

Chatbot z rozmytymi wyrażeniami regularnymi

Jakub Wiątek - Mateusz Świstak - Grzegorz Zaprzęta

Wyrażenia regularne

Expression

```
/[^\x00-\x7F]+\*(?:[^\x00-\x7F]|-)*-/g
```

Text

Tests

The river flowing from the Blue Dragon Mountain into the lake, the pure white palace floating and the silver Adventurers' Guild.

そして町を囲むように配置された七つの塔と、大きく広がる草原地帯。

The seven towers encircling the city, surrounded by a vast meadow.

.....あ。

...Ah.

テスト

...テスト... a ああ... bw

teテスト。

!@#\$%^&*()-+=.

.?!<>[]{}|\\/-ABCabc~|*→Ö×ë

テスト。

-Τα εννέα πουλιά -

MIRROR

Rozmyte wyrażenia regularne

- W porównaniu do standardowych wyrażeń regularnych, rozmyte wyrażenia regularne są bardziej elastyczne i pozwalają na uwzględnienie stopnia dopasowania do wzorca.
- Dobrym przykładem na zobrazowanie sposobu działania rozmytych wyrażeń regularnych są np. silniki wyszukiwania zaimplementowane w popularnych przeglądarkach. Jeśli przykładowo wpiszemy w pasek przeglądania słowo “facbok” zamiast “facebook” to przeglądarka domyśli się, że chodziło nam o drugie słowo i zwróci odpowiednie wyniki wyszukiwania.

TheFuzz

Biblioteka użyta do wykonania projektu to TheFuzz. Używa ona tzw. Odległości Levenshteina do obliczania różnic między wyrażeniami. Odległość ta jest minimalną liczbą edycji potrzebnych do zamiany jednego ciągu znaków na drugi.

Simple Ratio

```
>>> fuzz.ratio("this is a test", "this is a test!")  
97
```



Partial Ratio

```
>>> fuzz.partial_ratio("this is a test", "this is a test!")  
100
```



Token Sort Ratio

```
>>> fuzz.ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")  
91  
>>> fuzz.token_sort_ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear")  
100
```



Odległość Levenshteina

Odległość Levenshteina pomiędzy napisami identycznymi, np.

```
pies  
pies
```

jest zerowa – skoro są identyczne, to potrzeba zero działań, by jeden z nich przeprowadzić na drugi.

Odległość Levenshteina pomiędzy napisami:

```
granat  
granit
```

wynosi 1, ponieważ do przeprowadzenia pierwszego na drugi wystarcza jedno działanie: zamiana litery **a** na **i**.

Odległość pomiędzy napisami:

```
orczyk  
oracz
```

równa jest 3, ponieważ do przeprowadzenia pierwszego napisu na drugi potrzeba trzech działań: usunięcia liter **y** i **k** oraz wstawienia litery **a**.

Odległość pomiędzy napisami:

```
marka  
ariada
```

wynosi 4, ponieważ potrzeba co najmniej czterech działań, np.: usunięcia litery **m**, zamiany **k** na **i** oraz dodania **d** i **a**.

