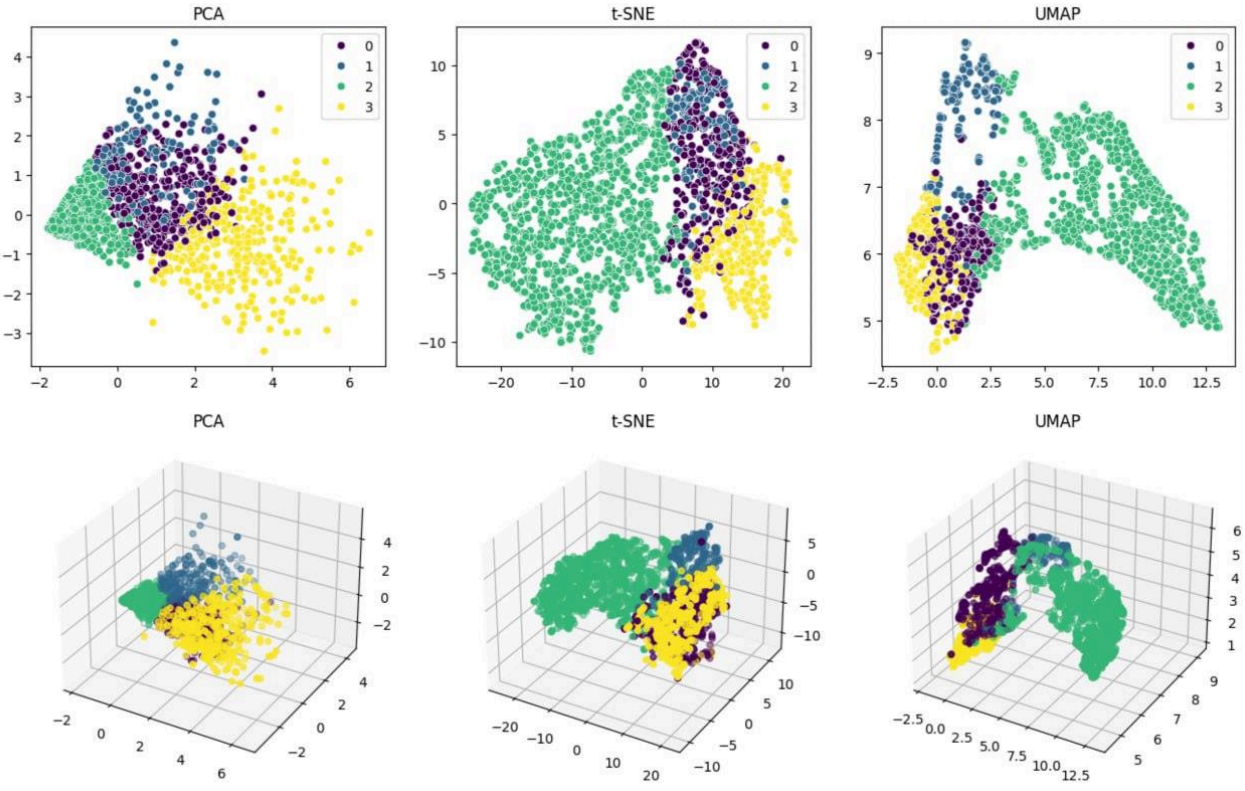
М.В. Хлопотов, А.В. Чернышева

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ОБРАБОТКЕ И ВИЗУАЛИЗАЦИИ ДАННЫХ НА PYTHON



Санкт-Петербург
2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

М.В. Хлопотов, А.В. Чернышева

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ОБРАБОТКЕ И ВИЗУАЛИЗАЦИИ ДАННЫХ НА PYTHON

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки
45.03.04 Интеллектуальные системы в гуманитарной сфере
в качестве Учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования бакалавриата

Санкт-Петербург
2024

Хлопотов М.В., Чернышева А.В., Методические указания к выполнению лабораторных работ по обработке и визуализации данных на Python– СПб: Университет ИТМО, 2022. – 64 с.

Рецензент(ы):
Мусаев Андрей Александрович, кандидат технических наук, доцент (квалификационная категория "доцент практики") факультета инфокоммуникационных технологий, Университета ИТМО.

Методические указания содержат набор лабораторных работ по некоторым разделам курсов «Основы обработки мультимедийных данных», «Обработка и визуализация данных на Python», «Язык Python для анализа данных». В пособии по каждой работе приводятся необходимые для выполнения теоретические сведения, краткое содержание, порядок выполнения и требования для сдачи лабораторного задания, а также перечень контрольных вопросов и дополнительных заданий для защиты. Методические указания предназначены для студентов бакалавриата, обучающихся по направлению 45.03.04 «Интеллектуальные системы в гуманитарной сфере» и осваивающих образовательную программу по профилям подготовки «Интеллектуальные системы в гуманитарной сфере» и «Языковые модели и искусственный интеллект».

ИТМО (Санкт-Петербург) — национальный исследовательский университет, научно-образовательная корпорация. Альма-матер победителей международных соревнований по программированию. Приоритетные направления: IT и искусственный интеллект, фотоника, робототехника, квантовые коммуникации, трансляционная медицина, Life Sciences, Art&Science, Science Communication.
Лидер федеральной программы «Приоритет-2030», в рамках которой реализовывается программа «Университет открытого кода». С 2022 ИТМО работает в рамках новой модели развития — научно-образовательной корпорации. В ее основе академическая свобода, поддержка начинаний студентов и сотрудников, распределенная система управления, приверженность открытому коду, бизнес-подходы к организации работы. Образование в университете основано на выборе индивидуальной траектории для каждого студента.
ИТМО пять лет подряд — в сотне лучших в области Automation & Control (кибернетика) Шанхайского рейтинга. По версии SuperJob занимает первое место в Петербурге и второе в России по уровню зарплат выпускников в сфере IT. Университет в топе международных рейтингов среди российских вузов. Входит в топ-5 российских университетов по качеству приема на бюджетные места. Рекордсмен по поступлению олимпиадников в Петербурге. С 2019 года ИТМО самостоятельно присуждает ученые степени кандидата и доктора наук.

Содержание

Введение

Для студентов направления подготовки 45.03.04 «Интеллектуальные системы в гуманитарной сфере» язык программирования Python является одним из основных инструментов выполнения профессиональных задач. Учебно-методическое пособие включает в себя семь лабораторных работ, охватывающих некоторые темы курсов «Основы обработки мультимедийных данных», «Обработка и визуализация данных на Python», «Язык Python для анализа данных». Студент найдёт здесь теоретические сведения, необходимые для квалифицированного выполнения каждой работы, содержание, порядок выполнения и требования для сдачи лабораторного задания, контрольные вопросы, а также примеры программных кодов. При выполнении лабораторных работ предполагается реализация алгоритмов на языке Python 3 для решения различных этапов обработки, анализа и визуализации данных.

Лабораторные работы посвящены следующим задачам в области обработки данных:

1. Манипуляции с табличными данными.
2. Базовые способы визуализации данных.
3. Обеспечение качества данных.
4. Рекомендательные системы.
5. Обработка изображений.
6. Обработка и классификация текстовых данных.
7. Построение и анализ графа ключевых слов.

Ознакомиться с заданиями для выполнения лабораторных работ можно в GitHub-репозитории по ссылке: https://github.com/anastasiiaCher/python-for-DPaV. Задания для выполнения лабораторных работ находятся в директории labs указанного репозитория. Каждое задание оформлено в формате Jupiter Notebook, содержащего теоретическую часть и комментарии, что и как должно выполняться в ходе выполнения работы.

Учебно-методическое пособие обобщает многолетний опыт авторов в ходе практической деятельности, а также преподавания курсов «Машинное обучение», «Анализ данных и информационный поиск» студентам факультета информационных технологий и программирования (ФИТиП), а также курсов «Программирование», «Методы визуализации данных», «Математические методы машинного обучения», «Основы обработки мультимедийных данных», «Язык Python для анализа данных», «Программные пакеты для анализа данных» студентам факультета инфокоммуникационных технологий (ФИКТ) Университета ИТМО. Материалы могут быть использованы при решении задач курсового и дипломного проекта, а также для выполнения студенческих научных

работ. Навыки, приобретаемые обучающимися в ходе выполнения лабораторных работ, способствуют формированию компетенций ОПК-4 «Способен к теоретическим и экспериментальным исследованиям в области профессиональной деятельности, включая постановку эксперимента, верификацию результатов, анализ, интерпретацию и презентацию данных» и ПК-1 «Способен управлять этапами жизненного цикла систем искусственного интеллекта в организации» (индикатор ПК-1.3 «Управление качеством данных для систем ИИ в организации»).

Лабораторная работа №1

МАНИПУЛЯЦИИ С ТАБЛИЧНЫМИ ДАННЫМИ

Цель: изучение основных способов обработки табличных данных и создания запросов к ним с помощью модуля pandas.

Краткое описание: в лабораторной работе представлено 8 заданий на сортировку, фильтрацию и группировку данных методами модуля pandas. Первые 4 задания являются обязательными. Из оставшихся 4 необходимо выполнить любые два на выбор обучающегося.

Данные: в папке Data/lab1 расположено несколько таблиц с обезличенными транзакционными банковскими данными. Все задания лабораторной работы необходимо выполнять по этим данным.

Описание данных

1. Таблица transactions.csv содержит историю транзакций клиентов банка за один год и три месяца.

   Формат данных:

   ```
   customer_id,tr_datetime,mcc_code,tr_type,amount,term_id
   111111,15 01:40:52,1111,1000,-5224,111111
   111112,15 15:18:32,3333,2000,-100,11122233
   ...
   ```

   Описание полей:

   - customer_id – идентификатор клиента;
   - tr_datetime – день и время совершения транзакции (дни нумеруются с начала данных);
   - mcc_code – mcc-код транзакции;
   - tr_type – тип транзакции;
   - amount – сумма транзакции в условных единицах со знаком:
     - \+ – начисление средств клиенту (приходная транзакция);

    - – списание средств (расходная транзакция);
  - term_id – идентификатор терминала.
2. Таблица gender_train.csv содержит информацию по полу для части клиентов, для которых он известен. Для остальных клиентов пол неизвестен.

   Формат данных:

   ```
   customer_id,gender
   111111,0
   111112,1
   ...
   ```

   Описание полей:
   - customer_id – идентификатор клиента;
   - gender – пол клиента.
3. Таблица tr_mcc_codes.csv содержит описание mcc-кодов транзакций.
   Формат данных:

   ```
   mcc_code;mcc_description
   1000;словесное описание mcc-кода 1000
   2000;словесное описание mcc-кода 2000
   ...
   ```

   Описание полей:
   - mcc_code – mcc-код транзакции;
   - mcc_description – описание mcc-кода транзакции.
4. Таблица tr_types.csv содержит описание типов транзакций.
   Формат данных:

   ```
   tr_type;tr_description
   1000;словесное описание типа транзакции 1000
   2000;словесное описание типа транзакции 2000
   ...
   ```

   Описание полей:
   - tr_type – тип транзакции;
   - tr_description – описание типа транзакции.

Порядок выполнения работы

Подготовка: загрузка данных. Допускается для таблицы transactions использование не всех записей.

Задания 1–4 нужно выполнить без использования функции merge и аналогичных по назначению.

Задание 1

1. Для столбца tr_type датафрейма transactions выберите произвольные 1000 строк с помощью метода sample().
2. В полученной на предыдущем этапе подвыборке найдите долю транзакций (столбец tr_description в датафрейме tr_types), в которой содержится подстрока “POS” или “ATM”.

