

Narzędzie do kategoryzowania żartów

Gałań Sebastian
Jajkiewicz Jakub
Kieljan Dominik

1. Abstrakt

Naszym celem było stworzenie programu, który jest w stanie przyporządkować kategorię do podanego żartu. Z uwagi na sposób i istotę działania programu, możliwe jest także użycie go do innych celów, w zależności od podanych kategorii, słów kluczowych oraz ich wag. Przykładowo, po odpowiedniej modyfikacji możliwe jest znalezienie głównego tematu wypowiedzi lub tekstu.

2. Wstęp

2.1 Cel

W ramach naszego projektu skupiliśmy się na opracowaniu programu do analizy tekstu zawartego w żartach i klasyfikowaniu ich do odpowiednich kategorii tematycznych. Proces analizy tekstu był fundamentem naszego projektu, w którym mogliśmy wykorzystać podstawowe zagadnienia z przetwarzania języka naturalnego. Stworzony program następnie możemy wykorzystać do budowy bazy danych z żartami podzielonymi na różne kategorie tematyczne, które następnie możemy wykorzystać np. w aplikacji mobilnej lub na stronie internetowej.

2.2 Zakres

Projekt obejmuje podstawowe zagadnienia związane z przetwarzaniem języka naturalnego, takie jak:

1. Tokenizacja
2. Normalizacja słownictwa:
 - ❖ Przekształcanie na małe litery
 - ❖ Usuwanie znaków interpunkcyjnych
 - ❖ Usuwanie stop words'ów
 - ❖ Stemming

Dodatkowo projekt obejmuje takie zagadnienia jak:

1. Wczytywanie pliku .csv
2. Wczytywanie pliku .json

2.3 Metodyka

Do stworzenia programu zostały wykorzystane następujące narzędzia:

- Język python
- Casual Tokenizator
- PorterStemmer
- NLTK StopWords Korpus

3. Część teoretyczna

1. Stopwords - czyli słowa uzupełniające wypowiedź, ale nie dodające do niej nowego sensu, nowej informacji.

Wykorzystany przez nas NLTK StopWords Corpus zawiera ponad 300 takich słów, pozwoliło nam to na pierwotne oczyszczenie tekstu.

2. Stemming - proces usunięcia końcówek fleksyjnych wyrazu w celu uzyskania pierwotnej jego formy, tzw. tematu wyrazu

W naszym projekcie użyliśmy PorterStemmer z biblioteki nltk, który działa na zasadzie przechodzenia przez kilka kroków, które po kolei usuwają lub podmieniają kolejne części wyrazów.

More on Porter stemmer:

- It contains about 300 lines of code.
- It contains a few steps: 1a, 1b, 1c, 2, 3, 4, 5a, 5b.

The steps are as follows:

- Step 1a—"s" and "es" endings
- Step 1b—"ed," "ing," and "at" endings
- Step 1c—"y" endings
- Step 2—"nounifying" endings such as "ational," "tional," "ence," and "able"
- Step 3—adjective endings such as "icate," b "ful," and "alize"
- Step 4—adjective and noun endings such as "ive," "ible," "ent," and "ism"
- Step 5a—stubborn "e" endings, still hanging around
- Step 5b—trailing double consonants for which the stem will end in a single "l"

3. Tokenizer - narzędzie umożliwiające podzielenie tekstu wejściowego na tokeny, które są fragmentami tekstu. Najpopularniejszą metodą jest podzielenie tekstu na wyrazy, ale istnieją także metody dzielące tekst co kilka liter, niezależnie czy są wyrazami czy nie.

Wykorzystaliśmy Casual Tokenizer z biblioteki nltk. Został on stworzony z myślą o wykorzystaniu w krótkich, nieformalnych tekstach. Jest to dokładnie to czego potrzebowaliśmy, główną częścią naszego projektu były żarty, czyli właśnie krótkie, nieformalne teksty.

4. Część praktyczna

Wczytywanie kategorii i słów kluczowych dla danej kategorii

```
19 file_path_categories = 'categories.csv'
20 categories = []
21 with open(file_path_categories, newline='', encoding='utf-8') as csvfile:
22     csv_reader = csv.reader(csvfile)
23
24     for row in csv_reader:
25         key = row[0]
26         values = row[1:]
27
28         values = [word.Word(stemmer.stem(item.split(':')[0]), int(item.split(':')[1])) for item in values if item != '']
29         categories.append(Category.Category(key, values))
30
```

Struktura pliku categories.csv

kategoria,słowo1:waga1,słowo2:waga2,słowo3:waga3...

