Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по заданию №4
«Частеречная разметка»

по дисциплине «Компьютерная лингвистика»

Автор: Лакиза Александр Николаевич

Факультет: ИКТ

Группа: К3242

Преподаватель: Чернышева Анастасия Владимировна

Санкт-Петербург 2021

Цель работы: Провести частеречную разметку в корпусе. Найти «нетипичные» документы, у которых доля той или иной части речи отклоняется более чем на 2 стандартных отклонения

Ссылка на исходный код и json файл с корпусом: https://github.com/alexanderlakiza/cs224
Вся работа была сделана в файлах task4.py и task4_stat.ipynb. JSON-файлы, которые были созданы в процессе работы **corpus_as_dict_of_norms_and_pos.json**. CSV-файлы, созданные в процессе работы **pos_ratios.csv**.

Ход Работы: Для начала посмотрим файл **task4.py**. Мне необходимо провести частеречную разметку. Для этого я сначала подгружаю файл corpus_as_dict_of_norms.json, где ключи = заголовки документов, а значения = списки с токенами в начальной форме слова.

```
with open("corpus_as_dict_of_norms.json") as f:
    normed_dict = json.load(f)
```

Далее я создаю новый словарь, где значениями также будут заголовки, а в значениях будут лежать списки с элементами следующего вида: (Токен, Часть речи токена)

```
# Создаём словарь с частью речи для каждого токена
normed_dict_with_pos = {}
for title in list(normed_dict.keys()):
    normed_dict_with_pos[title] = [(word, morph.parse(word)[0].tag.POS) for word in normed_dict[title] if morph.parse(word)[0].tag.POS is not None]
with open("corpus_as_dict_of_norms_and_pos.json", "w") as f:
    # Запишем корпус в виде словаря в json, чтобы потом использовать готовый словарь
    # вместо использования pymorphy2
    json.dump(normed_dict_with_pos, f, ensure_ascii=False)

 with open("corpus_as_dict_of_norms_and_pos.json") as f:
     corpus_tokens_pos = json.load(f)
```

Для частеречной разметки используем **.tag.POS** из pymorphy2.

Размечая части речи, pymorphy2 делит прилагательные на 3 вида: ADJF – полные прилагательыне, ADJS – краткие прилагательные, COMP – сравнительные прилагательные, а также делит все глаголы на два вида: INFN – глаголы в начальной форме, VERB - глаголы в личной форме.

Я решил объединить все прилагательные в один тэг ADJ, а все глаголы – в VERB

```
# Пишем все виды глаголов в одну категорию VERB
# Пишем все виды прилагательных в одну категорию ADJ
for title in list(corpus_tokens_pos.keys()):
    for word in list(corpus_tokens_pos[title]):
        if word[1] == 'ADJF':
```

```
            word[1] = 'ADJ'
        elif word[1] == 'ADJS':
            word[1] = 'ADJ'
        elif word[1] == 'COMP':
            word[1] = 'ADJ'
        elif word[1] == 'INFN':
            word[1] = 'VERB'
        else:
            pass
```

Затем создаю словарь, где ключи всё те же, а в значениях лежит список со значениями типа (Часть речи, её доля в этом документе)

```
# Создаём словарь, где для каждого документа есть
# доля каждой части речи в этом документе
corpus_pos_ratios = {}
for title in list(corpus_tokens_pos.keys()):
    value = corpus_tokens_pos[title]
    poses_of_doc = [token[1] for token in value]
    doc_poses_ratios = [[pos,
round(poses_of_doc.count(pos)/len(poses_of_doc), 3)]
                        for pos in set(poses_of_doc)]
    corpus_pos_ratios[title] = doc_poses_ratios
```

Потом я создаю отсортированный словарь, где в значениях появятся еще и нули у тех частей речи, которых нет в данном документе.

```
# Максимальный набор используемых частей речи
lengths_pos = list(map(len, list(corpus_pos_ratios.values())))
index_max_n_pos = lengths_pos.index(max(lengths_pos))
used_pos = list(corpus_pos_ratios.values())[index_max_n_pos]
used_pos = [i[0] for i in used_pos]  # 11 частей речи,
использующихся в корпусе


# Сортируем доли все частей речи в каждом доке в одном порядке
# согласно порядку в used_pos: List
    sorted_c_pos_ratios = {}
    for title in list(corpus_pos_ratios.keys()):
        subdict = {}
        for pos in used_pos:
            subdict[pos] = 0
```

```
        sorted_c_pos_ratios[title] = subdict

        for pos_ratio in corpus_pos_ratios[title]:
            subdict[pos_ratio[0]] = pos_ratio[1]
```

Далее я создаю из словаря датафрейм и пишу его в csv-файл, чтобы удобнее было работать с данными в двумерном массиве.

