Библиотека spaCy

Мати Рейнович Пентус, Алексей Андреевич Сорокин

МГУ им. М. В. Ломоносова
весенний семестр 2022/2023 учебного года
Межфакультетский курс
“Введение в компьютерную лингвистику”

Установка библиотеки spaCy

```
pip install -U spacy
python -m spacy download en_core_web_sm
python -m spacy download en_core_web_lg
python -m spacy download ru_core_news_sm
python -m spacy download ru_core_news_lg
```

Подробнее:
https://spacy.io/usage

Библиотека spaCy

```
import spacy
nlp = spacy.load("ru_core_news_sm")
doc = nlp("Клара украла у Карла красные кораллы.")
print("п.номер слово   лемма  ч.речи  тип зав. гл.слово")
print("-------+-------+-------+-------+-------+--------------")
for tt in doc:
    print(tt.i, tt.text, tt.lemma_, tt.pos_, tt.dep_,
          tt.head.i, tt.head.text, sep="\t")
```

```
п.номер слово   лемма  ч.речи  тип зав. гл.слово
-------+-------+-------+-------+-------+--------------
0       Клара   клара   PROPN   nsubj   1       украла
1       украла  украсть VERB    ROOT    1       украла
2       у       у       ADP     case    3       Карла
3       Карла   карл    PROPN   obl     1       украла
4       красные красный ADJ     amod    5       кораллы
5       кораллы коралл  NOUN    obj     1       украла
6       .       .       PUNCT   punct   1       украла
```

Библиотека spaCy

```
import spacy
nlp = spacy.load("en_core_web_lg")
doc = nlp("Did you enjoy those funny parties?")
print("п.номер слово   лемма  ч.речи  тип зав. гл.слово")
print("-------+-------+-------+-------+-------+-------------")
for tt in doc:
    print(tt.i, tt.text, tt.lemma_, tt.pos_, tt.dep_,
          tt.head.i, tt.head.text, sep="\t")
```

```
п.номер слово   лемма  ч.речи  тип зав. гл.слово
-------+-------+-------+-------+-------+-------------
0       Did     do      AUX     aux     2       enjoy
1       you     you     PRON    nsubj   2       enjoy
2       enjoy   enjoy   VERB    ROOT    2       enjoy
3       those   those   DET     det     5       parties
4       funny   funny   ADJ     amod    5       parties
5       parties party   NOUN    dobj    2       enjoy
6       ?       ?       PUNCT   punct   2       enjoy
```

Части речи I

Некоторые значения атрибута `pos_`

ADJ	прилагательное
ADP	предлог или послелог
ADV	наречие
CCONJ	сочинительный союз
DET	детерминатив (такой, мой, тот, весь...)
NOUN	существительное
PART	частица
PRON	местоимение
PROPN	имя собственное
PUNCT	знак пунктуации
SCONJ	подчинительный союз
VERB	глагол

Части речи II

AUX	вспомогательный глагол
INTJ	междометие
NUM	числительное
SPACE	пробел
X	другое

Полный список:

https://github.com/explosion/spaCy/blob/master/spacy/glossary.py

Грамматика зависимостей

Грамматика зависимостей описывает структуру предложения в терминах отношений «главное слово» — «зависимое слово» («хозяин» — «слуга»).

Некоторые значения атрибута `dep_`

ROOT	главный член предложения
advmod	обстоятельство
amod	определение, выраженное прилагательным
cc	сочинительный союз
det	детерминатив
nmod	определение, выраженное существительным
nsubj	именное подлежащее
obj	прямое дополнение
obl	косвенное дополнение
parataxis	сочинительная связь без союза

Часто встречающиеся слова I

```
from collections import Counter
import spacy

nlp = spacy.load("en_core_web_lg", exclude=["parser", "ner"])
print(nlp.pipe_names)
with open("in.txt", encoding="utf-8") as ff:
    text = ff.read()
doc = nlp(text)
word_counter = Counter(tt.text.lower() for tt in doc if tt.is_alpha)
for (word, count) in word_counter.most_common(10):
    print(count, word)
```