Implementacja

Struktura danych pliku answers.json

```
{ } answers.json > [ ] answers > { } 1 > [ ] triggers
```

```
1 {
2   "answers": [
3     {
4       "name": "greeting",
5       "triggers": [
6         "hi",
7         "hello",
8         "hey"
9       ],
10      "responses": [
11        {
12          "text": "Hello I'm {name} and I'm here to help you.",
13          "variables": [
14            {
15              "name": "name"
16            }
17          ]
18        },
19        {
20          "text": "Hello nice to meet you."
21        },
22        {
23          "text": "Hi there."
24        },
25        {
26          "text": "Hey there."
27        }
28      ]
29    },
30  ]
31 }
```

```
{ } answers.json > [ ] answers > { } 8 > [ ] triggers > 2
```

```
151 {
152   "name": "subtraction",
153   "triggers": [
154     {
155       "text": "what is {number1} minus {number2}",
156       "similarity": 70,
157       "variables": [
158         {
159           "name": "number1",
160           "context_extractor": "((\\d+)?=\\d+)[.\\s]"
161         },
162         {
163           "name": "number2",
164           "context_extractor": "((\\d+)?!(\\d+)[.\\s])"
165         }
166       ]
167     },
168     {
169       "text": "{number1} minus {number2}",
170       "similarity": 70,
171       "variables": [
172         {
173           "name": "number1",
174           "context_extractor": "((\\d+)?=\\d+)[.\\s]"
175         },
176         {
177           "name": "number2",
178           "context_extractor": "((\\d+)?!(\\d+)[.\\s])"
179         }
180       ]
181     }
182   ],
183   "responses": [
184     {
185       "text": "{number1} minus {number2} is {result}",
186       "variables": [
187         {
188           "name": "number1",
189           "context_extractor": "((\\d+)?=\\d+)[.\\s]"
190         },
191         {
192           "name": "number2",
193           "context_extractor": "((\\d+)?!(\\d+)[.\\s])"
194         },
195         {
196           "name": "result",
197           "method": {
198             "name": "subtraction",
199             "arguments": [
200               "number1",
201               "number2"
202             ]
203           }
204         }
205       ]
206     }
207   ]
208 }
```

```
{ } answers.json > [ ] answers > { } 8 > [ ] triggers > 2
```

```
325 {
326   "name": "city_info",
327   "triggers": [
328     {
329       "text": "Tell me about {city}",
330       "variables": [
331         {
332           "name": "city",
333           "fuzzy_extractor": [
334             "Paris",
335             "Rome",
336             "Prague",
337             "Barcelona",
338             "London",
339             "Amsterdam",
340             "Berlin",
341             "Vienna",
342             "Venice",
343             "Dublin",
344             "Edinburgh",
345             "Athens",
346             "Budapest",
347             "Lisbon"
348           ]
349         }
350       ]
351     },
352     {
353       "text": "{city_info}",
354       "variables": [
355         {
356           "name": "city_info",
357           "method": {
358             "name": "city_info",
359             "arguments": [
360               "city"
361             ]
362           }
363         }
364       ]
365     }
366   ],
367   "responses": [
368     {
369       "text": "{city_info}"
370     }
371   ]
372 }
```

Szukanie najbardziej pasującej odpowiedzi na podstawie wyzwalaczy

Wczytywanie odpowiedzi oraz błędów z pliku JSON

```
def load_answers(self, filename):  
    with open(filename, "r", encoding="utf-8") as file:  
        data = file.read()  
        json_data = json.loads(data)  
  
        self.answers = json_data["answers"]  
        self.errors = json_data["errors"]
```

```
def find_answer(self, text: str) -> Answer | None:  
    max_match = 50  
    best_answer = None  
  
    for answer in self.answers:  
        for trigger in answer["triggers"]:  
            if isinstance(trigger, str):  
                similarity = fuzz.token_set_ratio(text.lower(), trigger.lower())  
            else:  
                trigger_text = self.process_variables(  
                    text, trigger["text"], trigger.get("variables", [])  
                )  
  
                similarity = fuzz.token_sort_ratio(  
                    text.lower(), trigger_text.lower(), full_process=True  
                )  
  
            required_similarity = (  
                trigger.get("similarity", 50) if isinstance(trigger, dict) else 0  
            )  
  
            if similarity > max_match and similarity > required_similarity:  
                max_match = similarity  
                best_answer = answer  
  
    return best_answer
```


Przetwarzanie odpowiedzi

```
def process_answer(self, question: str, answer: Answer) -> str:
    response: Response = random.choice(answer["responses"])
    text = self.process_variables(
        question, response["text"], response.get("variables", [])
    )

    return text
```

Przetwarzanie zmiennych w tekście

```
def process_variables(
    self, question: str, text: str, variables: List[Variable]
) -> str:
    for variable in variables:
        text = self.process_variable(question, text, variable)

    return text
```

Przetwarzanie pojedynczej zmiennej

```
def process_variable(self, question: str, text: str, variable: Variable) -> str:
    context_extractor = variable.get("context_extractor", None)
    method = variable.get("method", None)
    fuzzy_extractor = variable.get("fuzzy_extractor", None)

    value = None

    if context_extractor:
        match = search(context_extractor, question)

        if match:
            value = match.group(1)

        self.ctx[variable["name"]] = f"{value if value else ''}"
    elif method:
        arguments = [self.ctx.get(arg, "") for arg in method["arguments"]]
        value = self.methods[method["name"]](*arguments)
        self.ctx[variable["name"]] = value
    elif fuzzy_extractor:
        [value, real_value] = self.fuzzy_extractor(question, fuzzy_extractor)
        self.ctx[variable["name"]] = value
        self.ctx[variable["name"] + "_real"] = real_value
    else:
        value = self.ctx.get(variable["name"], None)

    return text.replace(f"{{{variable['name']}}}", value if value else "")
```

