Лабораторна робота №8

Синтаксичні залежності у spaCy

Мета роботи: Ознайомитись з використанням класу Matcher. Ознайомитись із синтаксичними залежностями та їх застосуванням для виявлення намірів.

Короткі теоретичні відомості

Моделі spaCy дуже успішні для загальних цілей обробки природньої, таких як розуміння синтаксису речення, розбиття абзацу на речення та вилучення деяких сутностей. Однак іноді потрібно працювати над дуже специфічними доменами, які моделі spaCy не бачили під час навчання.

Вилучення інформації на основі правил є незамінним для будь-якого конвеєра обробки природньої мови. Певні типи об’єктів, як-от час, дата та номери телефонів, мають різні формати, які можна розпізнати за набором правил, без необхідності навчання статистичних моделей.

Клас Matcher може зіставляти попередньо визначені правила з послідовністю токенів в об’єктах Doc і Span; крім того, правила можуть посилатися на лексему або її лінгвістичні атрибути.

```
import spacy
from spacy.matcher import Matcher
doc = nlp("Good morning, I want to reserve a ticket.")
matcher = Matcher(nlp.vocab)
pattern = [{"LOWER": "good"},{"LOWER": "morning"},{"IS_PUNCT": True}]
matcher.add("morningGreeting", [pattern])
matches = matcher(doc)
for match_id, start, end in matches:
  m_span = doc[start:end]
  print(start, end, m_span.text)
```

ORTH і TEXT подібні до LOWER: вони означають точну відповідність тексту токена, включаючи регістр. LENGTH використовується для визначення довжини токена. Наступний блок атрибутів токенів — IS_ALPHA, IS_ASCII і IS_DIGIT. Ці функції зручні для пошуку символів чисел і звичайних слів. IS_PUNCT, IS_SPACE і IS_STOP зазвичай використовуються в шаблонах, які включають деякі допоміжні токени та відповідають токенам пунктуації, пробілу та стоп-слова. IS_SENT_START є ще одним корисним атрибутом; він відповідає лексемам початку речення. LIKE_NUM, LIKE_URL і LIKE_EMAIL – це атрибути, які відповідають токенам, які виглядають як числа, URL-адреси та електронні адреси.

Клас Matcher пропонує еквівалент операцій з регулярними виразами. Найпоширенішими операціями з регулярними виразами є необов’язковий збіг (?), збіг принаймні один раз (+) і збіг 0 або більше разів (*), заперечення (!).

spaCy пропонує рішення для порівняння тексту з довгими словниками – клас PhraseMatcher.

```
from spacy.matcher import PhraseMatcher
matcher = PhraseMatcher(nlp.vocab, attr="SHAPE")
ip_nums = ["127.0.0.0", "127.256.0.0"]
patterns = [nlp.make_doc(ip) for ip in ip_nums]
matcher.add("IPNums",patterns)
doc = nlp("This log contains the following IP addresses: 192.1.1.1 and 192.12.1.1 and 192.160.1.1 .")
for mid, start, end in matcher(doc):
   print(start, end, doc[start:end])
```

Також за допомогою класу Matcher можна шукати висловлювання за певним шаблоном.

Наприклад, шаблони для деяких висловлювань користувача діалогової системи, що не містять намірів:

```
[{"LOWER": {"IN": ["no", "nope"]}}, {"TEXT": {"IN": [",", "."]}}]
[{"TEXT": {"REGEX": "[Tt]hanks?"}}, {"LOWER": {"IN": ["you", "a lot"]}, "OP": "*"}]
[{"LOWER": {"IN": ["that", "that's", "thats", "that'll","thatll"]}}, {"LOWER": {"IN": ["is", "will"]}, "OP": "*"}, {"LOWER": "all"}]
```

Розбір залежностей пов’язаний з аналізом структур речень через залежності між лексемами. Синтаксичний аналізатор залежностей позначає синтаксичні зв’язки між лексемами речення та з’єднує синтаксично пов’язані пари лексем. Залежність або відношення залежності - це спрямований зв'язок між двома токенами. Результатом розбору залежностей завжди є дерево.

spaCy призначає кожному токену мітку залежності, так само як і з іншими лінгвістичними властивостями, такими як лема або тег частини мови. spaCy показує відносини залежності з спрямованими дугами.