Задание 2

1. Для столбца tr_type датафрейма transactions посчитайте частоту встречаемости всех типов транзакций tr_type в transactions.
2. Выведите датафрейм, содержащий топ-10 транзакций, отсортированных по убыванию частоты встречаемости. Датафрейм должен содержать столбцы tr_type, tr_description и столбец с частотой встречаемости.

Задание 3

1. В датафрейме transactions найдите клиента с максимальной суммой приходов на карту.
2. В датафрейме transactions найдите клиента с максимальной суммой расходов по карте.
3. Найдите модуль разницы для этих клиентов между суммой расходов и суммой приходов.

Задание 4

1. Найдите среднее арифметическое и медиану по столбцу amount для каждого типа транзакций из топ-10 в Задании 2.
2. Найдите среднее арифметическое и медиану по столбцу amount для каждого типа транзакций, совершенных клиентами из Задания 3.

Далее из заданий 5–8 нужно выполнить два любых на выбор.

Подготовка к выполнению заданий 5–8 включает объединение датафрейма transactions со всеми остальными таблицами (tr_mcc_codes, tr_types, gender_train). Причем объединение с таблицей gender_train выполняется с помощью left join, а с оставшимися датафреймами – через inner. После получения общей таблицы gender_train, tr_types и tr_mcc_codes можно удалить. В результате соединения датафреймов должно получиться 999584 строки.

```
transactions = pd.merge(transactions, gender_train, how='left')
transactions = pd.merge(transactions, tr_mcc_codes, how='inner')
transactions = pd.merge(transactions, tr_types, how='inner')
transactions.shape
```

Задание 5

1. Определите модуль разницы между средними тратами женщин и мужчин (трата – отрицательное значение в столбце amount).
2. Определите модуль разницы между средними поступлениями у мужчин и женщин.

Обратите внимание, что для вычисления модуля разности не требуется точных знаний о том, каким значением в таблице обозначены мужчины, а каким – женщины.

Задание 6

1. Для каждого типа транзакций рассчитайте максимальную сумму прихода на карту (из строго положительных сумм по столбцу amount) отдельно для мужчин и женщин (назовите ее max_income). Оставьте по 10 типов транзакций для мужчин и для женщин, наименьших среди всех типов транзакций по полученным значениям max_income.
2. Среди типов транзакций, найденных в пункте 1, выделите те, которые встречаются одновременно и у мужчин, и у женщин.

Задание 7

1. Найдите суммы затрат по каждой категории (mcc) для мужчин и для женщин.
2. Найдите топ-10 категорий с самыми большими относительными модулями разности в тратах для разных полов (в ответе должны присутствовать описания mcc-кодов).

Задание 8

1. Из поля tr_datetime выделите час tr_hour, в который произошла транзакция, как первые 2 цифры до ":".
2. Посчитайте количество транзакций со значением в столбце amount строго меньше 0 в ночное время для мужчин и женщин. Ночное время – это промежуток с 00:00 по 06:00 часов.

Порядок защиты и представление результатов выполнения лабораторной работы

Для защиты лабораторной работы студенту необходимо:

1. Предоставить результаты выполнения 6 заданий из 8 в виде ссылки на Google Colab.
2. Выполнить дополнительное задание в процессе сдачи работы. Дополнительное задание представляет собой запрос к данным лабораторной работы, по сложности аналогичный тем, что требовалось выполнить в рамках работы.

Примеры дополнительных заданий:

1. Найдите категорию транзакций, в которой совершается больше всего трат в ночное время (промежуток с 00:00 по 06:00). Выведите название категории и среднюю сумму трат в ней.
2. Найдите 3 категории транзакций, в которых средние траты мужчин и женщин различаются меньше всего.
3. Найдите терминал, через который проходит наибольшее число транзакций. Выведите идентификаторы 5 пользователей с наибольшей суммой отрицательных транзакций в этом терминале.

Преподаватель оставляет за собой право задать дополнительные вопросы, затрагивающие программную реализацию заданий.

Примечание

Рекомендуется выполнение заданий двумя способами – с помощью pandas и с помощью SQL с использованием модуля sqlite3.

Дополнительные материалы

1. Pandas. User Guide [Электронный ресурс]. URL: https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html (дата обращения: 14.03.2024).
2. МакКинни, У. Python и анализ данных [Текст] / У. МакКинни ; пер. с англ. Слинкиной А. А. – Москва : ДМК-Пресс, 2023. – 536 с.
3. Хейдт, М. Изучаем pandas [Текст] / М. Хейдт. – Москва : ДМК Пресс, 2019. – 700 с.

Лабораторная работа №2

БАЗОВЫЕ СПОСОБЫ ВИЗУАЛИЗАЦИИ ДАННЫХ

Цель работы: изучение основных способов визуализации табличных данных с помощью модулей seaborn, plotly и altair.

Краткое описание: в лабораторной работе представлено 10 заданий на построение графиков разных видов. В каждом задании указано, каким видом графика стоит воспользоваться для визуализации данных. По каждому заданию необходимо построить график дважды: первый раз с помощью модуля seaborn, второй – с помощью plotly или altair на выбор студента.

Данные: в папке Data/lab2 расположена таблица с информацией о песнях, исполненных на Евровидении с 2009 по 2023 год. Все задания лабораторной работы необходимо выполнять по этим данным. Данные в полном объеме можно найти по ссылке.

Описание данных

Формат данных:

```
year,country,artist_name,song_name,language,style,gender,main_singers,key,BPM,energy,danceability,happiness,loudness,acousticness,instrumentalness,liveness,speechiness,final_place
2023,Norway,Alessandra,Queen of Kings,English,Pop,Female,1,E Minor,110,36,64,23,10 dB,58,0,10,3,5
2023,Malta,The Busker,Dance (Our Own Party),English,Pop,Male,1,F Minor,103,78,70,82,6 dB,2,0,18,4,
…
```

Описание полей:

- year – год;
- country – страна участника;
- artist_name – исполнитель;
- song_name – название песни;
- language – язык, на котором исполняется песня;
- style – жанр;
- gender – пол участника;
- main_singers – количество вокалистов;
- key – регистр (высота тона);
- BPM – скорость композиции в целом (количество четвертых нот в минуту);
- energy – энергичность композиции;

- danceability – танцевальность (насколько трек подходит для танцев);
- happiness – жизнерадостность;
- loudness – громкость;
- acousticness – акустичность;
- instrumentalness – инструментальность;
- liveness – живость;
- speechiness – насколько много текста;
- final_place – место в фонале.

Порядок выполнения работы

Решением каждого задания должен являться график, требуемый по условию. По каждому заданию должен быть сделан вывод. Также нужно обязательно добавлять подпись графика и осей.

Задание 1

Проведите предобработку числовых значений в датасете:

1. В столбцах BPM, energy, danceability, happiness, acousticness, instrumentalness, liveness, speechiness замените отсутствующие значения и прочерки на 0.
2. В столбце loudness оставьте только число без единиц измерения.
3. Во всех перечисленных в пунктах 1 и 2 столбцах преобразуйте значения в тип данных float.

Задание 2

Покажите, какие характеристики музыки (BPM, energy, danceability, happiness, acousticness, instrumentalness, liveness, speechiness) коррелируют между собой. Для визуализации используйте диаграмму heatmap.

Задание 3

Покажите на диаграмме разброса взаимосвязь между жизнерадостностью (happiness) и энергичностью (energy) композиции. Добавьте на диаграмму вспомогательные элементы (отметку средних значений, выбросов) для упрощения интерпретации визуализации.

Задание 4

Покажите на горизонтальной столбиковой диаграмме распределение количества треков по жанрам. Отсортируйте диаграмму по убыванию

количества композиций в жанре. Выделите контрастным цветом жанр, в котором больше всего песен.

Задание 5

Покажите на круговой диаграмме страны, участники из которых побеждали чаще других, и в каких годах они выигрывали. Диаграмма должна быть одна.

Задание 6

На вертикальной столбчатой диаграмме с группировкой покажите количество мужчин и женщин среди участников в каждом году, представленном в датасете.

Задание 7

Постройте график плотности распределения данных в столбце danceability (танцевальность), сгруппированных по категориям жанрам.

Задание 8

Покажите на ящиковой диаграмме взаимосвязь позиции в финальном рейтинге со страной. Итоги каких стран похожи между собой?

Задание 9

Покажите на карте количество раз, когда каждая страна датасета проходила в финал.

Задание 10

Покажите на диаграмме вида treemap, сколько раз каждая страна входила в топ-3 победителей. В seaborn нет такого типа графика, с помощью этого модуля строить график не нужно.

Порядок защиты и представление результатов выполнения лабораторной работы

Для защиты лабораторной работы студенту необходимо:

1. Предоставить результаты выполнения 10 заданий в виде ссылки на Google Colab.
2. Выполнить дополнительное задание в процессе сдачи работы. Дополнительное задание представляет собой построение визуализации по

данным из лабораторной работы, по сложности аналогичной тем, что требовалось выполнить в рамках работы.

Примеры дополнительных заданий:

1. Покажите на ящиковой диаграмме распределение танцевальности для каждого жанра. Выполните с помощью seaborn.
2. На круговой диаграмме покажите исполнителей, которые участвовали больше одного раза и какие места они занимали. Выполните с помощью plotly.
3. Покажите на диаграмме вида treemap, сколько раз каждая страна проходила в финал. Выполните с помощью plotly.

Дополнительные материалы

1. seaborn: statistical data visualization [Электронный ресурс]. URL: https://seaborn.pydata.org/ (дата обращения: 01.04.2024).
2. Plotly Open Source Graphing Libraries [Электронный ресурс]. URL: https://plotly.com/graphing-libraries/ (дата обращения: 01.04.2024).
3. Vega-Altair: Declarative Visualization in Python [Электронный ресурс]. URL: https://altair-viz.github.io/index.html (дата обращения: 01.04.2024).

Лабораторная работа №3

ОБЕСПЕЧЕНИЕ КАЧЕСТВА ДАННЫХ

Цель работы: исследование и обработка данных для повышения их качества.

Краткое описание: в лабораторной работе приводится поэтапный план изучения и обработки намеренно не предобработанных данных. Необходимо выполнить все этапы плана и оценить результаты преобразований.

Данные: в папке Data/lab3 расположена таблица monster_com_job_sample.com с данными о вакансиях. Также эти данные с описанием и примерами обработки можно найти на kaggle.

Описание данных

Формат данных:

country,country_code,date_added,has_expired,job_board,job_description,job_title,job_type,location,organization,page_url,salary,sector,uniq_id
United States of America, US, "", No, jobs.monster.com, словесное описание вакансии, IT Support Technician Job in Madison, Full Time Employee, "Madison, WI 53702", "", http://jobview.monster.com/it-support-technician-job-madison-wi-us-167855963.aspx?mescoid=1500134001001&jobPosition=20, "", IT/Software Development, 11d599f229a80023d2f40e7c52cd941e
…

Описание полей:

- country – наименование страны;
- country_code – код страны;
- date_added – дата размещения вакансии;
- has_expired – актуальность вакансии;
- job_board – агрегатор вакансий, с которого собраны данные;
- job_description – описание вакансии;
- job_title – наименование должности;
- job_type – вид занятости;
- location – метоположение организации;
- organization – наименование организации;
- page_url – ссылка на страницу с вакансией;
- salary – заработная плата;
- sector – сфера деятельности;
- uniq_id – уникальный идентификатор записи о вакансии.

Порядок выполнения работы

Понимание данных

В датасете есть несколько столбцов, в каждой ячейке которых записано одно и то же значение. Есть столбцы, в которых формат записи значений не стандартизирован. А есть такие, где указана информация, не соответствующая названию столбца.