Переходим к **task4_stat.ipynb**. Сначала нахожу средние значения для долей частей речи во всех документах, в которых эта часть речи встречается

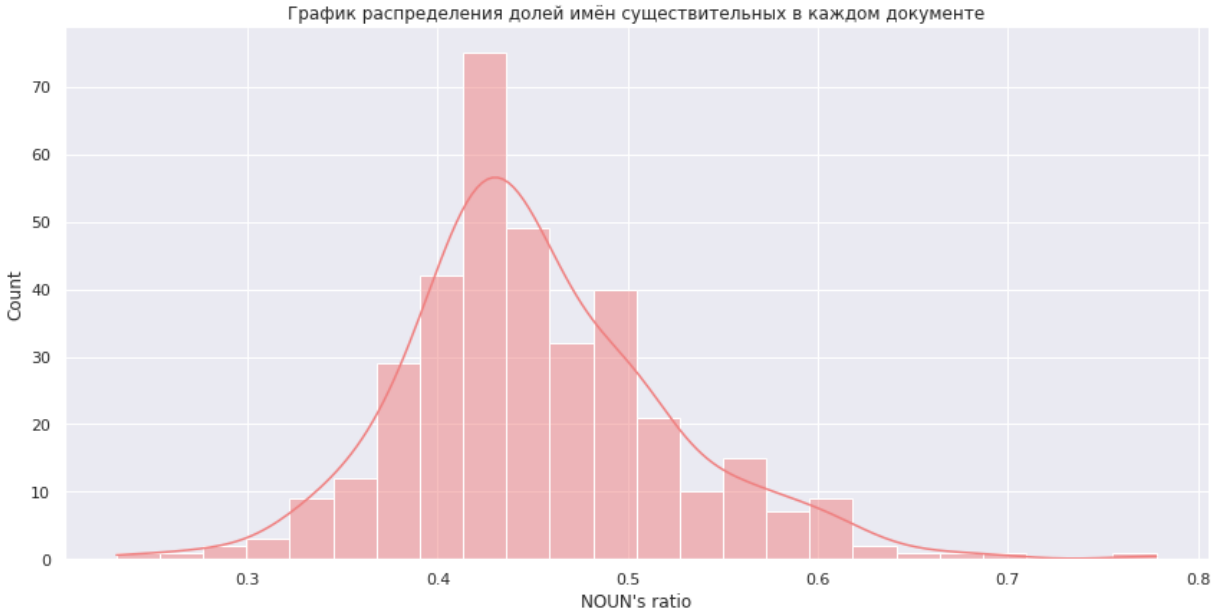
```
print('Средние значения долей частей речи в документах, в которых они появляются:\n')
for i in df.columns:
    print("{}'s mean is".format(i), round(df[df[i] != 0][i].mean(), 3))
```

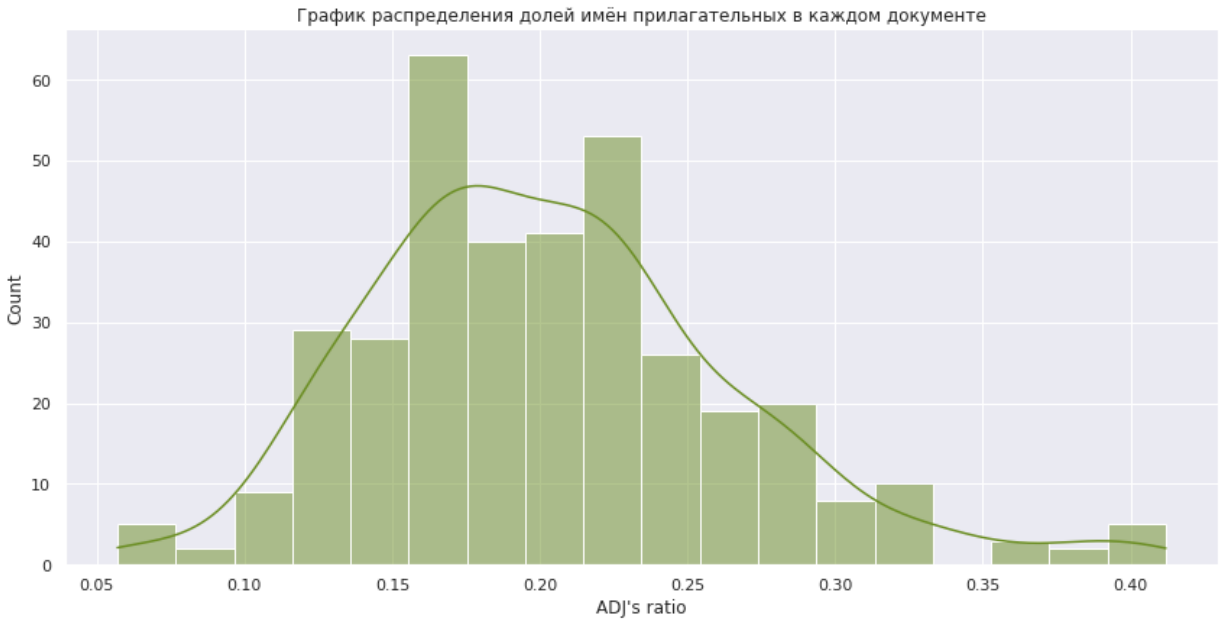
OUTPUT:

```
Средние значения долей частей речи в документах, в которых они появляются:

ADVB's mean is 0.035
NOUN's mean is 0.452
NUMR's mean is 0.016
ADJ's mean is 0.203
NPRO's mean is 0.019
PRED's mean is 0.01
VERB's mean is 0.104
CONJ's mean is 0.07
INTJ's mean is 0.01
PREP's mean is 0.132
PRCL's mean is 0.024
```

