**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи №8 з дисципліни

«Обробка та аналіз текстових даних на Python»

„Синтаксичні залежності у spaCy”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-11 Головня Олександр Ростиславович*

**Перевірив**

(прізвище, ім'я, по батькові)

Київ 2024

**Мета роботи:** Ознайомитись з використанням класу Matcher. Ознайомитись

із синтаксичними залежностями та їх застосуванням для виявлення намірів.

**Завдання до лабораторної роботи:**

Тематика: події (наприклад, спортивні). Файл events.json.

1. Виділити назви спортивних команд за допомогою класу Matcher. Виділити висловлювання користувача, що є підтвердженнями (наприклад, Awesome, sounds great!), за допомогою шаблонів.
2. Використати клас Matcher для виділення сутностей відповідно до варіанту Продемонструвати роботу.
3. Застосувати синтаксичні залежності для визначення намірів.

**Для початку я завантажив файл json, завантажив модель spaCy, ініцілізував клас Matcher**

with open('events.json', 'r') as f:

    data = json.load(f)

nlp = spacy.load('en\_core\_web\_sm')

matcher = Matcher(nlp.vocab)

**Далі я створив два шаблона для визначення команд та варіантів підтверджень, додав їй до Matcher:**

team\_patterns = [

    [{"LOWER": {"IN": ["fc", "team", "club"]}}, {"POS": "PROPN"}],

    [{"POS": "PROPN"}, {"LOWER": "vs"}, {"POS": "PROPN"}],

]

# Додавання шаблонів для різних варіантів підтверджень

confirm\_patterns = [

    [{"LOWER": {"IN": ["awesome", "great", "fantastic", "amazing"]}}],

    [{"LOWER": {"IN": ["sounds", "looks", "seems", "feels"]}}, {"LOWER": "good"}],

    [{"LOWER": {"IN": ["sounds", "looks", "seems", "feels"]}}, {"LOWER": "great"}],

]

**Створив дві функції для виділення команд та підтверджень, а також для виділення намірів:**

def extract\_entities(text):

    doc = nlp(text)

    matches = matcher(doc)

    entities = []

    confirms = []

    for match\_id, start, end in matches:

        if nlp.vocab.strings[match\_id] == "TEAM":

            entities.append(doc[start:end].text)

        elif nlp.vocab.strings[match\_id] == "CONFIRM":

            confirms.append(doc[start:end].text)

    return entities, confirms

def extract\_intents(text):

    doc = nlp(text)

    intents = []

    for token in doc:

        if token.dep\_ == "dobj":

