Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИУ»

КАФЕДРА «ИУ7»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1 «ИССЛЕД ВАХ ПОЛУПРОВОДНИКОВЫХ ДИОДОВ НА МО ЛАБОРАТОРНОГО СТЕНДА В ПРОГРАММЕ MICR

по дисциплине ««Основы электроники»

Тема Близость документов

Студент Батуев А.Г.

Группа ИУ7-36Б

Вариант 26

Преподаватели Оглоблин Д. И.

Москва, 2024

Содержание

ВВЕДЕНИЕ

Цель: исследовать файлы на похожесть

Задачи: создать векторы, которые будут репрезентовать соответствующие файлы в двух варинтах:

— выделением словоформ

— выделение начальных форм слов

1 Аналитическая часть

Близость файлов - это близость двух векторов, представляющих файл. Другими словами наша задача состоит в определении насколько различны эти вектора, используя те или иные способы сравнения. Обычно сравнение происходит по шкале от 0 (абсолютно не похожи) до 1 (идентичны).

Различных метрик сравнения файлов достаточно много, из них были использованы метод Жаккарда и косинусная мера близости.

Основная сЛожность задачи заключается в построении этих самых векторов, которые должны максимально точно репрезентировать файл. В данной задаче используется, пожалуй, самый примитивный метод сравнения: по словам и их количеству. Этот способ я назвал примитивным, потому что он не отражает смысловой нагрузки, которая содержится в тексте. Некоторые слова могут содержать разные смыслы, в зависимости от контекста, в котором они употребляются. Например, слово "стекло"может иметь значения:

1) Вещества и материала
2) Глагола совершенного вида (образовано от слова стечь)

К тому же роль играют и знаки препинания, которые могут придавать различные эмоциональные аспекты тексту. В данном случае, все знаки препинания считаются разделителями слов (в том числе и знак ’-’ который не всегда выступает в роли разделителя, например: что-то, где-то), а потому не учитываются в построении вектора.

Существуют и различные способы учета слов в построении вектора. Мы можем учитывать слово полностью или использовать только его начальную форму, таким образом:

1) Мама, мамы, маме, мам (все различные слова)
2) Мама, мамы, маме, мам (н.ф. - мама, поэтому они все одинаковые)

Вывод

По каждому файлу строится два n-мерных вектора в формате: слово : количество повторений. Первый вектор учитывает все слово целиком, второй только начальную форму. Каждый знак препинания считается как разделитель и не учитывается при построении вектора. Затем каждый вектор сравнивается с другими двумя методами: Жаккарда и косинусной мерой близости. По результатам сравнения строится график, который наглядно демонстрирует результаты сравнения.

2 Конструкторская часть

2.1 Подготовка данных

Название программы: preproc

Читает текст из документов с различными форматами в папке "тексты". По данному тексте строит два словаря (вектора) в формате слово : кол-во повторений, где в первом словаре сохраняется слово целиком, а во втором его начальная форма, оба вектора сохраняются в подпапку с именами vectors и norm vectors соответственно для дальнейшего анализа.

2.2 Метрики

Название программы: metrics.py

Содержит используемые для анализа метрики: метод Жаккарда и косинусной мерой близости. Сходство Жаккара — это простая, но иногда мощная метрика сходства. Даны две последовательности A и B: находим число общих элементов в них и делим найденное число на количество элементов обеих последовательностей.

Математическая формула: $Jac = \frac{len(A \cup B)}{len(A \cap B)}$

Косинусоидальное сходство вычисляет сходство двух векторов как косинус угла между двумя векторами. Это определяет, направлены ли два вектора примерно в одном направлении. Таким образом, если угол между векторами равен 0 градусам, то косинусоидальное сходство равно 1.

Математическая формула: $\cos(A, B) = \frac{A * B}{\|A\| \times \|B\|,}$

2.3 Получение результатов

Название программы: main.py

Использует полученные векторы и методы их анализа для получения матрицы схожести, которая, в свою очередь, используется для построения 4 графиков, отражающие результаты исследования. Графики сохраняются в папку "results"рядом с папкой "тексты".

Вывод

Подобная структура файлов, позволяет удобно ориентироваться между функциями каждой отдельной программы. Каждый файл выполняется свою важную функцию в общей программе.