И далее я нарисовал графики распределения долей для каждой части речи. Примеры NOUN и ADJ:





Затем я решил посмотреть, как часто встречается та или иная часть речи в документе

```
pos_appearances = [len(df[df[i] != 0]) for i in df.columns]
d = {}
for i in range(len(df.columns)):
    d[df.columns[i]] = [pos_appearances[i],
round(pos_appearances[i]/len(df), 2)]
    df2 = pd.DataFrame.from_dict(d)
df2
```

	ADVB	NOUN	NUMR	ADJ	NPRO	PRED	VERB	CONJ	INTJ	PREP	PRCL
0	245.00	363.0	83.00	363.0	187.00	28.00	348.00	334.00	12.00	355.00	192.00
1	0.67	1.0	0.23	1.0	0.52	0.08	0.96	0.92	0.03	0.98	0.53

В нижней строке показана доля документов, в которых встречается эта часть речи, относительно числа всех документов

И наконец находим нетипичные документы.

```
print('Стандартные отклонения долей частей речи в документах, в
которых они появляются:\n')
for i in df.columns:
    print("{}'s mean is".format(i), round(df[df[i] !=
0][i].std(), 3))
```

OUTPUT:

```
Стандартные отклонения долей частей речи в документах, в которых они появл
яются:

ADVB's mean is 0.024
```

```
NOUN's mean is 0.071
NUMR's mean is 0.013
ADJ's mean is 0.062
NPRO's mean is 0.012
PRED's mean is 0.006
VERB's mean is 0.035
CONJ's mean is 0.031
INTJ's mean is 0.009
PREP's mean is 0.04
PRCL's mean is 0.016
```

```
stds_2 = [(i, 2*round(df[df[i] != 0][i].std(), 3)) for i in df.columns]
stds_2
```

OUTPUT:

```
[('ADVB', 0.048),
 ('NOUN', 0.142),
 ('NUMR', 0.026),
 ('ADJ', 0.124),
 ('NPRO', 0.024),
 ('PRED', 0.012),
 ('VERB', 0.07),
 ('CONJ', 0.062),
 ('INTJ', 0.018),
 ('PREP', 0.08),
 ('PRCL', 0.032)]
```

```
for i in stds_2:
    submean = df[df[i[0]] != 0][i[0]].mean()
    print('Нетипичные документы по {}:'.format(i[0]))
    res = list(df[(df[i[0]] < (submean - 2*i[1])) | (df[i[0]] > (submean + 2*i[1]))].index)
    if not res:
        print('Таких нет')
    else:
        print(res)
    print('\n')
```

OUTPUT:

```
Нетипичные документы по ADVB:
['Сгущённое молоко', 'Цистерна']


Нетипичные документы по NOUN:
['Пенсильвания']


Нетипичные документы по NUMR:
Таких нет
```

```
Нетипичные документы по ADJ:
Таких нет


Нетипичные документы по NPRO:
['Сахар']


Нетипичные документы по PRED:
Таких нет


Нетипичные документы по VERB:
['Алвин']


Нетипичные документы по CONJ:
['Фильм-катастрофа']


Нетипичные документы по INTJ:
Таких нет


Нетипичные документы по PREP:
Таких нет


Нетипичные документы по PRCL:
['Лимон', 'Саундтрек']
```

Вывод: проанализировав документы на нетипичность по признаку долей частей речи в документе, выяснил, что нетипичными документами являются документы совсем небольшого объёма. В таких документах так мало слов, что какая-то часть речи занимает «нетипично» большую долю.