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

Отчет

по заданию №8
«Графовое представление ключевых слов корпуса»

по дисциплине «Компьютерная лингвистика»

Автор: Лакиза Александр Николаевич

Факультет: ИКТ

Группа: К3242

Преподаватель: Чернышева Анастасия Владимировна

Санкт-Петербург 2021

Цель работы: построить граф 100 ключевых слов корпуса

Ход работы:

При построении графа учтите следующие условия:

● ключевым словом может быть только существительное

● ключевое слово не может быть стоп-словом

● между двумя словами есть связь, если они встретились в одном документе; вес ребра между вершинами равен количеству документов, в которых два рассматриваемых слова встретились вместе

● не включайте в граф ребра с весом меньше 2, общее количество вершин не должно превышать 100

● для визуализации графа можете воспользоваться Gephi или Flourish

● при визуализации настройте размер вершины (по степени) и толщину ребер (по весу)

В описании работы сказано провести токенизацию, лемматизацию и нормализацию текста. Я уже делал это, поэтому просто беру файл **corpus_as_dict_of_norms.json**, где лежит словарь, в котором ключи – заголовки, значения – список токенов этого документа в начальной форме.

Выбираем из всех токенов только существительные, при чем те, которые не являются стоп словами

```
words = [word for word in a if morph.parse(word)[0].tag.POS == 'NOUN']
stops = stopwords.words("english") + stopwords.words("russian") +
["это", "который", "наш", "мочь", "год", "такой", "знать", "мы", "свой",
"один", "другой", "хотеть", "человек", "всё", "все", "весь", "очень",
"думать", "нужно", "большой", "время", "использовать", "говорить",
"сказать", "иметь", "сделать", "первый", "каждый", "день", "её", "ваш",
"стать", "больший", "ваше", "день", "самый", "понять", "просто", "ещё",
"проблема", "также", "например", "м", "с"]
words = [word for word in words if word not in stops]  # Удаляем стоп
слова
```

С помощью Counter из модуля Collections создаём словарь с кол-вом повторений каждого из самых популярных слов. Записываем 100 самых популярных слов в top_words. Далее записываем все узлы нашего графа

```
edges = []
for word in top_words:
    for second_word in top_words:
        count = 0
        if word != second_word:
            for doc in list(corpus.values()):
                if word and second_word in doc:
                    count += 1
```

```
            edges.append((word, second_word, count))
edges = [edge for edge in edges if edge[2] >= 2]
```

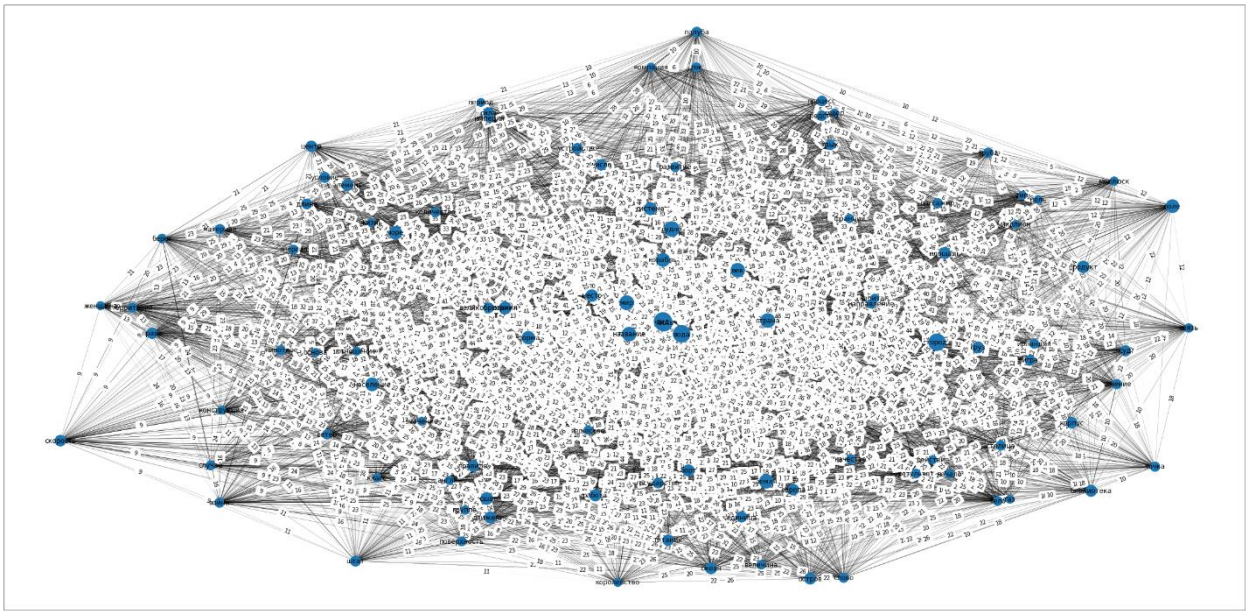
Строим наш граф

```
G = nx.Graph()
G.add_weighted_edges_from(edges)
pos = nx.spring_layout(G)
labels = {}
for i in edges:
    labels[(i[0],i[1])] = i[2]
widths = list(map(lambda x: x*0.01, list(labels.values())))
plt.figure(figsize=(40,20))
nx.draw_networkx(G, pos,  node_size=nodes_sizes, with_labels=True,
width=widths)
nx.draw_networkx_edge_labels(G, pos=pos, edge_labels=labels)
plt.show()
```