Przykład

School,school:4,teacher:3,classroom:4,student:4,homework:4

Klasa Category

```
1 class Category:
2     def __init__(self, name, words):
3         self.name = name
4         self.words = words
```

name - nazwa kategorii

words - lista słów kluczowych wraz z wagami

Klasa Word

```
1 class Word:
2     def __init__(self, word, weight):
3         self.word = word
4         self.weight = weight
```

word - słowo kluczowe

weight - waga z zakresu 1-4

```
values = [word.Word(stemmer.stem(item.split(':')[0]), int(item.split(':')[1])) for item in values if item != ""]
```

Powyższy kod odpowiedzialny jest za rozdzielenie słowa kluczowego i wagi. Rozdzielenie wykonywane jest za pomocą funkcji `split()`. Elementem dzielącym jest znak ":". Dodatkowo dla słowa wykonywana jest operacja stemmingu, aby sprowadzić słowo do podstawowej formy. W tym celu został wykorzystany obiekt klasy `PorterStemmer()`.

Wczytanie żartów z pliku jokes.json

```
34 file_path_jokes = 'jokes.json'
35 jokes = []
36 with open(file_path_jokes, 'r') as jsonfile:
37     jokes_data = json.load(jsonfile)
38
39     for joke_data in jokes_data:
40         category = joke_data["category"]
41         body = joke_data["body"]
42
43         for category_object in categories:
44             if category_object.name == category:
45                 jokes.append(joke.Joke(category, body))
46                 break
```

Struktura pliku jokes.json

```
128 {
129     "body": "Q: What has two legs, and bleeds?\r\n\r\nA: Half a dog!",
130     "category": "Gross",
131     "id": 23,
132     "title": "Two Legs"
133 },
```

Z pliku zostaje wczytany cały żart (`body`) oraz kategoria (`category`), aby można było ją porównać w późniejszym etapie z kategorią, którą otrzymaliśmy z naszego programu.

Nie wszystkie kategorie, które występują w pliku `jokes.json` są obsługiwane przez nasz program, dlatego poniższy kod odpowiedzialny jest za zapisywanie żartów do przetworzenia, które należą do kategorii, które są obsługiwane przez nasz program.

```
43     for category_object in categories:
44         if category_object.name == category:
45             jokes.append(joke.Joke(category, body))
46             break
```

Na tym etapie posiadamy już listę żartów oraz kategorii wraz ze słowami kluczowymi przekształconymi do formy podstawowej.

Naszym następnym celem była normalizacja słów występujących w żartach.

```

57 for joke in jokes:
58     joke.tokens = casual_tokenize(joke.body)
59
60     # Usunięcie znaków interpunkcyjnych i sprawdzenie, czy słowo składa się tylko z liter
61     joke.tokens = [w.strip(string.punctuation) for w in joke.tokens if w.isalpha()]
62
63     # Zamiana wszystkich liter na małe litery
64     joke.tokens = [x.lower() for x in joke.tokens]
65
66     # Usuwanie stop_wordsów
67     joke.tokens = [x for x in joke.tokens if x not in stop_words]
68
69     # Stemming
70     joke.tokens = [stemmer.stem(w) for w in joke.tokens]

```

1. Tokenizacja

```

56     joke.tokens = casual_tokenize(joke.body)

```

W pierwszej kolejności przeprowadzana jest tokenizacja całego żartu, aby podzielić go na tokeny. Wybraliśmy do tego zadania casual tokenizer, ponieważ uważaliśmy go za odpowiedni tokenizator do tej treści.