Rozmyte wnioskowanie wartości z tekstu

```
def fuzzy_extractor(self, text: str, true_values: List[str]) -> tuple[str, str]:
    words = text.split(" ")

    max_match = 50
    best_match = None
    real_value = None

    for word in words:
        for true_value in true_values:
            match = fuzz.token_sort_ratio(word.lower(), true_value.lower())

            if match > max_match:
                max_match = match
                best_match = true_value
                real_value = word

    return (best_match if best_match else "", real_value if real_value else "")
```

Pętla chatu

```
def chat(self):
    while True:
        try:
            question = input("You# ")
            answer = self.find_answer(question)

            if answer:
                print(f"{self.name}#", self.process_answer(question, answer))

                if answer["name"] == "exit":
                    break
            else:
                raise Exception()
        except KeyboardInterrupt:
            print("\nGoodbye!")
            break
        except Exception as e:
            print(random.choice(self.errors))
```

Funkcja uruchamiająca bota

```
def main():
    bot = Bot("Ravin")
    bot.chat()
```

Funkcje matematyczne

```
def addition(*args) -> str:
    return str(int(args[0]) + int(args[1]))

def subtraction(*args) -> str:
    return str(int(args[0]) - int(args[1]))

def multiplication(*args) -> str:
    return str(int(args[0]) * int(args[1]))

def division(*args) -> str:
    return str(int(args[0]) / int(args[1]))
```

Definicja typów danych

```
class Method(TypedDict):
    name: str
    arguments: List[str]

class Variable(TypedDict):
    name: str
    context_extractor: str
    method: Method
    fuzzy_extractor: List[str]

class Response(TypedDict):
    text: str
    variables: List[Variable]

class Trigger(Response):
    similarity: int

class Answer(TypedDict):
    name: str

    triggers: List[Trigger | str]
    responses: List[Response]
```

Prezentacja działania

```
You# hi
Ravin# Hi there.
You# what is 40 plus 3
Ravin# 40 plus 3 is 43
You# what is 40 times 3
Ravin# 40 times 3 is 120
You# tell about paris
Ravin# Paris, the 'City of Light,' is celebrated for its art, history, and gastronomy, featuring landmarks like the Eiffel Tower and Notre-Dame Cathedral. It's also a fashion and culinary capital, offering stylish boutiques and exquisite dining.
You# █
```

```
You# inspire me
Ravin# The only way to do great work is to love what you do. - Steve Jobs
You# music rec
Ravin# Feeling upbeat? 'Can't Stop the Feeling' by Justin Timberlake is great!
```

```
You# daily
Ravin# Remember to stay hydrated throughout the day, it's great for your health.
You# daily adv
Ravin# Take a short walk if you can, it's a good way to clear your mind and boost energy.
You# █
```

Randomizacja odpowiedzi

Silnik dobierania pytań do odpowiedzi bota działa w trybie relacji wielu do wielu. Oznacza to, że na jedno pytanie użytkownik może otrzymać wiele zróżnicowanych odpowiedzi aby zapobiec monotonii, a pytania o podobnej tematyce lecz innej strukturze spotkają się z podobną reakcją bota.

```
You# hi
Ravin# Hello nice to meet you.
You# hi
Ravin# Hello I'm Ravin and I'm here to help you.
You# hi
Ravin# Hello I'm Ravin and I'm here to help you.
You# hi
Ravin# Hi there.
You# █
```

Przypadek podania niepełnych wyrazów

You# smth intrestng?

Ravin# The Eiffel Tower can be 15 cm taller during the summer due to the expansion of the iron on hot days.

You# smth intrestng?

Ravin# A group of flamingos is called a 'flamboyance'. Quite fitting, don't you think?

You# tel about pari

Ravin# Paris, the 'City of Light,' is celebrated for its art, history, and gastronomy, featuring landmarks like the Eiffel Tower and Notre-Dame Cathedral. It's also a fashion and culinary capital, offering stylish boutiques and exquisite dining.

You# tel about brcl

Ravin# Barcelona is known for its art and architecture, highlighted by Gaudí's Sagrada Família and other modernist landmarks. The lively Las Ramblas street offers street performances and markets.

You# 40 min 3

Ravin# 40 minus 3 is 37

You# █

Dziękujemy za uwagę

Bibliografia

[1] <https://regexr.com/3acrs>

[2] <https://www.datacamp.com/tutorial/fuzzy-string-python>

[3] <https://github.com/seatgeek/thefuzz>

[4] https://pl.wikipedia.org/wiki/Odleg%C5%82o%C5%9B%C4%87_Levenshteina