Пояснение по построению графа:

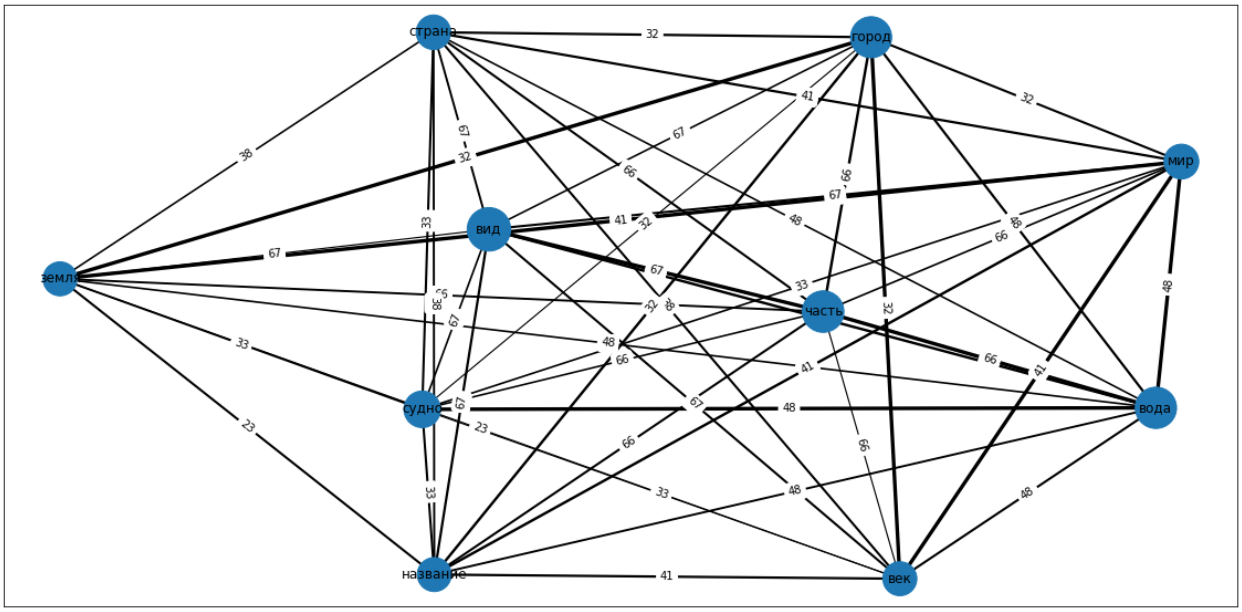
Я использую класс Graph(), куда добавляю ранее созданные узлы. Затем я создаю словарь labels, где хранятся веса узлов. В widths хранится толщина каждого узла на основе его веса. Далее используются две функции draw_networkx() и draw_networkx_edge_labels() для построения всех нужных нам деталей графа.

Получаем:



Картина получается жуткая, слишком много вершин (100) и узлов (9900).

Поэтому, чтобы получить хоть что-то внятное, я решил построить граф для 10 ключевых слов:



Тут дела обстоят гораздо лучше, но и тут не сказать, что легко считывать информацию с графа. В любом случае, впервые попрактиковался в построении графов с помощью networkx.

Ссылка на исходный код и json файл с корпусом:
https://github.com/alexanderlakiza/cs224/tree/main/task8