Первый этап изучения данных – понять, какие основные проблемы предстоит решить для каждого из столбцов. Для этого загрузите датасет и выведите случайные несколько строк из него. Постарайтесь понять, что из себя представляют данные в столбцах.

Формулировка задания: составьте таблицу, в которой будет содержаться название столбца, тип данных в нем и краткое описание проблем, которые вы заметили при первом взгляде на данные. Таблица должна иметь следующий вид:

Название столбца	Тип данных	Проблемы
country	…	…
country_code	…	…
date_added	…	…
has_expired	…	…
job_board	…	…
job_description	…	…
job_title	…	…
job_type	…	…
location	…	…
organization	…	…
page_url	…	…
salary	…	…
sector	…	…
uniq_id	…	…

Оценка пустых ячеек в датасете

Выясните, в каких столбцах, строках и в каком количестве отсутствуют данные:

1. С помощью модуля pandas составьте таблицу, где каждому названию столбца будет сопоставлено общее количество ячеек и количество пустых ячеек.
2. Подсчитайте количество строк датасета, в которых пустыми являются более 75% ячеек.
3. Визуализируйте число пропусков в датасете с помощью модуля, построив столбчатую диаграмму. Для удобства можно воспользоваться модулем missingno.

Проверьте, как изменится размер датасета, если принять решение удалить все строки, где есть пустые значения. Сделайте выводы.

Первичная оценка данных в столбцах

Изучите данные в столбцах датасета, ответив на следующие вопросы:

1. Есть ли столбцы, в которых встречается всего одно значение? Как их можно использовать в анализе?
2. Есть ли такие значения, которые встречаются более чем в одном столбце?
3. Есть ли столбцы, данные в которых не соответствуют названию столбца?

Дублирование данных

Для поиска дублирующихся строк выполните следующие действия:

1. В таблице есть столбец uniq_id, значения которого, как сказано в описании данных, являются уникальными идентификаторами записей. Проверьте, действительно ли в этом столбце все значения различны.
2. Проверьте, есть ли в таблице другие столбцы, значения которых в каждой строке различны.
3. Если игнорировать столбцы, где все значения различны, найдутся ли такие строки, для которых значения по всем остальным столбцам будут совпадать?

   Оцените результаты поиска дубликатов.

Детальная обработка данных в столбцах

Далее необходимо обработать значения в трех столбцах датасета: location, salary и job_type. Цель обработки – стандартизировать данные в каждом из столбцов так, чтобы их можно было анализировать.

Задание 1

В задании 1 необходимо провести обработку столбца location.

Задание 1.1

В столбце location представлена информация о местоположении организации, в которой открыта вакансия. Каждое значение столбца location содержит информацию о городе, коде штата и почтовом индексе.

Однако не в каждой строке есть все три характеристики местоположения, а также могут встречаться разные комбинации из них, записанные в виде строки в произвольном порядке и с разными разделителями.

Например, может быть указан только код штата, только название города или только индекс. Могут быть указаны любые две характеристики из трех.

Формулировка задания

1. Определите, какие форматы встречаются.
2. Напишите одно или несколько регулярных выражений для поиска выявленных форматов.
3. Визуализируйте количество найденных форматов на столбчатой или круговой диаграмме. Также покажите на ней количество строк, в которых формат определить не удалось. Какие форматы встречаются чаще всего?

Задание 1.2

В большей части строк удалось извлечь данные из ячейки столбца location, но остались и такие, где это не получилось. Вынесите в отдельный датасет записи, в которых не удалось определить формат записи информации о местоположении.

Над оставшимися строками проведите следующие преобразования:

1. Данные о местоположении разделите на 3 столбца: город, код штата, почтовый индекс.
2. Так как значения в столбце почтовый индекс являются пятизначными числами, преобразуйте их в целочисленный формат.

Рассмотрите отложенную часть выборки (ту, где не удалось данные из столбца location разделить на три поля) и сделайте выводы о возможности извлечь из данных город, код штата и почтовый индекс. Извлеките все, что возможно. Что невозможно – оставьте пустым.

Задание 2

В задании 2 необходимо провести обработку столбца salary.

Задание 2.1

В столбце salary, как и в location, данные не стандатизированы. В нем в основном можно встретить записи в следующих форматах:

- 10.00 - 20.00 $ /hour
- 10,000.00 - 20,000.00 $ /year
- $10.00 /hour
- $10,000.00+ /year
- Up to $20,000.00

Основные проблемы, которые стоит учесть при обработке форматов:

- В каждом из форматов может быть или не быть знака доллара.
- Может отсутствовать информация о периоде, за который указывается зарплата (/hour или /year в примерах).

- Зарплата может быть указана за месяц (/month) и за неделю (/week).
- В начале, конце и середине некоторых строк есть лишние пробелы, которые не стоит учитывать.

Формулировка задания:

1. Определите весь перечень встречающихся форматов и напишите одно или несколько регулярных выражений для их поиска.
2. Покажите на столбчатой диаграмме количество записей, содержащих каждый из форматов. Сделайте выводы.

Задание 2.2

1. Разделите столбец salary зарплаты три: минимальная граница зарплаты, максимальная граница зарплаты, период. Если значение суммы указано только одно, считайте, что это минимальная зарплата. Значение суммы в обоих столбцах преобразуйте в формат float.
2. Покажите на столбчатой диаграмме с группировкой среднюю и медианную минимальную зарплату в каждом периоде. Сделайте выводы.

Задание 2.3

Как было отмечено в задании 2.1, период, за который указывается зарплата, может отсутствовать. Восполнить этот пробел в данных нужно следующим образом:

1. Определить диапазон изменения (разброс) значений денежной суммы за каждый из встречающихся в данных периодов. Построить гистограмму для каждого периода.
2. Определить, к среднему значению за какой период ближе всего сумма, у которой период не указан. Указать для каждой суммы найденный период. В спорных случаях, где сумма одинаково близка к нескольким диапазонам, не нужно выбирать ничего.

Задание 3

В задании 3 необходимо провести обработку столбца job_type. Нужно выяснить, какие форматы значений есть в столбце job_type, сделать выводы. При необходимости – построить графики. Многие значения фактически одинаковые, но немного по-разному отформатированы. Их нужно привести к единому формату.

Столбец job_type состоит из двух частей: "полнота" занятости (Full Time, Part Time, Per Diem) и "тип" занятости (Employee, Intern). Кроме того, встречаются и неполные записи.

Формулировка задания:

1. Написать одно или несколько регулярных выражений для разделения столбца job_type на два: полнота занятости, тип занятости.
2. Если данные о полноте отсутствуют, внесите значение Full Time. Если отсутствуют данные о типе – внесите Employee.
3. Постройте столбчатую диаграмму со стековой группировкой для полноты и типа занятости. Сделайте выводы.

Порядок защиты и представление результатов выполнения лабораторной работы

Для защиты лабораторной работы студенту необходимо:

1. Предоставить результаты выполнения всех заданий в виде ссылки на Google Colab.
2. Выполнить дополнительное задание в процессе сдачи работы. Дополнительное задание представляет собой решение задачи на обработку данных лабораторной работы, меньшее по объему, но сопоставимое по сложности тем заданиям, которые требовалось выполнить в рамках работы.

Примеры дополнительных заданий:

1. В столбце sector часто представлены данные из столбца organization и наоборот. Устраните эту проблему.
2. В столбце job_title часто можно частично или полностью встретить указание местоположения организации. Сравните, совпадают ли в таких случаях данные с тем, что указано в столбцах город, код штата, почтовый индекс. Если в этих трех столбцах данные отсутствуют, заполните их данными из столбца job_title.
3. Практически во всех столбцах датасета вместо соответствующих названию данных иногда встречается текст из job_description. Найдите такие столбцы и записи с описанием вакансии в них. Покажите на столбчатой диаграмме, в каких столбцах и в каком количестве находится информация из столбца job_description.

Дополнительные материалы

1. DAMA-DMBOK: Свод знаний по управлению данными: Второе издание / DAMA International; [пер. с англ.]. – Москва: ООО "Олимп-Бизнес", 2021. – 828 с.
2. Макгрегор, С. Обработка данных на Python: Data Wrangling и Data Quality: [Текст] / С. Макгрегор. – Москва: БХВ, 2024. – 432 с.

Лабораторная работа №4

РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ

Цель работы: применение идей коллаборативной фильтрации и фильтрации, основанной на контенте, для формирования рекомендаций фильмов пользователям.

Краткое описание: в лабораторной работе необходимо выполнить 3 задания на исследование данных о фильмах с сайта MovieLens и применение двух подходов к построению рекомендаций фильмов – на основе предпочтений похожих пользователей и на основе предпочтений пользователя, для которого генерируется рекомендация.

Данные: набор данных ml-25m включает в себя описание 5-звездного рейтинга фильмов с сайта MovieLens – одного из старейших сервисов по рекомендации фильмов. Набор данных содержит 25 000 095 оценок и 1 093 360 применений тегов к 62 423 фильмам. Эти данные были созданы 162 541 пользователями в период с 9 января 1995 года по 21 ноября 2019 года – именно в этот день был сгенерирован датасет. Пользователи были выбраны случайным образом при условии, что пользователь оценил как минимум 20 фильмов. Демографическая информация в набор данных не включена. Каждый пользователь представлен только id. В папке Data/lab4 расположено несколько таблиц с характеристиками фильмов. Также эти данные доступны для скачивания по ссылке (как и другие наборы данных от GroupLens). Все задания лабораторной работы необходимо выполнять по этим данным.

Описание данных

1. Таблица tags.csv содержит все теги фильмов. Теги - это сгенерированные пользователями метаданные о фильмах. Каждый тег обычно представляет собой слово или короткую фразу. Значение, ценность и цель каждого тега определяется каждым пользователем.

Формат данных:

```
userId,movieId,tag,timestamp
3,260,classic,1439472355
3,260,sci-fi,1439472256
4,1732,dark,comedy,1573943598
…
```

Описание полей:

- userId – идентификатор пользователя;
- movieId – идентификатор фильма;
- tag – название тега;
- timestamp – количество секунд, прошедших с 1 января 1970.

```
# создание датафрейма тегов
tag=pd.read_csv('tags.csv')
tag.sample(3)
tag_counts = tag['tag'].value_counts()
tag_counts[:10].sort_values().plot(kind='barh', figsize=(8,4), colormap="Accent");
```

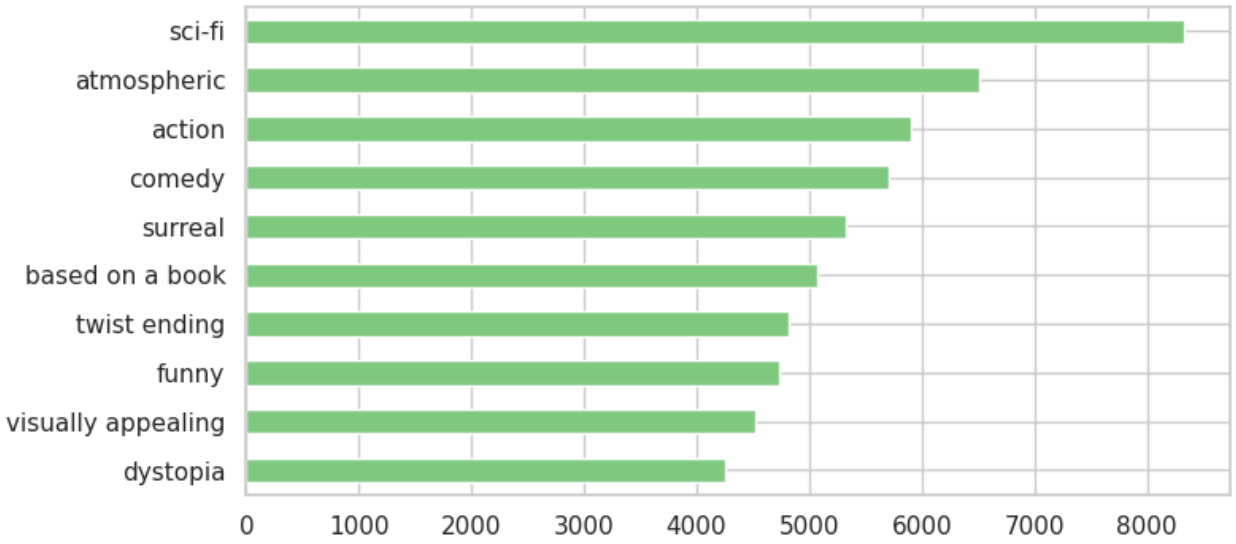


Рисунок 1 – Топ-10 наиболее частых тегов фильмов, полученный с помощью кода выше

2. В таблице ratings.csv представлены оценки пользователей за фильмы. Рейтинги представляют собой шкалу из 5 звезд с использованием половинки звезды (от 0.5 звёзд до 5.0 звезд).