Часто встречающиеся слова II

```
['tok2vec', 'tagger', 'attribute_ruler', 'lemmatizer']
3364 the
2657 i
2305 to
2258 and
2103 a
1959 of
1775 he
1542 was
1270 in
1231 that
```

Арифметика семантики

Весёлые примеры арифметики семантики английских слов (с word2vec) можно найти здесь.
`http://graceavery.com/word2vec-fish-music-bass/`
Например, `bicycle - road + ocean = surfboard`.
Это означает, что в каком-то смысле слово «road» относится к слову «bicycle» так же, как слово «ocean» к слову «surfboard».

Следующая программа делает то же с русскими словами.
(Наилучшее соответствие ищется среди слов, встречающихся в файле vocab.txt.)

Арифметика семантики

```
import re
import spacy
vocab = {}
nlp = spacy.load("ru_core_news_lg", exclude=["parser", "ner"])
with open("vocab.txt", "r", encoding="utf-8") as ff:
    for doc in nlp.pipe(line.strip() for line in ff):
        for tt in doc:
            if tt.vector_norm > 0:
                vocab[tt.text.lower()] = tt.vector / tt.vector_norm
while True:
    line = input("a-b+c:")
    if line == "": break
    words = ["+"] + [w.strip() for w in re.split(r"([-+])", line)]
    forbidden, result = words.copy(), [0] * nlp.vocab.vectors_length
    while words:
        v1 = nlp(words[1]).vector
        result += -v1 if (words[0] == "-") else v1
        words[0:2] = []
    bw, bd = "", -1 - sum(x * x for x in result)
    for w, v in vocab.items():
        d = sum(x * y for (x, y) in zip(v, result))
        if d > bd and w not in forbidden: bw, bd = w, d
    print(bw)
```

Литература

Васильев Ю. Обработка естественного языка. Python и spaCy на практике. — Спб.: Питер, 2021. — 256 с.

Гольдберг Й. Нейросетевые методы в обработке естественного языка. — Москва: ДМК Пресс, 2019. — 282 с.

Грас Дж. Data Science. Наука о данных с нуля. — СПб.: БХВ-Петербург, 2021. — 336 с.

Классификация текстов на основе частей речи и грамматических связей

Около 400 литературных произведений на русском языке, находящихся в общественном достоянии (более 80 мегабайт) можно скачать из репозитория
`https://github.com/d0rj/RusLit`
Попробуем научить простейшую нейронную сеть из одного нейрона различать прозу Ф. М. Достоевского и М. Горького. Иными словами, используем линейный классификатор.
Разделим файлы из папки prose/Dostoevsky на тренировочную часть (train/Dostoevsky) и тестовую часть (test/Dostoevsky).
Также разделим файлы из папки prose/Gorky на тренировочную часть (train/Gorky) и тестовую часть (test/Gorky).

Предобработка данных

Следующая программа соберёт из папки train все абзацы длины хотя бы 100 символов в один файл, назовём его epar.trn.
Так выглядит файл epar.trn.

```
Dostoevsky,Ёлка и свадьба.txt,    7,"На днях я видел свадьбу
Dostoevsky,Ёлка и свадьба.txt,   11,"Кроме этой фигуры, таки
⋮
Gorky,Ярмарка в Голтве.txt,  148,"- Пойте славу бога нашего-
Gorky,Ярмарка в Голтве.txt,  150,"Кое-где уже вспыхнули ого
```

Та же программа соберёт из папки test все абзацы длины хотя бы 100 символов в один файл, назовём его epar.tst.