Uzyskane tokeny:

```

['So', ',', 'this', 'guy', 'walks', 'into', 'a', 'bar', '.', 'And', 'says', ',', 'ouch', 'ouch', '.']
['There's', 'this', 'dyslexic', 'guy', '...', 'he', 'walked', 'into', 'a', 'bra', '...']
['What's', 'the', 'difference', 'between', 'a', 'bad', 'golfer', 'and', 'a', 'bad', 'skydiver', '?', 'A', 'bad', 'golfer']
['A', 'little', 'old', 'lady', 'goes', 'to', 'the', 'doctor', 'and', 'says', 'Doctor', 'I', 'have', 'this', 'pr']
['Doctor', ':', 'Well', 'I', 'hope', 'you', 'enjoy', 'changing', 'diapers', 'Mrs', 'Jones', '?', 'Mrs', 'Jones', ':']
['What', 'do', 'you', 'call', 'a', 'blond', 'with', 'half', 'a', 'brain', '?', 'Gifted', '.']
['Q', ':', 'What', 'has', 'two', 'legs', 'and', 'bleeds', '?', 'A', ':', 'Half', 'a', 'dog', '!']
['An', 'Essex', 'girl', 'walks', 'into', 'the', 'local', 'dry', 'cleaners', 'She', 'places', 'a', 'garment', 'on', '']
['You', 'know', 'you're', 'a', 'redneck', 'if', 'you', 'introduce', 'a', 'friend', 'to', 'your', 'wife', 'and', 'sister', '']
['A', 'patient', 'wakes', 'up', 'after', 'having', 'surgery', 'to', 'remove', 'a', 'gangrenous', 'leg', '.', 'Doctor', ':']

```

Otrzymany rezultat programu na tym etapie:

```

Ilość wszystkich żartów: 2342
Ilość zgodnych kategorii: 1030 (43.98%)

```

2. Usunięcie znaków interpunkcyjnych i ciągów, które nie są słowami

```

60     joke.tokens = [w.strip(string.punctuation) for w in joke.tokens if w.isalpha()]

```

Uzyskane tokeny:

```

['So', 'this', 'guy', 'walks', 'into', 'a', 'bar', 'And', 'says', 'ouch']
['this', 'dyslexic', 'guy', 'he', 'walked', 'into', 'a', 'bra']
['the', 'difference', 'between', 'a', 'bad', 'golfer', 'and', 'a', 'bad', 'skydiver', 'A', 'bad', 'golfer', 'goes', 'whack', 'dang', 'A']
['A', 'little', 'old', 'lady', 'goes', 'to', 'the', 'doctor', 'and', 'says', 'Doctor', 'I', 'have', 'this', 'problem', 'with', 'wind', '']
['Doctor', 'Well', 'I', 'hope', 'you', 'enjoy', 'changing', 'diapers', 'Mrs', 'Jones', 'Mrs', 'Jones', 'Why', 'Am', 'I', 'pregnant', 'Do']
['What', 'do', 'you', 'call', 'a', 'blond', 'with', 'half', 'a', 'brain', 'Gifted']
['Q', 'What', 'has', 'two', 'legs', 'and', 'bleeds', 'A', 'Half', 'a', 'dog']
['An', 'Essex', 'girl', 'walks', 'into', 'the', 'local', 'dry', 'cleaners', 'She', 'places', 'a', 'garment', 'on', 'the', 'counter', 'be']
['You', 'know', 'a', 'redneck', 'if', 'you', 'introduce', 'a', 'friend', 'to', 'your', 'wife', 'and', 'sister', 'and', 'he', 'only', 'has']
['A', 'patient', 'wakes', 'up', 'after', 'having', 'surgery', 'to', 'remove', 'a', 'gangrenous', 'leg', 'Doctor', 'I', 'have', 'good', '']

```

Otrzymany rezultat programu na tym etapie:

```

Ilość wszystkich żartów: 2342
Ilość zgodnych kategorii: 1030 (43.98%)

```

Wniosek:

Przeprowadzenie tej operacji nie wpłynęło na zwiększenie ilości dopasowanych kategorii,

ale można zauważyć, że zmniejszyła się ilość tokenów, dzięki czemu następne operacje będą wykonywać się szybciej. Gdybyśmy również chcieli zapisać uzyskane tokeny do pliku, to dzięki tej operacji, plik miałby mniejszy rozmiar.