Формат данных:

```
userId,movieId,rating,timestamp
1,296,5.0,1147880044
1,306,3.5,1147868817
1,307,5.0,1147868828
…
```

Описание полей:

- userId – идентификатор пользователя;
- movieId – идентификатор фильма;
- rating – оценка, которую пользователь поставил фильму;
- timestamp – количество секунд, прошедших с 1 января 1970.

Строки упорядочены по userId, а затем по movieId.

```
# загрузка данных о рейтингах
rating=pd.read_csv('ratings.csv')
rating.sample(3)

# гистограмма рейтингов
rating["rating"].plot(kind='hist', figsize=(8,4), colormap="Paired",
xticks=np.arange(0.5,5.5,0.5));
```

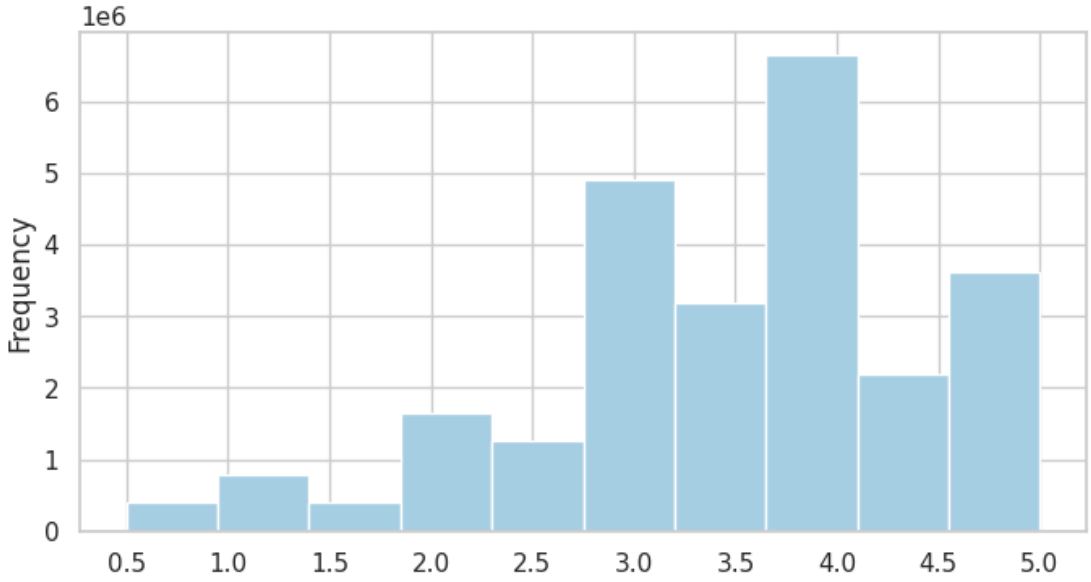


Рисунок 2 – Распределение оценок фильмов, полученное с помощью кода выше

3. Таблица movies.csv содержит информацию о фильмах. Каждая строка представляет информацию об одном фильме. Названия фильмов вводились вручную или были импортированы с сайта https://www.themoviedb.org/.

Формат данных:

```
movieId,title,genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
…
```

Описание полей:

- movieId – идентификатор фильма;
- title – название фильма и год выпуска в скобках после названия;
- genres – список жанров фильма, все жанры записываются в одном поле через разделитель “|”.

Допустимые жанры:

1. Action
2. Adventure
3. Animation
4. Children's
5. Comedy
6. Crime
7. Documentary
8. Drama
9. Fantasy
11. Horror
12. Musical
13. Mystery
14. Romance
15. Sci-Fi
16. Thriller
17. War
18. Western
19. (no genres listed)

4. Таблица genome-tags.csv содержит tag genome – структуру данных, которая описывает релевантность тегов по отношению к фильму. Заданы 1128 тэгов, и по каждому фильму указаны значения релевантности для каждого тега. Tag genome был рассчитан с помощью алгоритма на основе пользовательского контента, включая теги, рейтинги и текстовые описания. Таблица содержит описания тегов.

Формат данных:

```
tagId,tag
2,007 (series)
4,1920s
11,3d
22,adapted from:book
…
```

Описание полей:

- tagId – идентификатор тега, значения сгенерированы в тот момент, когда экспортировался весь набор данных, так что могут быть не

такими, как в других версиях датасетов от MovieLens (например, 1M и 20M).;

- tag – название тега.

5. Таблица genome-scores.csv содержит релевантность тегов по отношению к фильмам.

   Формат данных:

```
movieId,tagId,relevance
1,3,0.0625
1,5,0.14075
1,8,0.20375
…
```

   Описание полей:

   - movieId – идентификатор фильма;
   - tagId – идентификатор тега;
   - relevance – релевантность тега фильму.

```
# загрузка данных о фильмах и tag genome
movies = pd.read_csv("movies.csv")
genome_scores = pd.read_csv('genome-scores.csv')
genome_tag = pd.read_csv('genome-tags.csv')

# выведем 5 фильмов с самой высокой релевантностью тега "dragon"
genome_tag[genome_tag.tag == 'dragon']
top5_dragon_genom = genome_scores.query("tagId == 321").nlargest(5, "relevance")
movies[movies.movieId.isin(top5_dragon_genom.movieId)]
```

	movieId	title	genres
642	653	Dragonheart (1996)	Action|Adventure|Fantasy
4933	5039	Dragonslayer (1981)	Action|Adventure|Fantasy
14477	76093	How to Train Your Dragon (2010)	Adventure|Animation|Children|Fantasy|IMAX
20603	106489	Hobbit: The Desolation of Smaug, The (2013)	Adventure|Fantasy|IMAX
21757	112175	How to Train Your Dragon 2 (2014)	Action|Adventure|Animation

Рисунок 3 – Пять фильмов с самой высокой релевантностью тега "dragon"

6. Таблица links.csv содержит идентификаторы фильмов с других ресурсов. В каждой строке находятся ссылки на один фильм.

Формат данных:

```
movieId,imdbId,tmdbId
1,114709,862
2,113497,8844
3,113228,15602
…
```

Описание полей:

- movieId – идентификатор фильма на https://movielens.org. Например, фильм "Toy Story" имеет ссылку https://movielens.org/movies/1;
- imdbId – идентификатор фильма на http://www.imdb.com. Например, фильм Toy Story имеет ссылку http://www.imdb.com/title/tt0114709/;
- tmdbId – идентификатор фильма на https://www.themoviedb.org. Например, фильм Toy Story имеет ссылку https://www.themoviedb.org/movie/862.

Понимание данных

В таблице movies.csv представление жанров не очень хорошее: не очень понятно, что с ними можно делать. Очень часто для подобных данных используется следующий подход: под каждый жанр создается новый столбец, в соответствующем жанру столбце у фильма записывается 1, если в перечне был такой жанр, и 0 – если не было. Этот подход чем-то похож на one-hot encoding.

Вторая проблема с данными – наличие в столбце title года выпуска фильма. Лучше всего под год выпуска выделить отдельный столбец.

```
# объединим таблицы с описаниями фильмов и их рейтингами в одну
data = pd.merge(movies,rating)
# 5 случайных строк из объединенной таблица
data.sample(5)
```

	movieId	title	genres	userId	rating	timestamp
15442716	4448	Score, The (2001)	Action|Drama	150277	4.0	1121817385
15894264	4896	Harry Potter and the Sorcerer's Stone (a.k.a. ...	Adventure|Children|Fantasy	52444	4.0	1008613198
14638104	3988	How the Grinch Stole Christmas (a.k.a. The Gri...	Children|Comedy|Fantasy	117286	2.5	1564945592
4546095	736	Twister (1996)	Action|Adventure|Romance|Thriller	149751	4.0	967997079
17682996	6687	My Boss's Daughter (2003)	Comedy|Romance	91496	2.0	1446858882

Рисунок 4 – Пять случайных записей о фильмах, полученных с помощью кода выше

```
# в таблице number_rating будет храниться общее количество оценок фильму
number_rating = data.groupby('title')['rating'].count().rename(
"rated_by_users").reset_index()

# т.к. обработка всей таблицы с рейтингами фильмов от пользователей
# перегружает оперативную память, для примера возьмем случайные 10K строк из нее
data_train = data.sample(10000)

# составим сводную таблицу рейтингов, который каждый пользователь ставил каждому фильму
movie_pivot = data_train.pivot_table(index=["userId"],
                        columns=["title"],
                        values="rating")
movie_pivot.head().T
```

userId	4	61	73	83	131
title					
'burbs, The (1989)	NaN	NaN	NaN	NaN	NaN
(500) Days of Summer (2009)	NaN	NaN	NaN	NaN	NaN
*batteries not included (1987)	NaN	NaN	NaN	NaN	NaN
10 Things I Hate About You (1999)	NaN	NaN	NaN	NaN	NaN
10 to Midnight (1983)	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...
[REC] (2007)	NaN	NaN	NaN	NaN	NaN
eXistenZ (1999)	NaN	NaN	NaN	NaN	NaN
xXx (2002)	NaN	NaN	NaN	NaN	NaN
xXx: State of the Union (2005)	NaN	NaN	NaN	NaN	NaN
¡Three Amigos! (1986)	NaN	NaN	NaN	NaN	NaN

Рисунок 5 – Сводная таблица рейтингов по фильмам и пользователям

Матрица предпочтений, полученная с помощью кода выше, состоит практически из одних нулей. В такой матрице маловероятно найти хоть что-то. Например, поищем похожий фильм.

```
watched_movie = movie_pivot['10 Things I Hate About You (1999)']
similar_movies = movie_pivot.corrwith(watched_movie)
similar_movies = similar_movies.sort_values(ascending=False)
similar_movies.head()
```

Судя по результату на рисунке 6, фильм похож только сам на себя.

```
title
10 Things I Hate About You (1999)    1.0
'burbs, The (1989)                   NaN
(500) Days of Summer (2009)          NaN
*batteries not included (1987)       NaN
10 to Midnight (1983)                NaN
dtype: float64
```

Рисунок 6 – Результат поиска наиболее похожих фильмов на выбранный

Порядок выполнения работы

Задание 1

Выход из ситуации с неудачным поиском похожих фильмов такой: будем делать рекомендации не всем и сразу, а конкретному пользователю. Формировать movie_pivot будем только для этого конкретного пользователя.