Предобработка данных

```
import os
import re
import sys

with open("0epar.txt", "w", encoding="utf-8") as outfile:
  for d in os.listdir('.'):
    if os.path.isdir(d):
      for f in os.listdir(d):
        p = os.path.join(d, f)
        if os.path.isfile(p) and p.endswith(".txt"):
          with open(p, encoding="utf-8") as ff:
            for (i, line) in enumerate(ff, 1):
              if len(line) > 99:
                line = line.strip().replace('"', "'")
                print(f'{d},{f},{i:5d},"{line}"', file=outfile)
```

Предобработка данных

Следующая программа укажет для каждого токена часть речи и тип грамматическ связи в грамматике зависимостей. Используем только 12 частей речи и 10 типов зависимостей, остальные заменим на значение по умолчанию (0).

```
spacy_posdep.py epar.trn posdep.trn ""
spacy_posdep.py epar.tst posdep.tst ""
```

Так выглядит файл posdep.trn.

```
Dostoevsky,Ёлка и свадьба.txt,    7,"[(2, 0), (6, 9), (8, 7), (12, 1
Dostoevsky,Ёлка и свадьба.txt,   11,"[(2, 0), (5, 5), (6, 9), (10, 0
    ⋮
Gorky,Ярмарка в Голтве.txt,  148,"[(10, 0), (12, 1), (6, 8), (6, 6),
Gorky,Ярмарка в Голтве.txt,  150,"[(8, 7), (8, 7), (3, 2), (3, 2), (
```

Предобработка данных

```
# spacy_posdep.py
import re
import sys
import spacy

nlp = spacy.load("ru_core_news_lg", exclude=["ner"])
all_pos = "X,ADJ,ADP,ADV,CCONJ,DET,NOUN,PART,PRON,PROPN,PUNCT,SCONJ,VERB".split(",")
all_dep = "X,ROOT,advmod,amod,cc,det,nmod,nsubj,obj,obl,parataxis".split(",")
with open(sys.argv[1], "r", encoding="utf-8") as infile, \
        open(sys.argv[2], "w", encoding="utf-8") as outfile:
    for line in infile:
        m = re.search(r'^(.*?),(.*?),(.*?),"(.*)"\n', line)
        if m and m[3].endswith(sys.argv[3]):
            vec = []
            for token in nlp(m[4]):
                pos = all_pos.index(token.pos_) if token.pos_ in all_pos else 0
                dep = all_dep.index(token.dep_) if token.dep_ in all_dep else 0
                vec.append((pos, dep))
            print(f'{m[1]},{m[2]},{m[3]},"{vec}"', file=outfile)
```

Линейная регрессия с scikit-learn

Следующая программа читает результат работы предыдущей программы, обучает линейный классификатор на одних произведениях и проверяет его на других произведениях. Для каждого абзаца считаются доли различных частей речи и различных типов зависимостей. В результате каждый абзац представляется вектором из 24 чисел между 0 и 1. Для отладки программа выводит векторы первых 3 абзацов, правильные ответы для первых 3 абзацов (0 означает, что абзац взят из произведения Ф. М. Достоевского, а 1 — М. Горького) и названия файлов, из которых взяты первые 3 абзаца. Далее выводятся коэффициенты обученного линейного классификатора и статистика его предсказаний (например, (0, 1) означает, что линейный классификатор предсказал ответ 0, а правильный ответ — 1). В последних шести строках программа для каждого тестового произведения выводит предсказание, вычисляемое, как среднее предсказаний для каждого абзаца этого произведения, а также указывает настоящего автора (0 или 1) и имя файла. Ошибочные предсказния авторства текста помечены восклицательным знаком.

```
sklearn_author.py posdep.trn posdep.tst Dostoevsky Gorky
```

Линейная регрессия с scikit-learn I

```
# sklearn_author.py
from collections import Counter, defaultdict
import re
import sys
import numpy as np
from sklearn.linear_model import LogisticRegression
all_authors = sys.argv[3:5]
def prepare_data(filename):
    authors, titles, vectors = [], [], []
    with open(filename, "r", encoding="utf-8") as ff:
        for line in ff:
            m = re.search(r'^(.*?),(.*?),(.*?),"(.*)"\n', line)
            if m and m[1] in all_authors:
                tokens = eval(m[4])
                vector = [0] * (24)
                for i in range(len(tokens)):
                    vector[tokens[i][0]] += 1
                    vector[tokens[i][1] + 13] += 1
                vectors.append([v / len(tokens) for v in vector])
                authors.append(all_authors.index(m[1]))
                titles.append(m[2])
    return (np.array(vectors), np.array(authors), titles)
```