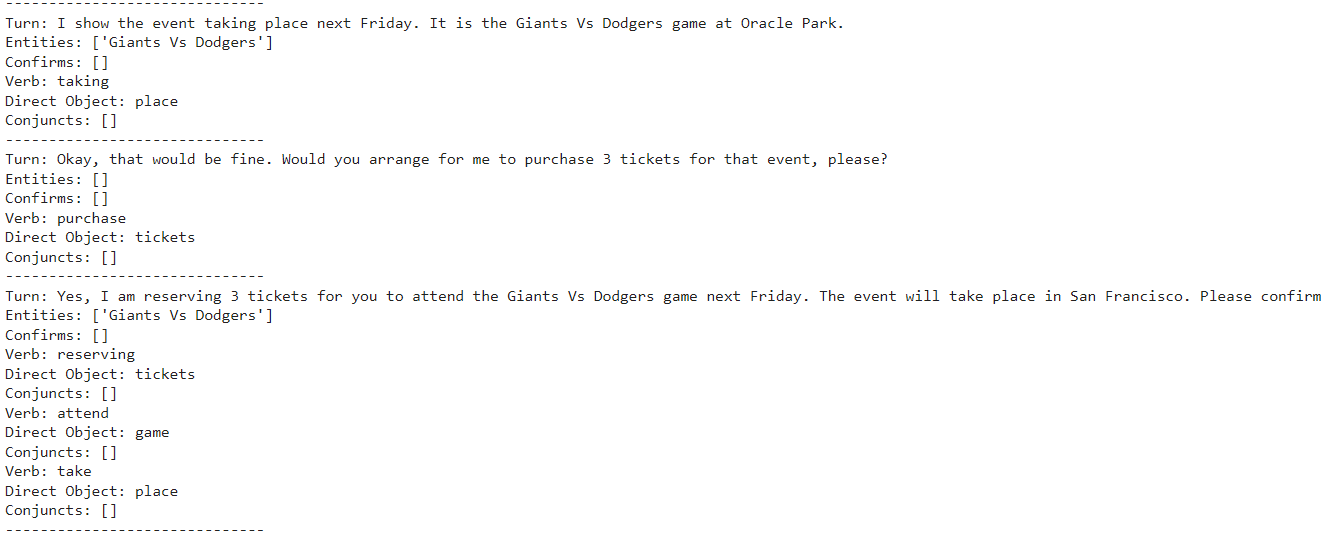
            dobj = token.text

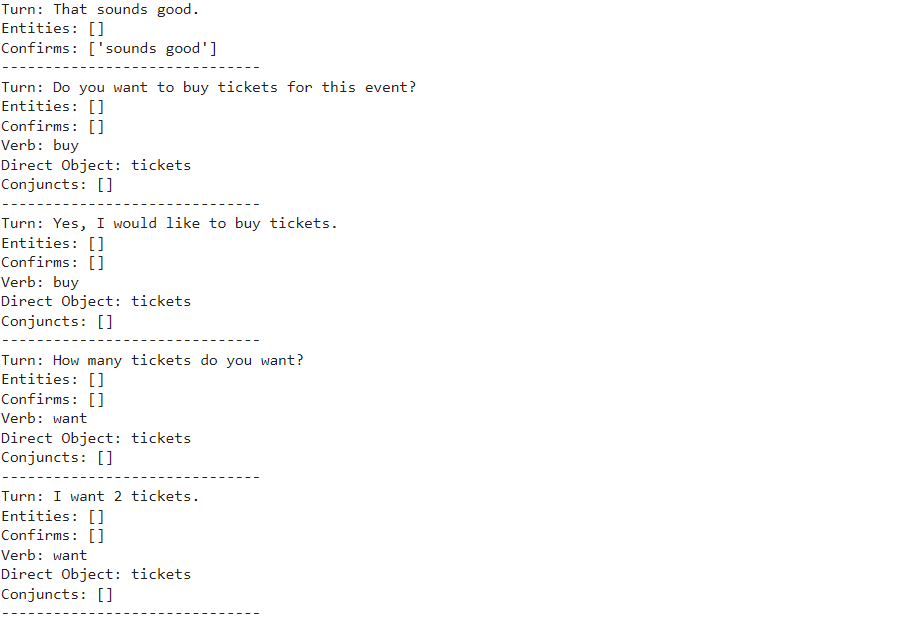
            verb = token.head.text

            conjuncts = [t.text for t in token.head.children if t.dep\_ == "conj"]

            intents.append((verb, dobj, conjuncts))

    return intents

Використав і вивів результат:  




**Висновок:** Під час розробки лабораторної роботи, я ознайомився на практиці з використанням класу Matcher. Ознайомився із синтаксичними залежностями та їх застосуванням для виявлення намірів.

**Код програми:**

import spacy

nlp = spacy.load("en\_core\_web\_sm")

text = [

    'The Academy of Country Music (ACM) honored outstanding performers and songwriters who have significantly contributed to the evolution of country music.'

    'The Jazz Foundation of America celebrated the career of iconic saxophonist Wayne Shorter with a Lifetime Achievement Award, recognizing his impact on the jazz world.'

    'The Montreux Jazz Festival in Switzerland drew musicians from various genres to engage in workshops, jam sessions, and international competitions.'

    'The Boston Symphony Orchestra, led by maestro Andris Nelsons, presented an innovative concert series that blended classical symphonies with contemporary dance performances.'

]

for sent in text:

    doc = nlp(sent)

    for ent in doc.ents:

        print(ent.text, ent.label\_)

import random

from spacy.training import Example

nlp = spacy.blank("en")

ruler = nlp.add\_pipe("entity\_ruler")

patterns = [

    {"label": "ORG", "pattern": "ACM"},

    {"label": "GPE", "pattern": "Switzerland"},

    {"label": "PERSON", "pattern": "Andris Nelsons"},

    {"label": "PERSON", "pattern": "Wayne Shorter"},

]

ruler.add\_patterns(patterns)

for sentence in text:

    doc = nlp(sentence)

    for ent in doc.ents:

        print(ent.text, ent.label\_)

import spacy

from spacy.training import Example

# Приклади даних для навчання

train\_data = [

    ("I want to order two tickets for tomorrow's concert.", "ticket\_order"),

    ("Can you provide me with more information about this event?", "event\_info"),

    ("No, I'm not interested. Thanks anyway.", "not\_interested"),

    ("Yes, that sounds good. Please proceed with the booking.", "proceed\_booking"),

    ("Thank you for your help. Goodbye!", "thanks\_goodbye"),

]

# Ініціалізація моделі spaCy

nlp = spacy.blank("en")

textcat = nlp.add\_pipe("textcat")

textcat.add\_label("ticket\_order")

textcat.add\_label("event\_info")

textcat.add\_label("proceed\_booking")

textcat.add\_label("thanks\_goodbye")

textcat.add\_label("not\_interested")

# Навчання моделі

optimizer = nlp.begin\_training()

for i in range(20):

    losses = {}

    for text, label in train\_data:

        doc = nlp.make\_doc(text)

        example = Example.from\_dict(doc, {"cats": {label: 1.0}})

        nlp.update([example], sgd=optimizer, losses=losses)

    print(f"Epoch {i+1}, Loss: {losses}")

# Демонстрація роботи моделі на прикладах

test\_data = [

    "I would like to order two tickets for tomorrow's concert.",

    "Can you provide me with more information about this event?",

    "No, I'm not interested. Thanks anyway.",

    "Yes, that sounds good. Please proceed with the booking.",

    "Thank you for your help. Goodbye!",

]

for text in test\_data:

    doc = nlp(text)

    intent = max(doc.cats, key=doc.cats.get)

    print(f"Utterance: '{text}'\nPredicted intent: {intent}\n")