1. Возьмем все оценки, которые поставил пользователь *U.*
2. По фильмам, которые он оценил, получим всех пользователей, которые ставили этим фильмам оценки.
3. Скорее всего, на этом этапе список получится большим (для случайного пользователя может получиться около 2 миллионов строк). Например, оставить только пользователей, у которых много общих фильмов.
4. Для каждого пользователя посчитаем похожесть (например, корреляцию).
5. Отсортируем пользователей по похожести.
6. Оставим только *k* самых похожих пользователей.

Пример решения

```
# выбираем случайного пользователя
all_users = rating['userId'].value_counts()
user = all_users.sample(1).index

# получаем фильмы, которые пользователь посмотрел
user_movies = data[data.userId == user[0]]
user_movies = user_movies.drop(['genres', 'timestamp', 'userId'], axis = 1)
```

```
# теперь найдем пользователей, которые смотрели эти фильмы
new_data = data[data.movieId.isin(user_movies.movieId)]
new_data = new_data.drop(['genres', 'timestamp'], axis = 1)
new_data.sample(10)
```

	movieId	title	userId	rating
17035858	5952	Lord of the Rings: The Two Towers, The (2002)	99032	5.0
13171361	3176	Talented Mr. Ripley, The (1999)	106957	2.0
17869138	6874	Kill Bill: Vol. 1 (2003)	159523	4.5
17440212	6377	Finding Nemo (2003)	95641	3.0
16008807	4973	Amelie (Fabuleux destin d'Amélie Poulain, Le) ...	106881	4.0
18344288	7361	Eternal Sunshine of the Spotless Mind (2004)	161707	5.0
15812345	4878	Donnie Darko (2001)	47940	2.5
12841059	3052	Dogma (1999)	13724	3.0
18175246	7153	Lord of the Rings: The Return of the King, The...	156458	4.0
9991037	2080	Lady and the Tramp (1955)	75344	2.0

Рисунок 7 – Пример вывода 10 строк о пользователях, которые смотрели те же фильмы, что и тот, для которого делается рекомендация

```
# по ID пользователя получим список его фильмов
users_group = new_data.groupby(['userId'])

# найдем фильмы трех пользователей, оценивших больше всего фильмов в выборке
sorted_users_group = sorted(users_group,  key=lambda x: len(x[1]), reverse=True)
# найдем ID пользователя, у которого больше всего похожих фильмов
similar_movies_user = sorted_users_group[1][0]
# скорее всего, в sorted_users_group[0][0] окажется сам пользователь
person2 = users_group.get_group(similar_movies_user).sort_values(by='movieId')

# получим список одинаковых фильмов для двух пользователей с оценками
temp = user_movies[user_movies['movieId'].isin(users_group.get_group(similar_movies_user)['movieId'])]
person1 = temp.sort_values(by='movieId')
```

Далее посчитаем коэффициент корреляции Пирсона для этих двух пользователей.

```
from scipy.stats import pearsonr
pearsonr(person1.rating, person2.rating)[0]  # 0.47140511653997375
```

С этого места нужно дописать самостоятельно. Нужно пройтись по всем пользователям и отсортировать их по похожести и оставить k самых похожих. Далее в коде приведены примеры расчетов для случайных пользователей. Их нужно заменить своими.

```
users = # PUT YOUR CODE HERE
pearsonCorDict = {}
for u in users:
  # PUT YOUR CODE HERE
    pearsonCorDict[u] = # PUT YOUR CODE HERE
pearsonDF = pd.DataFrame.from_dict(pearsonCorDict, orient='index')
pearsonDF.columns = ['similarityIndex']
pearsonDF['userId'] = pearsonDF.index
pearsonDF.index = range(len(pearsonDF))
topUsers=pearsonDF.sort_values(by='similarityIndex', ascending=False)[:7]
topUsers
```

Теперь у нас есть похожие пользователи. Получим оценки, которые они ставили за все фильмы.

```
topUsersRating = topUsers.merge(data, left_on='userId', right_on='userId', how='inner')
```

	similarityIndex	userId
4	0.997176	74212
6	0.965176	155627
12	0.905947	93755
0	0.842318	77311
15	0.775616	143732
18	0.756650	151243
16	0.753673	109758

Рисунок 8 – Пример списка наиболее похожих пользователей, полученный с помощью кода выше

Уберем лишние столбцы и получим взвешенную оценку за фильм.

```
topUsersRating = topUsersRating.drop(['genres', 'timestamp'], axis = 1)
topUsersRating['weightedRating'] =
topUsersRating['similarityIndex']*topUsersRating['rating']
topUsersRating.head()
```

	similarityIndex	userId	movieId	title	rating	weightedRating
0	0.997176	74212	1	Toy Story (1995)	4.0	3.988706
1	0.997176	74212	3	Grumpier Old Men (1995)	2.0	1.994353
2	0.997176	74212	5	Father of the Bride Part II (1995)	2.5	2.492941
3	0.997176	74212	10	GoldenEye (1995)	1.0	0.997176
4	0.997176	74212	39	Clueless (1995)	3.5	3.490118

Рисунок 9 – Результат формирования взвешенной оценки фильма

Теперь нужно сгруппировать по фильмам и получить сумму взвешенных оценок и получить рекомендацию.

```
tempTopUsersRating = # PUT YOUR CODE HERE
recommendation_df = pd.DataFrame()
recommendation_df['score'] =
tempTopUsersRating['sum_weightedRating']/tempTopUsersRating['sum_similarit
yIndex']

recommendation_df['movieId'] = tempTopUsersRating.index
recommendation_df = recommendation_df.sort_values(by='score',
ascending=False)

# найдем количество рекомендаций с рейтингом выше 4.5
recommendation_df = recommendation_df[recommendation_df.score > 4.5]
recommendation_df = recommendation_df.drop(['movieId'], axis = 1)
print(recommendation_df.shape)  # (59, 1)
# выведем топ-5 рекомендаций
recommended_movies=recommendation_df.merge(movies, how = 'inner', on =
'movieId')
recommended_movies.head()
```

Результат есть (рисунок 10). Но теперь нужно оценить качество этой рекомендации. Задача следующая: разделить исходные данные на train и test. Получить предсказания на train и проверить их на test.

	movieId	score	title	genres
0	353	5.0	Crow, The (1994)	Action|Crime|Fantasy|Thriller
1	78499	5.0	Toy Story 3 (2010)	Adventure|Animation|Children|Comedy|Fantasy|IMAX
2	508	5.0	Philadelphia (1993)	Drama
3	1620	5.0	Kiss the Girls (1997)	Crime|Drama|Mystery|Thriller
4	1552	5.0	Con Air (1997)	Action|Adventure|Thriller

Рисунок 10 – Вывод первых пяти фильмов, которые можно порекомендовать пользователю

Задание 2

Проведите EDA (exploratory data analysis, разведочный анализ данных):

1. Посмотрите, как менялась популярность тегов со временем.
2. Сделайте и оцените кластеризацию тегов и кластеризацию фильмов.
3. Посчитайте и визуализируйте статистики по тегам, жанрам, годам.
4. Изучите "тег геном" для поиска похожих фильмов.

Задание 3

Сделайте рекомендацию фильмов на основе контента:

1. Будем считать, что описание фильма состоит из его тегов.
2. Когда пользователь ставит оценку за фильм, ее нужно засчитать как оценку за теги.
3. Рекомендацией будем считать фильмы с тегами, которые пользователь оценил выше всего.

Порядок защиты и представление результатов выполнения лабораторной работы

Для защиты лабораторной работы студенту необходимо:

1. Предоставить результаты выполнения 3 заданий в виде ссылки на Google Colab.
2. Выполнить дополнительное задание в процессе сдачи работы. Дополнительное задание представляет собой обработку данных лабораторной работы, по сложности аналогичную той, что требовалось выполнить в рамках работы.

Примеры дополнительных заданий:

1. Сделайте неперсонализированную рекомендацию фильма по имеющимся данным. Можно, например, найти фильм с самой высокой средней оценкой по каждому жанру или за какой-то период.
2. В текущем задании был реализован user-based подход. Перепишите код так, чтобы был реализован item-based подход.

Дополнительные материалы

1. Рекомендательные системы [Электронный ресурс]. URL: https://neerc.ifmo.ru/wiki/index.php?title=Рекомендательные_системы (дата обращения: 02.04.2024).
2. Surprise: A Python scikit for recommender systems [Электронный ресурс]. URL: https://surpriselib.com/ (дата обращения: 02.04.2024).

Лабораторная работа №5

ОБРАБОТКА ИЗОБРАЖЕНИЙ

Цель работы: изучение методов обработки изображений с помощью python.

Краткое описание: в ходе выполнения работы необходимо произвести квантование изображения и цветовую субдискретизацию типа 4:2:0, после чего оценить степень сжатия.

Данные: для выполнения работы нужно загрузить по ссылке любую цветную картинку.

Порядок выполнения работы

Подготовка данных

Ниже приведены формулы для перевода RGB в формат YCbCr. В этом формате Y представляет собой картинку в градациях серого (grayscale).

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = -0.1687R - 0.3313G + 0.5B + 128$$
$$Cr = 0.5R - 0.4187G - 0.0813B + 128$$

Переведите картинку в формат YCbCr, используя формулы, описанные выше. Выведите каналы Y, Cb и Cr отдельно.

Задание 1

В задании 1 нужно будет работать только с каналом Y. Предварительно сделайте его копию Y_copy, так как этот канал понадобится нам в задании 2 в неизмененном виде.

Теория

Дан сигнал X, представляющий собой последовательность значений x_i. Для X известно (или может быть легко вычислено) множество допустимых значений. Это множество допустимых значений может быть бесконечным, например, множеством действительных чисел.

В задаче квантования элементам последовательности X сопоставить элементы из множества Y. При этом множество Y обязательно конечно и включает в себя меньшее, чем в X, количество различных элементов.

Множество Y называют аппроксимирующим множеством. Элементы из этого множества называют аппроксимирующими значениями. Входная последовательность – исходная последовательность элементов.

Квантование – это разбиение диапазона допустимых значений сигнала X на конечное число уровней и округление этих значений до ближайших к ним уровней, то есть сопоставление каждому x_i ближайших к нему аппроксимирующих значений.

Равномерное квантование – разбиение диапазона значений отсчетов на отрезки равной длины. Такие отрезки называют квантами. При этом обычно аппроксимирующие значения выбираются в центрах этих отрезков. При неравномерном квантовании отрезки могут быть разной длины, а аппроксимирующие значения могут находиться не в центрах отрезков.

Обозначим через Q множество номеров квантов. Квантованная последовательность – это последовательность номеров квантов, которые соответствуют элементам входной последовательности.

Восстановленная последовательность – это последовательность аппроксимирующих значений, соответствующих элементам входной последовательности.

Скалярное квантование – одному элементу из X сопоставляется ровно одно аппроксимирующее значение. При векторном квантовании одно аппроксимирующее значение может быть сопоставлено сразу нескольким элементам из X.

В равномерном квантовании с переменной скоростью изначально задается ширина квантов (обычно ее называют “шаг”, step). В равномерном квантовании с постоянной скоростью задается количество бит, которое нужно потратить на

один отчет в квантованной последовательности. Вычислительно квантование с постоянной скоростью значительно сложнее, так как нужно находить границы квантов и определять, попадает ли элемент входной последовательность внутрь кванта.