Мітка залежності описує тип синтаксичного відношення між двома лексемами наступним чином: одна з лексем є синтаксичним батьком (HEAD), а інша є його залежною (CHILD).

```
for token in doc:
   print(token.text, token.dep_)
```

Можна візуалізувати залежності за допомогою displacy

```
from spacy import displacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("I have a cat")
displacy.render(doc, style='dep’)
```

І клас Matcher, і дерева синтаксичної залежності можуть використовуватись для того, щоб виділити певні сутності, наприклад, для діалогової системи. Також за допомогою синтаксичних залежностей можна визначати наміри користувача.

Наприклад, якщо є запити на замовлення авіарейсів, то з них можна виділити пункти відправлення та призначення за допомогою класу Matcher.

```
import spacy
from spacy.matcher import Matcher
 nlp = spacy.load("en_core_web_md")
matcher = Matcher(nlp.vocab)
pattern = [{"POS": "ADP"}, {"ENT_TYPE": "GPE"}]
matcher.add("prepositionLocation", [pattern])
doc = nlp("show me flights from denver to boston on tuesday")
matches = matcher(doc)for mid, start, end in matches:
   print(doc[start:end])
```

Для виділення певних сутностей можна використати синтаксичні залежності: переглянути дерево залежностей. Перегляд дерева залежностей означає відвідування токенів у спеціальному порядку, не обов'язково зліва направо. ROOT — це спеціальна мітка залежності, яка завжди присвоюється головному дієслову речення. spaCy показує синтаксичні зв'язки з дугами.

Таким чином, перебираючи висловлювання користувача діалогової системи і будуючи дерева залежностей, можна знайти синтаксичний зв'язок між словами, що не стоять поруч у реченні.

Наприклад, у реченні «I'm going to a meeting in London» можна виявити, що прийменник to має зв'язок з Лондоном.

```
text = ("I'm going to a meeting in London")
text_doc = nlp(text)
displacy.render(text_doc, style='dep')
```

За допомогою синтаксичних залежностей також можна визначати наміри. Зазвичай, наміри включають певну дію (виражену дієсловом) та об'єкт, до якого вона застосовується: знайти ресторан, забронювати квиток і т.д.

Для розпізнавання наміру до висловлювання користувача застосовуються такі кроки:

1.Поділ речення на токени.

2.Розбір залежностей. Далі потрібно виконати перехід по дереву залежностей, щоб витягти токени та потрібні відносини, тобто дієслово та його об'єкт.

```
doc = nlp("find a flight from London to Kyiv")
for token in doc:
        if token.dep_ == "dobj":
                print(token.head.text + token.text.capitalize())
```

У одному реченні може буде декілька намірів (виражених дієсловами), що будуть об'єднані сполучником.

```
doc = nlp("show all flights and fares from London to Kyiv ")
for token in doc:
        if token.dep_ == "dobj":
                dobj = token.text
```

```
        conj = [t.text for t in token.conjuncts]
        verb = donj.head
    print(verb, dobj, conj)
```

Завдання до лабораторної роботи

1. Використати клас Matcher для виділення сутностей відповідно до варіанту. Використати файл з 7-ї лабораторної роботи (не обов’язково усі висловлювання). Продемонструвати роботу.
2. Застосувати синтаксичні залежності для визначення намірів. Використати файл з 7-ї лабораторної роботи (не обов’язково усі висловлювання).

Оформити звіт. Звіт повинен містити:

— титульний лист;
— код програми;
— результати виконання коду;

Продемонструвати роботу програми та відповісти на питання стосовно теоретичних відомостей та роботи програми.