Линейная регрессия с scikit-learn II

```
x, y, titles = prepare_data(sys.argv[1])
print(x[:3], y[:3], titles[:3], sep="\n")
cls = LogisticRegression()
cls.fit(x, y)
print(cls.coef_)

x, y, titles = prepare_data(sys.argv[2])
y_predict = cls.predict(x)
stat = Counter((y_predict[i], y[i]) for i in range(len(y)))
for key in sorted(stat):
    print(f"{key} occurred in {stat[key]} fragments")
dd = defaultdict(list)
for i in range(len(y)):
    dd[(y[i], titles[i])].append(float(y_predict[i]))
for pair in dd:
    p = sum(dd[pair]) / len(dd[pair])
    print(" " if round(p) == pair[0] else "!", f"{p:.3f} {pair}")
```

Линейная регрессия с scikit-learn III

```
[[0.04207921 0.07425743 0.08910891 0.09158416 0.04207921 0.04207921
  0.16089109 0.03465347 0.09405941 0.         0.17821782 0.03217822
  0.11881188 0.49009901 0.04455446 0.11386139 0.05940594 0.03960396
  0.03712871 0.01980198 0.07920792 0.03217822 0.07178218 0.01237624]
 [0.03954802 0.06214689 0.10169492 0.06214689 0.03389831 0.05649718
  0.19774011 0.01694915 0.08474576 0.02824859 0.16949153 0.02259887
  0.12429379 0.49152542 0.04519774 0.06779661 0.04519774 0.03389831
  0.05084746 0.05084746 0.07344633 0.04519774 0.09039548 0.00564972]
 [0.03469388 0.08367347 0.08571429 0.07346939 0.03877551 0.04693878
  0.2        0.0244898  0.08571429 0.00612245 0.16326531 0.01428571
  0.14285714 0.46938776 0.04489796 0.0877551  0.05510204 0.03877551
  0.04489796 0.05918367 0.0755102  0.04285714 0.07755102 0.00408163]]
[0 0 0]
['Ёлка и свадьба.txt', 'Ёлка и свадьба.txt', 'Ёлка и свадьба.txt']
[[-14.20507291   8.79096982  -0.17188967   0.89079592   1.60570136
   -4.09082856  11.22249132  -9.01887483   2.79117572  -6.93464443
   10.38139506  -5.99869576   4.73288828   0.08098048  -0.66341672
   -1.20191739  -2.43955993   1.38436098  -4.94559728   6.25262632
    3.32087339   3.24120973   3.63464113  -8.66878939]]
```

Линейная регрессия с scikit-learn IV

```
(0, 0) occurred in 2346 fragments
(0, 1) occurred in 1105 fragments
(1, 0) occurred in 669 fragments
(1, 1) occurred in 2536 fragments
  0.068 (0, 'Белые ночи.txt')
  0.075 (0, 'Вечный муж.txt')
  0.231 (0, 'Господин Прохарчин.txt')
  0.310 (0, 'Дневник писателя.txt')
  0.153 (0, 'Записки из подполья.txt')
  0.232 (0, 'Маленький герой.txt')
  0.306 (0, 'Неточка Незванова.txt')
  0.250 (0, 'Роман в девяти письмах.txt')
  0.278 (0, 'Честный вор.txt')
  0.970 (1, 'Город жёлтого дьявола.txt')
  0.569 (1, 'Дело с застёжками.txt')
  0.593 (1, 'Емельян Пиляй.txt')
  0.583 (1, 'Коновалов.txt')
  0.745 (1, 'Мать.txt')
! 0.427 (1, 'Мещане.txt')
  0.780 (1, 'Несвоевременные мысли.txt')
```

Линейная регрессия с scikit-learn V

```
0.638 (1, 'Скуки ради.txt')
0.603 (1, 'Старуха Изергиль.txt')
0.872 (1, 'Человек.txt')
```