Разберем на примере. Допустим, у нас есть grayscale-изображение. Яркость пикселей – это входная последовательность. Множество допустимых значений элементов входной последовательность – целые числа от 0 до 255. В результате квантования каждый элемент входной последовательности будет заменен ближайшим элементом из аппроксимирующего множества. Получим сначала квантованную последовательность. Для этого нам необходимо задать ширину кванта step. В задаче с яркостью изображений ширина кванта задается степенью числа 2. Фактически в исходном изображении у нас шаг 1, то есть 2^0. Допустим, $step=2^2=4$.

Обозначим через y_q элемент восстановленной последовательности. Тогда, $y_q = \left(\left\lfloor \frac{x_i}{step} \right\rfloor + 0.5 \right) \cdot step$. При этом $\left\lfloor \frac{x_i}{step} \right\rfloor$ – это номер соответствующего кванта.

Допустим, в исходной последовательности встретились пиксели яркости 71, 73, 74. Тогда соответствующие им номера квантов будут равняться 17, 18, 18, а элементы восстановленной последовательности будут равняться 70, 74, 74.

Формулировка задания

Напишите алгоритм равномерного скалярного квантования с переменной скоростью. Примените его к картинке Y_copy. Выведите получившуюся картинку. Вам потребуется подобрать значение step такое, чтобы качество картинки оставалось приемлемым. Выведите первые 10 элементов входной последовательности, квантованной последовательности, восстановленной последовательности. Выведите элементы аппроксимирующего множества.

Для получения квантованной и восстановленной последовательности важно написать разные функции, так как это разные операции.

Задание 2

Ранее уже были приведены формулы для перевода из RGB и YCbCr. В этом задании нужно будет работать в основном с каналами Cb и Cr.

Теория

Ниже на рисунке 11 приведена иллюстрация операции, которая называется pooling (точнее – max pooling). Эта операция проводится на специальных слоях в сверточной нейронной сети, предназначенной для анализа изображений. Суть операции сводится к следующему: в исходной матрице выбираются блоки

заданного размера (в иллюстрации это блок 2×2), в каждом таком блоке выбирается элемент с максимальным значением. Этот максимальный элемент остается, а остальные отбрасываются.

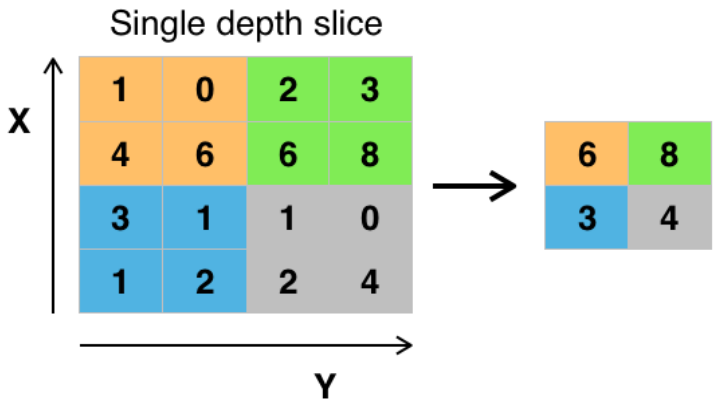


Рисунок 11 – Иллюстрация операции max pooling

Альтернативой операции max pooling является операция average pooling (в этой операции выбирается не максимальный элемент, а считается среднее арифметическое, иногда с округлением).

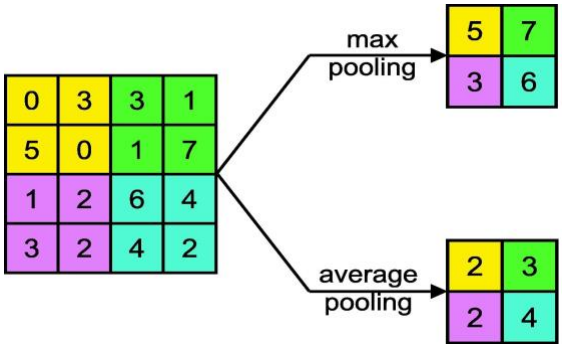


Рисунок 12 – Пример, демонстрирующий различия в операциях max pooling и average pooling

Похожая операция реализуется в стандарте JPEG. Эта часть называется цветовая субдискретизация. Наиболее распространенный тип субдискретизации – это 4:2:0.

Суть этого подхода к обработке изображений сводится к следующему. Каналы Cb и Cr называются цветоразностными. Для цветоразностных каналов делается децимация (прореживание, то есть отбрасывание элементов исходной матрицы). Для субдискретизации типа 4:2:0 нужно в цветоразностных каналах для каждого блока 2×2 оставить только одно значение.

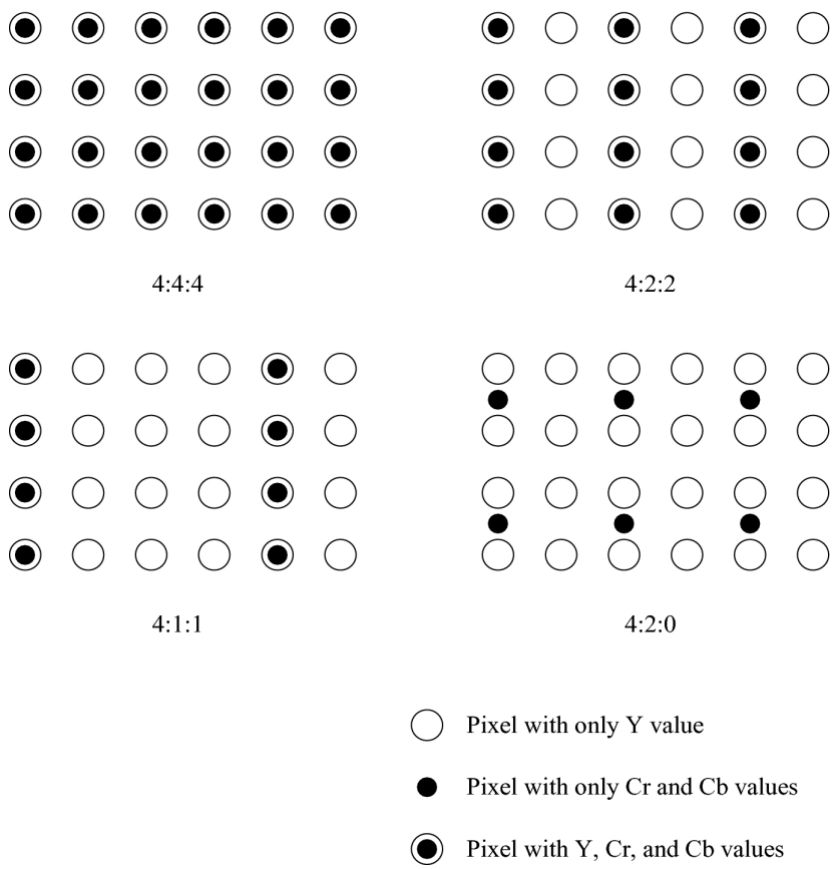


Рисунок 13 – Типы цветовой субдискретизации

После цветовой субдискретиазации количество элементов в каналах Cb и Cr уменьшится. Для 4:2:0 количество элементов сократится в 4 раза. Чтобы снова перейти от формата YCbCr к RGB, нужно восстановить цветоразностные каналы, а затем сделать пересчет по формулам:

$$R = Y + 1.402(Cr - 128)$$
$$G = Y - 0.34414(Cb - 128) - 0.71414(Cr - 128)$$
$$B = Y + 1.772(Cb - 128)$$

Формулировка задания

Для каналов Cb и Cr нужно сделать цветовую субдискретизацию типа 4:2:0. Нужно реализовать три типа обработки: прореживание (то есть отбрасывание всех элементов, кроме одного), среднее и максимальное.

В отдельной функции (методе) реализуйте восстановление цветоразностных каналов. В результате восстановления размеры этих матриц должны стать такими же, как у матрицы Y.

После этого переведите изображение из YCbCr в формат RGB и выведите результат на экран. При переводе используйте каналы Cb, Cr, полученные после восстановления, иначе будет несовпадение по размерам.

Всего должно получиться 6 картинок. Для неизмененного канала Y должно получиться три картинки для каждого типа обработки при цветовой субдискретизации. Также три картинки должны получиться для канала Y после квантования в задании 1.

Задание 3

В этом задании нужно будет оценить качество сжатия данных. Оценивать будем каждую из шести картинок по двум параметрам: RMSE и степень сжатия (через энтропию).

Теория

MSE и ее модификация RMSE (корень квадратный из MSE) – метрики, часто использующиеся в разных областях. В частности, их используют в статистике и машинном обучении для оценки качества регрессионной модели.

$$MSE = \frac{1}{\ell}\sum_{i=1}^{\ell}(\hat{y}_i - y_i)^2$$

Здесь ℓ – количество пикселей в картинке (и в исходной, и после обработки), $\hat{y}_i$ и y_i – яркости пикселей в исходной картинке и в картинке после обработки. Степень сжатия рассчитывается по формуле:

$$C = \frac{H_y}{H_{\hat{y}}}$$

H_y – это среднее количество бит на один пиксель в исходном изображении. В этом задании это должно быть 24 бита. $H_{\hat{y}}$ – это оценка среднего количества бит на один пиксель в изображении после обработки.

Для оценки $H_{\hat{y}}$ потребуется вычисление энтропии для последовательности элементов. Также это значение должно учитывать проведенную в задании 2 субдискретизацию. То есть нужно рассчитывать среднее количество бит не на восстановленном изображении, а на закодированном. Помните о том, что в закодированном изображении количество элементов матрицы в два раза меньше.

Энтропия Шеннона для системы с N возможными состояниями определяется по формуле:

$$H = -\sum_{i=0}^{N} p_i \log_2 p_i$$

где p_i – вероятности нахождения системы в i-ом состоянии. Вероятность может быть оценена через частоту.

Это очень важное понятие теории информации, которое позволяет оценить количество информации (степень хаоса в системе). Чем выше энтропия, тем менее упорядочена система, и наоборот. С помощью энтропии мы рассчитываем среднее количество бит на элемент последовательности при условии, что эту последовательность получится оптимально закодировать.

Энтропия – важное понятие и для машинного обучения. Например, с ее помощью формализуют функционал качества для разделения выборки при построении решающих деревьев, а кросс-энтропия используется как функционал качества при обучении (настройки параметров) разных моделей, в том числе – нейронных сетей.

Приведем код для вычисления энтропии.

```
import numpy as np

def entropy(x):
    _, counts = np.unique(x, return_counts=True)
    probabilities = counts / counts.sum()
    entropy = sum(probabilities * -np.log2(probabilities))

    return entropy
```

Формулировка задания

Для изображения в формате grayscale в задании 1 рассчитайте степень сжатия и MSE. Для каждой их 6 картинок вычислите RMSE и степень сжатия. Выведите результат в удобочитаемом формате.

Картинка сжималась путем снижения яркостного разрешения (в задании 1) и путем цветовой субдискретизации (задание 2). Ответьте на несколько вопросов:

1. Какой из этих методов дал наибольший вклад в сжатие данных?
2. Какой их этих методов внес наибольшую ошибку?
3. Какой тип обработки при цветовой субдискретизации оказался наиболее эффективным?

Порядок защиты и представление результатов выполнения лабораторной работы

Для защиты лабораторной работы студенту необходимо:

1. Предоставить результаты выполнения 3 заданий в виде ссылки на Google Colab.
2. Выполнить дополнительное задание в процессе сдачи работы. Дополнительное задание представляет собой обработку данных лабораторной работы, по сложности аналогичную тем, что требовалось выполнить в рамках работы.

