Лабораторна робота №7, Обробка та аналіз текстових даних на Python, Варіант 14

Виконав: студент групи ІП-11, Лошак Віктор Іванович
Перевірив: Юлія Тимофєєва Сергіївна

Тема роботи: Навчання моделей spaCy
Мета роботи: Ознайомитись з додаванням власних прикладів до моделей spaCy та компонентом для класифікації текстів.

24.05.2024

Завдання:

1. Створити кілька своїх прикладів у форматі json за тематикою:
   авіарейси(англійською або українською мовою) для розпізнавання нового типу сутностей (обрати самостійно). створити програму, що додає ці приклади до існуючої моделі spaCy, та навчає модель. Продемонструвати роботу. Для цього завдання використовуватимемо англійську мову.
2. Застосувати компонент TextCategorizer для визначення намірів. Дані для навчання за тематикою варіанту обрати самостійно або скористатись вказаним файлом (utterance містить висловлювання, intent - намір). Дані файли містять приклади діалогів користувачів з системою-помічником за певною тематикою, наприклад, замовлення квитків і т.д. Навчити компонент та продемонструвати роботу.

Task:

1. Create several of your own examples in json format on the topic: flights

(in English or Ukrainian) to recognize the new type entities (choose independently). Create a program that adds these examples to the existing spaCy model, and trains the model. Demonstrate execution of the program. We will use english for this task. 2. Apply the TextCategorizer component to determine intent. Provide your own training data on the given topic(flights) or use flights.json(utterance contains statements, intent - intention). Given files contain examples of user dialogs with the helper system on a certain topic, for example, ticket purchase, etc. Train the component and demonstrate execution of the program..

Task 1

In [ ]:
```
import spacy
from spacy.tokens import DocBin
from spacy.training import Example
import random
import json
import warnings
```

```
warnings.filterwarnings('ignore')
nlp = spacy.load("en_core_web_sm")
```

In [ ]:
```
with open('flight_number_data.json', 'r') as f:
    data = json.load(f)

trainset = [
    (item['text'], {'entities': [(ent['start'], ent['end'], ent['label']) for en
    for item in data
]

for entry in trainset:
    print(entry)
```

```
('I need to check-in for flight UA345 leaving tomorrow.', {'entities': [(31, 36,
'FLIGHT_NUMBER')]})
('Is flight DL456 delayed?', {'entities': [(9, 14, 'FLIGHT_NUMBER')]})
('Can I get an update on AA123?', {'entities': [(22, 27, 'FLIGHT_NUMBER')]})
('What gate is BA432 departing from?', {'entities': [(13, 18, 'FLIGHT_NUMBER')]})
('Has flight LH678 landed yet?', {'entities': [(9, 14, 'FLIGHT_NUMBER')]})
('Please confirm my seat on VX321.', {'entities': [(26, 31, 'FLIGHT_NUMBER')]})
('Reservation details for flight AC980 are needed.', {'entities': [(28, 33, 'FLIG
HT_NUMBER')]})
('I missed my flight QR404, what are my options?', {'entities': [(17, 22, 'FLIGHT
_NUMBER')]})
('Is there a meal on flight KE567?', {'entities': [(23, 28, 'FLIGHT_NUMBER')]})
('How do I get to flight SQ738 now?', {'entities': [(20, 25, 'FLIGHT_NUMBER')]})
('I want to cancel my booking on flight CX921.', {'entities': [(35, 40, 'FLIGHT_N
UMBER')]})
('What time does flight EK302 depart?', {'entities': [(19, 24, 'FLIGHT_NUMBE
R')]})
('Flight TK403 has been rescheduled.', {'entities': [(7, 12, 'FLIGHT_NUMBER')]})
('Please book me on the next flight, ideally AF1234.', {'entities': [(46, 52, 'FL
IGHT_NUMBER')]})
('Was flight NZ885 on time today?', {'entities': [(8, 13, 'FLIGHT_NUMBER')]})
```

We can observe that many pipes which are not required for the task are included into the pipeline.

In [ ]:
```
nlp.pipe_names
```

Out[ ]:
```
['tok2vec', 'tagger', 'parser', 'attribute_ruler', 'lemmatizer', 'ner']
```

In [ ]:
```
other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']
epochs = 20

with nlp.disable_pipes(*other_pipes):
    optimizer = nlp.create_optimizer()
    for i in range(epochs):
        random.shuffle(trainset)
        losses = {}
        for text, annotations in trainset:
            doc = nlp.make_doc(text)
            example = Example.from_dict(doc, annotations)
            nlp.update([example], drop=0.5, sgd=optimizer, losses=losses)
        print(f"Losses at iteration {i}: {losses}")

ner = nlp.get_pipe("ner")
ner.to_disk(r'C:\Users\vikto\Workspace\GitRepos\KPI\NLP\lab7')
```

```
Losses at iteration 0: {'ner': 8.198539205385648}
Losses at iteration 1: {'ner': 5.816164242863161}
Losses at iteration 2: {'ner': 4.830822096031077}
Losses at iteration 3: {'ner': 5.145288105843724}
Losses at iteration 4: {'ner': 4.9629774651041325}
Losses at iteration 5: {'ner': 4.190643396153424}
Losses at iteration 6: {'ner': 4.135249327713489}
Losses at iteration 7: {'ner': 4.7103573122109825}
Losses at iteration 8: {'ner': 3.56356409730847}
Losses at iteration 9: {'ner': 1.9511685934992546}
Losses at iteration 10: {'ner': 0.7691909895630822}
Losses at iteration 11: {'ner': 0.9344228917571324}
Losses at iteration 12: {'ner': 0.010430016615196068}
Losses at iteration 13: {'ner': 0.001498123025870885}
Losses at iteration 14: {'ner': 0.00115349515220772}
Losses at iteration 15: {'ner': 0.023137258718527873}
Losses at iteration 16: {'ner': 1.8658484620242685e-05}
Losses at iteration 17: {'ner': 2.66160134264602e-05}
Losses at iteration 18: {'ner': 0.004479485865791526}
Losses at iteration 19: {'ner': 3.290591965246148e-06}
```

Despite fairly small size of training data the accuracy is satisfactory.