3 Технологическая часть

3.1 Средства написания программы

Среда разработки: PyCharm 2023 community edition

Язык разработки: python 3.10

Используемые библиотеки:

— os - для доступа к файлам системы и создания папок/файлов URL: https://docs-python.ru/standart-library/modul-os-python/

— numpy - для удобной и быстрой работы с массивами, сохранения промежуточных данных и связи их с другими библиотеками URL: https://numpy.org/doc/

— re - для выделения слов URL: https://docs-python.ru/standart-library/modul-re-python/

— collections.defaultdict - для создания начальных словарей в формате (строка : число) URL: https://docs-python.ru/standart-library/modul-collections-python/klass-defaultdict-modulja-collections/

— bs4.BeautifulSoup - для чтения html файлов URL: https://www.crummy.com/software/Beautifu

— docx.Document - для чтения docx файлов URL: https://docs-python.ru/packages/modul-python-docx-python/klass-document/

— odf.opendocument.load и odf.text.P - для чтения odt файлов URL: https://pypi.org/project/odfpy

— pymorphy3 - для выделения начальных форм слова URL: https://pypi.org/project/pymorphy3/

— matplotlib.pyplot - для отображения графиков URL: https://matplotlib.org/stable/index.html

— seaborn - для построения графиков URL: https://seaborn.pydata.org/

— sklearn.metrics.pairwise.cosine similarity - косинусная мера близости URL: https://scikit-learn.org/stable/

3.2 Подготовка к анализу

Основные функции программы preproc.py, которая подготавливает данные для их дальнейшего анализа.

3.2.1 Построение векторов

Функция создания вектора

Листинг 3.1 — Построение векторов (обычного и нормализированного для одного документа)

```
    def build_vectors(text):
        morph = pymorphy3.MorphAnalyzer()
        words = list(word.lower() for word in re.split(r"[ -.,!?():;\n\t\r
            ]+", text) if word)
        word_count = defaultdict(int)
        word_count_norm = defaultdict(int)
```

```
    for word in words:
        word_count[word] += 1
        word_count_norm[morph.normal_forms(word)[0]] += 1
    return word_count, word_count_norm
```

3.2.2 Рекурсивное чтение файлов, создания векторов и запись в файл

Основная функция preproc.py, выполняющая все необходимые действия (чтение, создание, запись) для получения готовых к анализу векторов.

Листинг 3.2 — Получение всех векторов

```
def build_vectors_recursively(directory):
    for folder in os.listdir(directory):
        folder_path = os.path.join(directory, folder)
        vectors_dir_path = os.path.join(folder_path, "vectors")
        norm_vectors_dir_path = os.path.join(folder_path, "
            norm_vectors")

        if not os.path.exists(vectors_dir_path):
            os.mkdir(vectors_dir_path)

        if not os.path.exists(norm_vectors_dir_path):
            os.mkdir(norm_vectors_dir_path)

        for filename in os.listdir(folder_path):
            file_path = os.path.join(folder_path, filename)
            if file_path.endswith(EXT):
                text = ext_choice(file_path)
                if text is None:
                    print(f"None pointer return for file: {file_path}\
                        n")
                    continue
                vec, vec_norm = build_vectors(text)
                write_into_file(vec, vectors_dir_path, filename)
                write_into_file(vec_norm, norm_vectors_dir_path,
                    filename)
```

3.3 Метрики

Метод Жаккара и косинусная мера близости.

3.3.1 Метод Жаккара

Ранее уже рассмотренный метод Жаккара: См. 2.2

Листинг 3.3 — Метод Жаккара

```
def jacquard_similarity(v1: dict, v2: dict):
    a = set(v1.keys())
    b = set(v2.keys())
    shared = a.intersection(b)
    total = a.union(b)
    return len(shared) / len(total)
```

3.3.2 Косинусная мера близости

Ранее уже рассмотренная мера близости: См. 2.2

Листинг 3.4 — Косинусная мера близости

```
def cosine_metric(v1: dict, v2: dict):
    keys = sorted(set(v1.keys()).union(v2.keys()))
    vector1 = np.array([v1.get(key, 0) for key in keys])
    vector2 = np.array([v2.get(key, 0) for key in keys])
    return cosine_similarity([vector1], [vector2])[0][0]
```

3.4 Анализ

3.4.1 Матрица близости

Матрица близости - это матрица, которая в каждой ячейке (i, j) содержит сравнительный анализ двух векторов.

Листинг 3.5 — Получение матрицы близости

```
def get_corr_matrix(vectors):
    vecs_len = len(vectors)
    na = np.zeros((vecs_len, vecs_len))
    nb = np.zeros((vecs_len, vecs_len))
    for i in range(vecs_len):
        for j in range(vecs_len):
            na[i][j] = metrics.jacquard_similarity(vectors[i], vectors
                [j])
            nb[i][j] = metrics.cosine_metric(vectors[i], vectors[j])
    return na, nb
```

3.4.2 Карта сравнения

На основе матрицы, полученной из предыдущей функции, строится график в seaborn. Данный график является результатом всей работы.