Примеры дополнительных заданий:

1. В картинке, с которой работали в этом задании, поменяйте местами красный и синий каналы.
2. Сгенерируйте последовательность 20 целых чисел в диапазоне от -3 до 3 из нормального распределения. Пусть это будет квантованная последовательность *x*.
    - Оцените вероятности (частоты) значений.
    - Рассчитайте энтропию этой последовательности.
    - Постройте неравномерный код и оцените среднее количество бит на один элемент в этом неравномерном коде (опишите, как вы строили неравномерный код).
    - Выведите коды для каждого элемента входной последовательности.
    - Сравните расчетную энтропию и среднее количество бит в закодированной последовательности.
3. Напишите свою реализацию равномерного скалярного квантователя с постоянной скоростью.

Дополнительные материалы

1. Сэломон Д. Сжатие данных, изображений и звука. – М.: Техносфера, 2006. – 367 с.
2. Гонсалес Р. С., Вудс Р. Е. Цифровая обработка изображений. – М.: Техносфера, 2019. – 1104 с.
3. scikit-image: Image processing in Python [Электронный ресурс]. URL: https://scikit-image.org/ (дата обращения: 01.04.2024).

Лабораторная работа №6

ОБРАБОТКА И КЛАССИФИКАЦИЯ ТЕКСТОВЫХ ДАННЫХ

Цель работы: применение наивного байесовского классификатора для определения тональности текстов на русском языке.

Краткое описание: в ходе выполнения лабораторной работы необходимо провести предобработку сырых текстовых данных и настроить гиперпараметры нескольких векторайзеров так, чтобы точность классификации (accuracy) наивным байесовским классификатором из sklearn составила не менее 0.74.

Данные: в папке Data/lab6 расположена таблица с русскоязычными текстовыми отзывами на предметы одежды. Также эти же данные можно найти по ссылке. Все задания лабораторной работы необходимо выполнять по этим данным.

Описание данных

Формат данных:

review sentiment
Очень плохо упакован, весь мятый и смотрится не так как на фото negative
заказ не пришёл но деньги вернули neautral
Мягкая удобная тёплая. Все замечательно. Ничего не торчит запахов нет. Продавцу + в карму positive
…

Описание полей:

- review – сырой текст отзыва в большинстве случаев на русском языке, встречаются отдельные записи на английском;
- sentiment – тональность отзыва (позитивный, нейтральный, негативный), классы сбалансированы.

Порядок выполнения работы

Предварительная обработка данных

Токенизация

Токенизировать – значит разделить текст на части: слова, ключевые слова, фразы, символы и т.д., иными словами – токены.

Самый простой способ токенизировать текст – разделить с помощью метода split(). Но split упускает очень много всего, например, не отделяет

пунктуацию от слов. Кроме этого, есть ещё много менее тривиальных проблем, поэтому лучше использовать готовые токенизаторы, к примеру:

- Yargy парсер. Документация: https://github.com/natasha/yargy
- Парсер из модуля nltk. Учебник по nltk: https://www.nltk.org/book

```
# токенизация через split()
text = '''
Продаётся LADA 4x4. ПТС 01.12.2018, куплена 20 января 19 года, 10 000 км пробега.
Комплектация полная. Новая в салоне 750 000, отдам за 650 000.
Возможен обмен на ВАЗ-2110 или ВАЗ 2109 с вашей доплатой.
Краснодар, ул. Миклухо-Маклая, д. 4/5, подьезд 1
Тел. 8(999)1234567, 8 903 987-65-43, +7 (351) 111 22 33
И.И. Иванов (Иван Иванович)
'''
tokens = text.split()
print(f"split() определил {len(tokens)} токенов")  # 56

# токенизация через yargy
!pip install yargy  # установка модуля
from yargy.tokenizer import MorphTokenizer
tknzr = MorphTokenizer()
tokens = [_.value for _ in tknzr(text)]
print(f"yargy определил {len(tokens)} токенов")  # 107

# токенизация через nltk
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.tokenize import word_tokenize
tokens = word_tokenize(text)
print(f"nltk определил {len(tokens)} токенов")  # 81
```

В nltk есть довольно много токенизаторов. Токенизатор подбирается, исходя из языка и требований задачи.

Очистка текста

Очистка текста включает в себя несколько этапов:

- удаление пунктуации;
- преобразование регистра;
- удаление стоп-слов.

Сюда же включается исправление опечаток и этапы очистки от лишней информации (тегов html-разметки, url, непонятных символов и др.) и приведения отдельных потенциально важных элементов текстов к единому виду (времени или дат, например).

```
# набор пунктуационных символов зависит от задачи и текста
import string


# удаление стандартной пунктуации
punct = string.punctuation
clean_words = [w.strip(punct) for w in word_tokenize(text)]

# преобразование регистра
clean_words = [w.lower() for w in clean_words if w != '']
```

Стоп-слова – высокочастотные слова, которые не дают никакой информации о конкретном тексте. Они составляют верхушку частотного списка в любом языке. Набор стоп-слов не универсален, он будет зависеть от задачи. Обычно начинают с готовых списков стоп-слов, а потом их дополняют.

```
# список стоп-слов для русского языка в nltk
from nltk.corpus import stopwords
sw = stopwords.words('russian')
print(f"Количество стоп-слов: {len(sw)}")  # 151

# удаление стоп-слов
print([w if w not in sw else print(w) for w in clean_words])
```

Так будет выглядеть текст из примера выше после приведенных в коде шагов предобработки:

```
['продаётся', 'lada', '4x4', 'птс', '01.12.2018', 'куплена', '20', 'января', '19', 'года', '10', '000', 'км', 'пробега', 'комплектация', 'полная', 'новая', None, 'салоне', '750', '000', 'отдам', None, '650', '000', 'возможен', 'обмен', None, 'ваз-2110', None, 'ваз', '2109', None, 'вашей', 'доплатой', 'краснодар', 'ул', 'миклухо-маклая', 'д', '4/5', 'подьезд', '1', 'тел', '8', '999', '1234567', '8', '903', '987-65-43', '7', '351', '111', '22', '33', 'и.и', 'иванов', 'иван', 'иванович']
```

Нормализация текста

Нормализация токенов (token normalization) – это процесс приведения токенов к канонической форме, чтобы устранить несущественные различия между последовательностями символов.

Для многих задач естественно рассматривать как отдельный признак каждое слово, а не каждую его отдельную форму.

Стемминг (англ. stemming – находить происхождение) – это процесс нахождения основы слова для заданного исходного слова. Основа слова не обязательно совпадает с морфологическим корнем слова.

Лемматизация – это сведение разных форм одного слова к начальной форме – лемме.

Пример: лексема *saw* после стемминга может стать буквой *s*, а в ходе лемматизации либо словом *see*, либо *saw*, в зависимости от того, глагол это или существительное.

Стемминг

Стемминг – отсечение от слова окончаний и суффиксов, чтобы оставшаяся часть, называемая stem, была одинаковой для всех грамматических форм слова. Стем необязательно совпадает с морфологической основой слова. Одинаковый стем может получиться и не у однокоренных слов и наоборот – в этом проблема стемминга.

1-ый вид ошибки: *белый*, *белка*, *белье* $\Longrightarrow$ *бел.*

2-ой вид ошибки: *трудность*, *трудный* $\Longrightarrow$ *трудност*, *труд.*

3-ий вид ошибки: *быстрый*, *быстрее* $\Longrightarrow$ *быст*, *побыстрее* $\Longrightarrow$ *побыст.*

Самый простой алгоритм, алгоритм Портера, состоит из 5 циклов команд, на каждом цикле – операция удаления / замены суффикса. Возможны вероятностные расширения алгоритма.

Snowball stemmer - улучшенный вариант стеммера Портера; в отличие от него умеет работать не только с английским текстом.

```
from nltk.stem.snowball import SnowballStemmer

poem = '''
По морям, играя, носится
с миноносцем миноносица.
Льнет, как будто к меду осочка,
к миноносцу миноносочка.
И конца б не довелось ему,
```

```
благодушью миноносьему.
Вдруг прожектор, вздев на нос очки,
впился в спину миноносочки.
Как взревет медноголосина:
Р-р-р-астакая миноносина!
'''
words = [w.strip(punct).lower() for w in word_tokenize(poem)]
words = [w for w in words if w not in sw and w != '']

snowball = SnowballStemmer("russian")
for w in words:
   print("%s: %s" % (w, snowball.stem(w)))
```

Часть результата выполнения стемминга с помощью приведенного выше кода:

```
морям: мор
играя: игр
носится: нос
миноносцем: миноносц
миноносица: миноносиц
льнет: льнет
меду: мед
осочка: осочк
миноносцу: миноносц
миноносочка: миноносочк
конца: конц
…
```

Лемматизация

Лемматизация – процесс приведения словоформы к лемме, т.е. нормальной (словарной) форме. Это более сложная задача, чем стемминг, но и результаты дает гораздо более осмысленные, особенно для языков с богатой морфологией.

Примеры лемматизации:

- кошке, кошку, кошкам, кошкой ⟹ кошка
- бежал, бежит, бегу ⟹ бежать
- белому, белым, белыми ⟹ белый

Pymorphy2 – полноценный морфологический анализатор, целиком написанный на Python. Для русского языка используются словари OpenCorpora и граммемы, принятые в OpenCorpora (с небольшими изменениями).

```
from pymorphy2 import MorphAnalyzer
pymorphy2_analyzer = MorphAnalyzer()

words = [pymorphy2_analyzer.parse(w)[0].normal_form for w in words]
```

Лемматизация слов из примера выше в сравнении с результатом стемминга:

```
морям: мор :море
играя: игр :играть
носится: нос :носиться
миноносцем: миноносц :миноносец
миноносица: миноносиц :миноносица
льнет: льнет :льнуть
меду: мед :мёд
осочка: осочк :осочок
миноносцу: миноносц :миноносец
миноносочка: миноносочк :миноносочек
конца: конц :конец
…
```

Векторизация текста

Допустим, мы хотим применить вот такую модель для классификации текстов. Например, хотим определять спам и не спам.

$$\hat{y} = \psi(w_1 x_1 + w_2 x_2 + ... + w_k x_k + b)$$

$\hat{y} = \{spam, ham\}$, w_i и b - это параметры, которые нужно найти в процессе обучения (настройки, fitting) модели.

А что такое для текстов x_i? Нам нужно получить преобразование текстов в численный вектор, с которым может работать стандартный алгоритм машинного обучения. Как это сделать? Тут может быть много разных ответов. Самая простая идея – это 0 или 1, где 0 – если соответствующего токена нет в тексте, а 1 – если есть.

Мешок n-грамм

Самые мелкие структуры языка, с которыми можно работать, называются n-граммами. У n-граммы есть параметр n – количество слов, которые попадают в такое представление текста.

- Если n = 1 – то смотрим на то, сколько раз каждое слово встретилось в тексте. Получаются униграммы.
- Если n = 2 – то смотрим на то, сколько раз каждая пара подряд идущих слов, встретилась в тексте. Получаются биграммы.

Численное представление текста получается путем подсчета количества n-грамм в текстах. Так как результат не зависит от порядка слов в текстах, то говорят, что такая модель представления текстов в виде векторов получается из гипотезы представления текста как мешка слов.

Прежде чем получать n-граммы, нужно разделить текст на токены и провести все необходимые для задачи этапы предобработки текста. Векторизация – последний этап перед построением модели.

Самый простой способ извлечь признаки из текстовых данных – использовать класс CountVectorizer из sklearn.