3. Zamiana wszystkich liter na małe litery:

```
63 joke.tokens = [x.lower() for x in joke.tokens]
```

Uzyskane tokeny:

```
['so', 'this', 'guy', 'walks', 'into', 'a', 'bar', 'and', 'says', 'ouch']  
['this', 'dyslexic', 'guy', 'he', 'walked', 'into', 'a', 'bra']  
['the', 'difference', 'between', 'a', 'bad', 'golfer', 'and', 'a', 'bad', 'skydiver', 'a', 'bad', 'golfer', 'goes', 'whack', 'dang', 'a', 'bad',  
['a', 'little', 'old', 'lady', 'goes', 'to', 'the', 'doctor', 'and', 'says', 'doctor', 'i', 'have', 'this', 'problem', 'with', 'wind', 'but', 'it  
['doctor', 'well', 'i', 'hope', 'you', 'enjoy', 'changing', 'diapers', 'mrs', 'jones', 'mrs', 'jones', 'why', 'am', 'i', 'pregnant', 'doctor', 't  
['what', 'do', 'you', 'call', 'a', 'blond', 'with', 'half', 'a', 'brain', 'gifted']  
['q', 'what', 'has', 'two', 'legs', 'and', 'bleeds', 'a', 'half', 'a', 'dog']  
['an', 'essex', 'girl', 'walks', 'into', 'the', 'local', 'dry', 'cleaners', 'she', 'places', 'a', 'garment', 'on', 'the', 'counter', 'be', 'back  
['you', 'know', 'a', 'redneck', 'if', 'you', 'introduce', 'a', 'friend', 'to', 'your', 'wife', 'and', 'sister', 'and', 'he', 'only', 'has', 'to',  
['a', 'patient', 'wakes', 'up', 'after', 'having', 'surgery', 'to', 'remove', 'a', 'gangrenous', 'leg', 'doctor', 'i', 'have', 'good', 'news', 't
```

Otrzymany rezultat programu na tym etapie:

```
Ilość wszystkich żartów: 2342  
Ilość zgodnych kategorii: 1169 (49.91%)
```

Wniosek:

Przeprowadzenie tej operacji nieznacznie zwiększyło współczynnik dopasowań. Jest to ważny etap, który należało przeprowadzić przed następnymi etapami, abyśmy mieli pewność, że będziemy pracować tylko na małych literach.

4. Usuwanie stop wordsów

```
53 nltk.download('stopwords')
```

```
54 stop_words = nltk.corpus.stopwords.words('english')
```

```
63 joke.tokens = [x for x in joke.tokens if x not in stop_words]
```

Uzyskane tokeny:

```
['guy', 'walks', 'bar', 'says', 'ouch']  
['dyslexic', 'guy', 'walked', 'bra']  
['difference', 'bad', 'golfer', 'bad', 'skydiver', 'bad', 'golfer', 'goes', 'whack', 'dang', 'bad', 'skydiver', 'goes', 'dang', 'whack']  
['little', 'old', 'lady', 'goes', 'doctor', 'says', 'doctor', 'problem', 'wind', 'really', 'bother', 'much', 'never', 'smell', 'always', 'si  
['doctor', 'well', 'hope', 'enjoy', 'changing', 'diapers', 'mrs', 'jones', 'mrs', 'jones', 'pregnant', 'doctor', 'bowel', 'cancer']  
['call', 'blond', 'half', 'brain', 'gifted']  
['q', 'two', 'legs', 'bleeds', 'half', 'dog']  
['essex', 'girl', 'walks', 'local', 'dry', 'cleaners', 'places', 'garment', 'counter', 'back', 'tomorrow', 'afternoon', 'pick', 'dress', 'sa  
['know', 'redneck', 'introduce', 'friend', 'wife', 'sister', 'shake', 'one', 'hand']  
['patient', 'wakes', 'surgery', 'remove', 'gangrenous', 'leg', 'doctor', 'good', 'news', 'bad', 'news', 'patient', 'bad', 'news', 'doctor',
```

Otrzymany rezultat programu na tym etapie:

```
Ilość wszystkich żartów: 2342  
Ilość zgodnych kategorii: 1169 (49.91%)
```

Wniosek:

Przeprowadzenie tej operacji, nie wpłynęło na zwiększenie ilości dopasowań, ale pomogło zredukować ilość tokenów.