Листинг 3.6 — Получение результирующих графиков

```
    def build_heatmaps_recursive(directory):
        vectors = list()
        norm_vectors = list()
        files = list()
        for folder in os.listdir(directory):
            folder_path = os.path.join(directory, folder)
            vectors_dir_path = os.path.join(folder_path, "vectors")
            norm_vectors_dir_path = os.path.join(folder_path, "
                norm_vectors")
            for filename in os.listdir(vectors_dir_path):
                files.append(filename)
                file_path = os.path.join(vectors_dir_path, filename)
                file_path_norm = os.path.join(norm_vectors_dir_path,
                    filename)
                vectors.append(get_vectors(file_path))
                norm_vectors.append(get_vectors(file_path_norm))

        matrix_jacquard, matrix_cosine = get_corr_matrix(vectors)
        plt.figure(figsize=(19.2, 10.8))
        sea.heatmap(matrix_jacquard, xticklabels=files, yticklabels=files,
            square=True, linewidths=.3, cmap="YlGnBu")
        plt.savefig(os.path.join(RES_DIR, "Heatmap Jac.png"), dpi=100)
        plt.clf()

        plt.figure(figsize=(19.2, 10.8))
        sea.heatmap(matrix_cosine, xticklabels=files, yticklabels=files,
            square=True, linewidths=.3, cmap="YlGnBu")
        plt.savefig(os.path.join(RES_DIR, "Heatmap cos.png"), dpi=100)
        plt.clf()

        plt.figure(figsize=(19.2, 10.8))
        matrix_jacquard, matrix_cosine = get_corr_matrix(norm_vectors)
        sea.heatmap(matrix_jacquard, xticklabels=files, yticklabels=files,
            square=True, linewidths=.3, cmap="YlGnBu")
        plt.savefig(os.path.join(RES_DIR, "Heatmap Jac norm.png"), dpi
            =100)
```

```
        plt.clf()

        plt.figure(figsize=(19.2, 10.8))
        sea.heatmap(matrix_cosine, xticklabels=files, yticklabels=files,
            square=True, linewidths=.3, cmap="YlGnBu")
        plt.savefig(os.path.join(RES_DIR, "Heatmap cos norm.png"), dpi
            =100)
```

Вывод

В результате получаем 4 графика (на каждую метрику отводится нормализированный и не нормализированный вектор). Эти графики в явном виде отображают близость каждого файла.

4 Исследовательская часть

4.1 Оборудование

Характеритстики ноутбука:

— Процессор intel-core i5-12500H

— ОЗУ 16 Гб DDR4

— ОС Windows 11

Цель исследования: разобраться, как происходит сравнение тектов на похожесть. Изучить методы сравнения текстов.

Исследование происходило при подключенном кабеле питания, при выключенных сторонних приложениях.

