normalize_country_name и match_country_name есть необязательные аргументы. Рассмотрим их применение.

dif_acc (функция normalize_country_name)

Параметр точности (коэффициент совпадения строк). Отвечает за минимальное расстояние между строками, при котором они считаются похожими. Принимает значения от 0.0 до 1.0 и по умолчанию равен 0.7. Чем больше данный параметр, тем точнее будет результат - это напрямую регулирует исправление опечаток. При низком dif_acc даже для сильных опечаток будет найден подходящий вариант, но

при этом возможно нахождение подходящего варианта для бессмысленной последовательности букв типа “azazazaz”. При высоком dif_acc уменьшится количество исправляемых опечаток и на очевидные для человека ошибки программа будет выдавать, что совпадений не найдено.

Стоит отметить, что если для заданного возможного имени есть полностью совпадающий ключ, то его значение будет возвращено независимо от dif_acc, так как расстояние между этими строками будет равно 0 (коэффициент совпадения - 1.0, предельное корректное значение). Поэтому dif_acc влияет только на исправление опечаток.

prioritet (функция match_country_name)

Приоритет ключа, принимающий значения 1 или 2 (по умолчанию - 2) и определяющий, что содержится в ключе: название, сокращение, индекс или перевод названия страны, если приоритет равен 1, и все остальное, если приоритет равен 2. Так как большинство ключей, подходящих под приоритет 1, уже в базе, то возможно задать приоритет по умолчанию равный 2.

Работа с ошибками

Пример кода, обрабатывающего функцию normalize_country_name:

```
name = normalize_country_name('Some Name')
if name == 'Invalid argument type':
    # Your code
elif name == 'DatabaseError':
    # Your code
else:
    # Your code
```

Для функций match_country_name и del_country_name принцип тот же.

Если Вы уверены в том, что на вход функций подаются корректные аргументы и с базой данных все в порядке, то в обработке ошибок нет необходимости.