In [ ]:
```
test_text = "How many times do I have to tell you? PM572 is delayed, come home t
doc = nlp(test_text)
print("Entities in '%s'" % test_text)
for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
Entities in 'How many times do I have to tell you? PM572 is delayed, come home to
Olive Street PT15'
PM572 FLIGHT_NUMBER
```

Task 2

In [ ]:
```
import spacy
import json
from spacy.util import minibatch, compounding
from spacy.training import Example

with open("flights.json") as file:
    data = json.load(file)
```

In [ ]:
```
train_data = []
for dialogue in data:
    for turn in dialogue["turns"]:
        if turn["speaker"] == "USER":
            utterance = turn["utterance"]
            intent = turn["frames"][0]["state"]["active_intent"]
            train_data.append((utterance, intent))

train_data[:10]
```

Out[ ]:
```
[('I want to find a one way flight.', 'SearchOnewayFlight'),
 ('I am flying to Phoenix, AZ. I want 1 ticket from Seattle, WA.',
  'SearchOnewayFlight'),
 ("I want to depart 11th of March. I don't care which airline.",
  'SearchOnewayFlight'),
 ('A different airline please, in Premium Economy.', 'SearchOnewayFlight'),
 ('Find me something else. I want to fly with Southwest Airlines. Look for them
from Atlanta.',
  'SearchOnewayFlight'),
 ('Alright.', 'SearchOnewayFlight'),
 ('No thanks for your help.', 'NONE'),
 ('I would like to find a one way flight. I am interested in flights from Vega
s.',
  'SearchOnewayFlight'),
 ('I will be heading to Toronto, Canada. I would like to start my travel on the
13th of March.',
  'SearchOnewayFlight'),
 ('What is the airport the flight will be landing at?', 'SearchOnewayFlight')]
```

In [ ]:
```
set([element[1] for element in train_data]), len(train_data)
```

Out[ ]:
```
({'NONE', 'ReserveOnewayFlight', 'SearchOnewayFlight'}, 744)
```

In [ ]:
```
nlp = spacy.blank("en")

textcat = nlp.add_pipe("textcat", last=True)
textcat.add_label("SearchOnewayFlight")
textcat.add_label("NONE")
textcat.add_label("ReserveOnewayFlight")
```

Out[ ]:
```
1
```

We need to convert the data into the format spaCy accepts before proceeding.

In [ ]:
```
train_examples = []
for text, label in train_data:
    cat = {'cats': {
        "SearchOnewayFlight": label == "SearchOnewayFlight",
        "NONE": label == "NONE",
        "ReserveOnewayFlight": label == "ReserveOnewayFlight"
    }}
    train_examples.append(Example.from_dict(nlp.make_doc(text), cat))
```

In [ ]:
```
optimizer = nlp.begin_training()
for i in range(15):
    losses = {}
    batches = minibatch(train_examples, size=compounding(4., 32., 1.001))
    for batch in batches:
        nlp.update(batch, sgd=optimizer, drop=0.5, losses=losses)
    print(f"Losses at iteration {i}: {losses}")
```

```
Losses at iteration 0: {'textcat': 31.91867504242873}
Losses at iteration 1: {'textcat': 34.66760695278206}
Losses at iteration 2: {'textcat': 35.396802616697116}
Losses at iteration 3: {'textcat': 34.29862304196533}
Losses at iteration 4: {'textcat': 30.782300744441606}
Losses at iteration 5: {'textcat': 29.546233219329565}
Losses at iteration 6: {'textcat': 26.898122131081905}
Losses at iteration 7: {'textcat': 26.116174917600972}
Losses at iteration 8: {'textcat': 23.246139856456722}
Losses at iteration 9: {'textcat': 19.77220083573154}
Losses at iteration 10: {'textcat': 19.64046903320241}
Losses at iteration 11: {'textcat': 18.357357815298116}
Losses at iteration 12: {'textcat': 16.028874019542968}
Losses at iteration 13: {'textcat': 14.493248600514107}
Losses at iteration 14: {'textcat': 13.161071026858847}
```

In [ ]:
```
test_text = "I need a flight from New York to London next Monday."
doc = nlp(test_text)
print(test_text, doc.cats)
```

```
I need a flight from New York to London next Monday. {'SearchOnewayFlight': 0.000
9650694555602968, 'NONE': 2.225606112915557e-05, 'ReserveOnewayFlight': 0.9990127
086639404}
```

Висновок:

В ході виконання даної лабораторної роботи я ознайомився з основами роботи з бібліотекою spaCy для обробки та аналізу текстових даних. Зокрема, я навчився створювати власні набори даних у форматі JSON для розпізнавання нових типів сутностей і використовувати ці дані для навчання моделі NER (Named Entity Recognition). Це дозволило розширити функціональність стандартної моделі, додаючи здатність виявляти і класифікувати номери рейсів у текстах.

Крім того, я застосував компонент TextCategorizer для класифікації текстів за намірами користувачів. Цей компонент допоміг мені структурувати діалоги користувачів зі штучним інтелектом для різних цілей, таких як пошук або бронювання рейсів. Я навчив модель розпізнавати різні типи запитів, базуючись на вмісті утерансів користувачів, що може бути корисним для подальшої розробки чат-ботів або систем автоматичної відповіді.