4.2 Графики

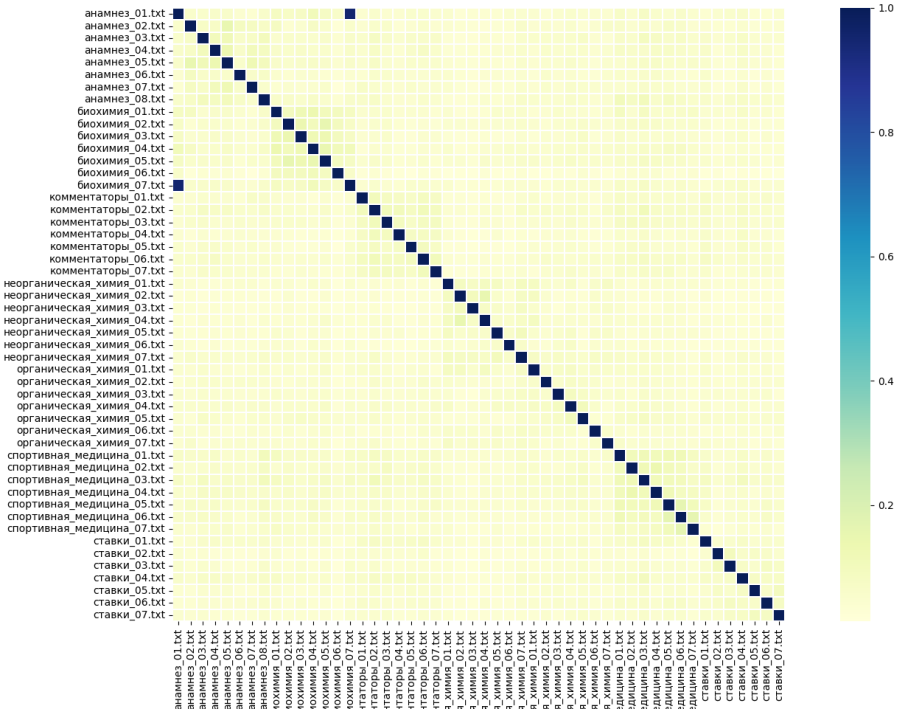

Рисунок 4.1 — Карта сравнения документов методом Жаккара

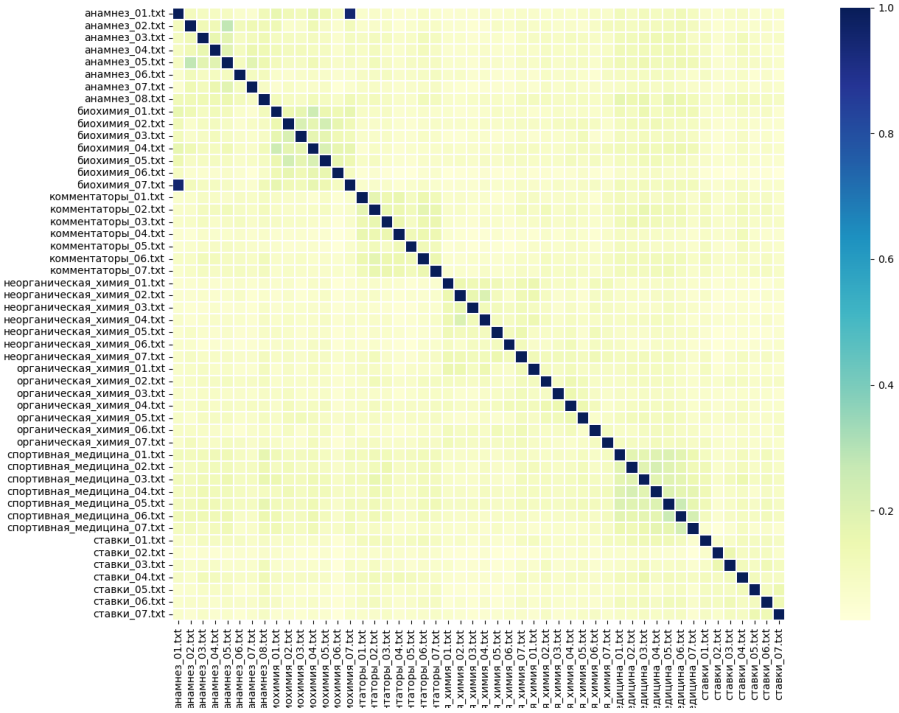

Рисунок 4.2 — Карта сравнения документов методом Жаккара, нормализированный

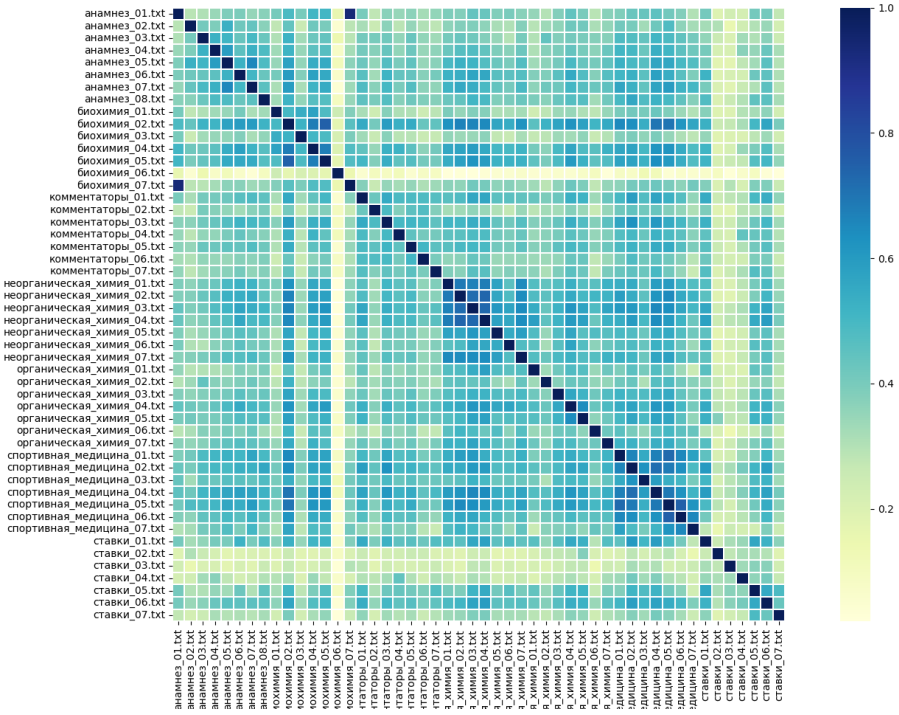

Рисунок 4.3 — Карта сравнения документов косинусной мерой близости

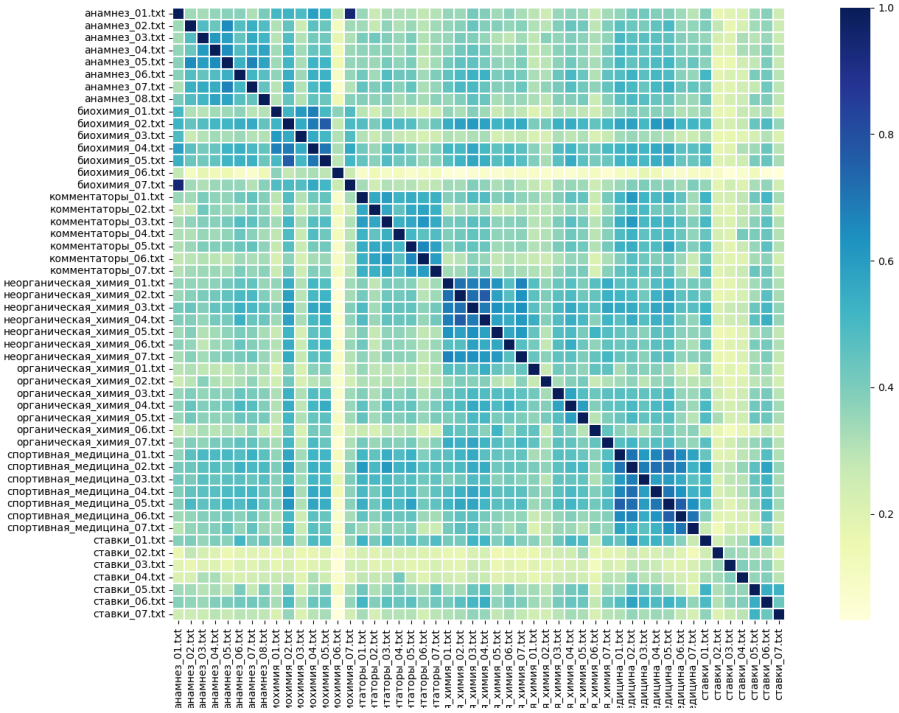

Рисунок 4.4 — Карта сравнения документов косинусной мерой близости, нормализированный

4.3 Вывод

В результате исследования наилучшим образом показал себя косинусный метод сравнения. На графике четко видна степень различия файлов. Метод Жаккара практически не показывает разницу между текстами.

Нормализированные вектора абсолютно не меняют рисунок графика, только усиливают корреляцию файлов.

ЗАКЛЮЧЕНИЕ

Удалось достигнуть выдвинутой цели: исследовать файлы на похожесть.

Была выполнена поставленная задача: успешно построены вектора, репрезентующие содержимое каждого файла в двух варинатах:

— с выделением словоформ

— с выделением начальных форм слов

В результате получили графики, на которых можно проследить близость файлов. Эти результаты подчеркивают важность выбора подходящей метрики для выбранной задачи. Так метод Жаккара практически не отражает схожесть документов, что идет в разрез с косинусной мерой.

Помимо прочего четкто прослеживается как нормализированный график отличается от обычного. Рисунок не изменяется, а вот оттенки цветов изменяются, преимущественно в большую сторону. То есть нормализированные вектора лишь подчеркивают уже вырисовывающуюся близость.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. N. Reimers, I. Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (2019), Proceedings of the 2019 Conference on Empirical Methods in 2019. URL: https://arxiv.org/abs/1908.10084 (Дата обращения 21.09.2024)

2. Seaborn: statistical data visualization URL: https://seaborn.pydata.org/ (Дата обращения 22.09.2024)

3. Руководство пользователя по pymorhpy2. URL: https://pymorphy2.readthedocs.io/en/stable/user/guide (Дата обращения 22.09.2024)

4. Metrics and scoring: quantifying the quality of predictions URL: https://scikit-learn.org/stable/modules/model_evaluation.html (Дата обращения 22.09.2024)