5. Stemming

```
15 stemmer = PorterStemmer()
```

70

```
joke.tokens = [stemmer.stem(w) for w in joke.tokens]
```

Uzyskane tokeny:

```
['guy', 'walk', 'bar', 'say', 'ouch']
['dyslex', 'guy', 'walk', 'bra']
['differ', 'bad', 'golfer', 'bad', 'skydiv', 'bad', 'golfer', 'goe', 'whack', 'dang', 'bad', 'skydiv', 'goe', 'dang', 'whack']
['littl', 'old', 'ladi', 'goe', 'doctor', 'say', 'doctor', 'problem', 'wind', 'realli', 'bother', 'much', 'never', 'smell', 'al']
['doctor', 'well', 'hope', 'enjoy', 'chang', 'diaper', 'mr', 'jone', 'mr', 'jone', 'pregnant', 'doctor', 'bowel', 'cancer']
['call', 'blond', 'half', 'brain', 'gift']
['q', 'two', 'leg', 'bleed', 'half', 'dog']
['essex', 'girl', 'walk', 'local', 'dri', 'cleaner', 'place', 'garment', 'counter', 'back', 'tomorrow', 'afternoon', 'pick', 'd']
['know', 'redneck', 'introduc', 'friend', 'wife', 'sister', 'shake', 'one', 'hand']
['patient', 'wake', 'surgeri', 'remov', 'gangren', 'leg', 'doctor', 'good', 'news', 'bad', 'news', 'patient', 'bad', 'news', 'd']
```

Otrzymany rezultat programu na tym etapie:

```
Ilość wszystkich żartów: 2342
Ilość zgodnych kategorii: 1723 (73.57%)
```

Wniosek:

Przeprowadzenie stemmingu bardzo pozytywnie wpłynęło na ilość dopasowań

Naszym następnym celem było dobranie kategorii dla żartów na podstawie słów kluczowych oraz wag.

```
75 for joke in jokes:
76     cat_weight = []
77
78     for category in categories:
79         weight = 0
80
81         for joke_word in joke.tokens:
82             for category_word_obj in category.words:
83                 if joke_word == category_word_obj.word:
84                     weight = weight + category_word_obj.weight
85
86         cat_weight.append(weight)
87
88         index = 0
89         for i in range(len(cat_weight)):
90             if cat_weight[i] > cat_weight[index]:
91                 index = i
92
93         if cat_weight[index] != 0:
94             joke.our_category = categories[index]
95         else:
96             joke.our_category = None
```

Powyższy kod przechodzi przez wszystkie żarty i dla każdego żartu przechodzi przez wszystkie kategorie oraz oblicza sumę uzyskanej wagi.

Następnie wyszukiwana jest kategoria, która posiada największą liczbę uzyskanych wag i jest przydzielana do naszego żartu.

```
87     index = 0
88     for i in range(len(cat_weight)):
89         if cat_weight[i] > cat_weight[index]:
90             index = i
91
92     if cat_weight[index] != 0:
93         joke.our_category = categories[index]
94     else:
95         joke.our_category = None
```

5. Podsumowanie

Projekt osiągnął swoje główne założenie, czyli stworzenie efektywnego programu, który jest w stanie precyzyjnie przyporządkować kategorię do podanego żartu. Dzięki innowacyjnemu podejściu do analizy treści oraz elastycznej strukturze programu, nasza aplikacja nie tylko spełniła pierwotny cel, ale również otworzyła drzwi do szerszego spektrum zastosowań.

Warto zaznaczyć, że nasz program, bazując na kategorii, słowach kluczowych i ich wagach, ma potencjał do adaptacji do różnych scenariuszy. Po dokonaniu odpowiednich modyfikacji, możliwe jest wykorzystanie go do identyfikacji głównego tematu wypowiedzi lub tekstu, co podkreśla jego wszechstronność i elastyczność.

Otrzymaliśmy zgodność naszych kategorii z kategoriami odczytanymi ze strony na poziomie 70%.

6. Bibliografia

Źródło danych: <https://github.com/taivop/joke-dataset/tree/master>

Dr Radosław Kycia, Wykład 7, Przetwarzanie Języka Naturalnego

<https://pl.wikipedia.org/wiki/Stemming>

<https://github.com/jedijulia/porter-stemmer/blob/master/stemmer.py>

<https://ermlab.com/blog/technicznie/wyszukiwanie-danych-w-czasie-rzeczywistym-z-wykorzystaniem-elasticsearch/>