Объект CountVectorizer делает следующее:

- строит для каждого документа (каждой пришедшей ему строки) вектор размерности n, где n – количество слов или n-грам во всём корпусе;
- заполняет каждый i-тый элемент количеством вхождений слова в данный документ.

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

# инициализация векторайзера с униграммами, биграммами и триграммами
vectorizer = CountVectorizer(ngram_range=(1, 3))

# fit_transform сначала обучает векторайзер, а потом сразу применает его к набору данных
vectorized_text = vectorizer.fit_transform([" ".join(tokens)])

# первые 5 объектов вектора
pd.DataFrame(vectorized_text.T.todense(),
index=vectorizer.get_feature_names_out(), columns=["n-gram score"]).head()
```

Мешок символьных n-грамм

В некоторых задачах в качестве признаков могут быть использованы n-граммы символов. Для этого необходимо установить в CountVectorizer() параметр analyzer = 'char', то есть анализировать символы.

```
# инициализация векторайзера для символов
char_vectorizer = CountVectorizer(analyzer='char', ngram_range=(2, 8))

# обучаем его и сразу применяем тексту
char_vectorized_text = char_vectorizer.fit_transform([" ".join(tokens)])

# первые 5 объектов вектора
pd.DataFrame(char_vectorized_text.T.todense(),
index=char_vectorizer.get_feature_names_out(), columns=["Char n-gram score"]).head()
```

Символьные т-граммы используются, например, для задачи определения языка. Ещё одна замечательная особенность признаков-символов – для них не нужна токенизация и лемматизация, можно использовать такой подход для языков, у которых нет готовых анализаторов.

TF-IDF векторизация

TfidfVectorizer делает то же, что и CountVectorizer, но в качестве значений выдает TF-IDF каждого слова.

TF (term frequency) – относительная частотность слова в документе:

$$TF(t,d) = \frac{n_t}{\sum_k n_k}$$

IDF (inverse document frequency) – обратная частота документов, в которых есть это слово:

$$IDF(t,D) = \log\frac{|D|}{|d : t \in d|}$$

Перемножаем их:

$$TFIDF(t,d,D) = TF(t,d) \times IDF(i,D)$$

Идея в том, что если слово часто встречается в одном документе, но в целом по корпусу встречается в небольшом количестве документов, у него высокий TF-IDF.

Для получения векторного представления текста с помощью TF-IDF удобно использовать TfidfVectorizer(), действующий, как CountVectorizer().

```
from sklearn.feature_extraction.text import TfidfVectorizer

# инициализация векторайзера для символов
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 4))

# обучаем его и сразу применяем тексту
tfidf_vectorized_text = tfidf_vectorizer.fit_transform([" ".join(tokens)])

# первые 5 объектов вектора
pd.DataFrame(tfidf_vectorized_text.T.todense(),
index=tfidf_vectorizer.get_feature_names_out(), columns=["TF-IDF"]).head()
```

n-gram score	
вот	3
вот дальний	1
вот дальний город	1
вот наш	1
вот наш дом	1

Char n-gram score	
в	3
во	3
вот	3
вот	3
вот д	1

TF-IDF	
вот	0.273861
вот дальний	0.091287
вот дальний город	0.091287
вот дальний город вот	0.091287
вот наш	0.091287

а) б) в)

Рисунок 14 – Первые 5 объектов векторов, полученных с помощью разных векторайзеров: а) мешок n-грамм, б) мешок символьных n-грамм, в) TF-IDF

Помимо диапазона n-грамм, в каждом из трех векторайзеров можно в качестве гиперпараметра передать максимальное и минимальное число вхождения n-граммы в документ (max_df и min_df), а также максимальное количество используемых из словаря n-грамм (max_features). При настройке гиперпараметров векторайзеров их тоже следует учитывать.

Задание 1

Для обработки текста выполните следующие шаги:

- токенизация;
- приведение к нижнему регистру;
- удаление стоп-слов (с расширением стандартного списка);
- очистка текста с помощью регулярных выражений;
- нормализация (лемматизация или стемминг);
- векторизация (с настройкой гиперпараметров).

Обязательными этапами предобработки являются токенизация и векторизация. Необходимо ли включать остальные шаги, нужно решить в

процессе настройки модели. Порядок выполнения шагов предобработки тоже может отличаться от приведенного в списке.

Обязательно использование векторайзеров:

1. Мешок n-грамм.
2. TF-IDF.
3. Символьные n-граммы.

Для каждого из векторайзеров нужно настроить значения следующих гиперпараметров:

- ngram_range;
- max_df;
- min_df;
- max_features.

Построение модели

Наивный байесовский классификатор

Рассмотрим задачу классификации.

Есть объект $x^{(i)}$, для него нужно оценить вероятность того, что его метка класса – это k. Это можно записать с использованием условной вероятности.

$$P(y^{(i)} = k|x^{(i)})$$
$$P(y^{(i)} = k)! = P(y^{(i)} = k|x^{(i)})$$

Условной вероятностью A при условии B, называется $P(A|B) = \frac{P(AB)}{P(B)}$. Здесь $P(AB)$ – вероятность одновременного наступления событий A и B, а $P(B) > 0$.

Теорема (формула) Байеса

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}\,, P(B) > 0$$

Заметим, что если использовать формулу полной вероятности, то можно записать формулу Байеса в виде:

$$P(A_i|B) = \frac{P(B|A_i)\,P(A_i)}{\sum_{i=1}^{n} P(B|A_i)P(A_i)}$$

Как формула Байеса связана с задачей классификации? Пусть X множество объектов, Y конечное множество имен классов, множество $X \times Y$ является

вероятностным пространством с плотностью распределения $p(x, y) = p(y)p(x|y)$. Вероятности появления объектов каждого из классов $P_y = p(y)$ называются "априорными вероятностями классов". Плотности распределения $p_y(x) = p(x|y)$ называются "функциями правдоподобия классов".

По известным плотностям распределения $p_y(x)$ и априорным вероятностям P_y всех классов $y \in Y$ строят алгоритм $a(x)$, минимизирующий вероятность ошибочной классификации. По формуле Байеса:

$$p(y|x) = \frac{p(x|y)\,p(y)}{p(x)}$$

По формуле полной вероятности:

$$p(x) = \sum_{y_i} (p(x|y_i)p(y_i)$$

Априорные вероятности классов P_y можно оценить согласно закону больших чисел, тогда частота появления объектов каждого из классов равна $P'_y = \frac{l_y}{l}$ где $l_y = |X^l_y|, y \in Y$ сходится по вероятности к P_y $при $l_y \to \infty$. Чем больше длина выборки, тем точнее выборочная оценка P'_y.

Функции правдоподобия нам неизвестны, на самом деле, но по данным мы можем построить их эмпирические оценки.

Для применения наивного байесовского классификатора при работе с текстом в sklearn существует MultinomialNB.

Оценка качества классификации

Точность (Accuracy) показывает долю правильных ответов классификатора от общего числа предсказаний.

Точность (Precision, Positive Predictive Value) отражает, какой процент положительных объектов (т.е. тех, что мы считаем положительными) правильно классифицирован.

Полнота (Sensitivity, True Positive Rate, Recall, Hit Rate) отражает, какой процент объектов положительного класса мы правильно классифицировали.

Легко построить алгоритм со 100%-й полнотой: он все объекты относит к классу 1, но при этом точность может быть очень низкой. Нетрудно построить алгоритм с близкой к 100% точностью: он относит к классу 1 только те объекты, в которых уверен, при этом полнота может быть низкая.

F1-мера (F1 score) является средним гармоническим точности и полноты, максимизация этого функционала приводит к одновременной максимизации этих двух «ортогональных критериев».

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}$$

Также рассматривают весовое среднее гармоническое точности и полноты – F_β-меру:

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{(\beta^2 \cdot \mathrm{precision}) + \mathrm{recall}} = \frac{(1 + \beta^2) \cdot \mathrm{tp}}{(1 + \beta^2) \cdot \mathrm{tp} + \beta^2 \cdot \mathrm{fn} + \mathrm{fp}}$$

Изменение β позволяет делать один из критериев (точность или полноту) важнее при оптимизации.

Для расчета сразу нескольких метрик качества классификации удобно воспользоваться функцией classification_report() из sklearn.metrics.

Задание 2

По предобработанным данным постройте несколько моделей по разным векторайзерам с разными гиперпараметрами и оцените качество. В качестве классификатора нужно использовать наивный байесовский классификатор.

Для сравнения моделей с разными векторайзерами между собой используйте метрики precision, recall, f1-score и accuracy. Для этого сформируйте таблицу (датафрейм на pandas), в котором в строках будут разные векторайзеры, а в столбцах разные метрики качества, а в ячейках будут значения этих метрик для соответствующих векторайзеров. Другими словами, таблица должна иметь следующий вид:

Векторайзер и его параметры	**recall**	**precision**	**f1-score**	**accuracy**
Векторайзер 1	…	…	…	…
Векторайзер 2	…	…	…	…
Векторайзер 3	…	…	…	…
…	…	…	…	…
Векторайзер N	…	…	…	…

Порядок защиты и представление результатов выполнения лабораторной работы

Для защиты лабораторной работы студенту необходимо:

1. Предоставить результаты выполнения задания в виде ссылки на Google Colab.
2. Ответить на контрольные вопросы по содержанию работы.

Примеры контрольных вопросов:

1. Какие шаги предобработки из приведенного в задании списка не использовались в вашей работе? Почему?
2. Объясните отличие стемминга от лемматизации. Какой вид нормализации текста использовался в вашей работе?
3. Как влияет указание конкретного значения max_features на формирование вектора документа через TfidfVectorizer()?
4. В чем разница между accuracy и precision?
5. Почему для классификации текстов в этой лабораторной логично применять MultinomialNB, а не другую вариацию наивного байесовского классификатора из sklearn?

Дополнительные материалы

1. Автоматическая обработка текстов на естественном языке и анализ данных : учеб. пособие / Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С. – М.: Изд-во НИУ ВШЭ, 2017. – 269 с.
2. Naive Bayes. Scikit-learn [Электронный ресурс]. URL: https://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes (дата обращения: 02.04.2024).

Лабораторная работа №7

ПОСТРОЕНИЕ И АНАЛИЗ ГРАФА КЛЮЧЕВЫХ СЛОВ

Цель работы: применение навыков обработки текстовых данных и представление их структуры в виде графа.

Краткое описание: в ходе выполнения лабораторной работы необходимо обработать текстовые записи, содержащие частичное или полное дублирование

информации, построить по ним граф учебных сущностей (ключевых слов) и проанализировать его.

Данные: в папке Data/lab7 находятся две таблицы с данными о дисциплинах Университета ИТМО, рабочие программы которых расположены в Конструкторе ОП. Все задания лабораторной работы необходимо выполнять по этим данным.

Описание данных

1. Таблица items.xlsx содержит учебные сущности, встречающиеся в рабочих программах дисциплин и распределенные по предметным областям.
   Формат данных:

   id,name,domain
   202,Алгоритм Форда-Фалкерсона,Теория графов (часть Математики)
   16866,Обратные задачи биофизики,Биотехнологии
   …
   Описание полей:
   - id – идентификатор учебной сущности;
   - name – наименование учебной сущности;
   - domain – предметная область, к которой относится учебная сущность (может отсутствовать).
2. Таблица descriptions.xlsx содержит текстовые аннотации к дисциплинам.
   Формат данных:

   id,title,description
   22694,Анализ и моделирование бизнес-процессов,
   14958,Исследуя виртуальную реальность: методы и подходы,Курс направлен на знакомство с методами взаимодействия и изучения игр. В рамках курса студенты разберут разные подходы к изучению игр с использованием методов цифровых гуманитарных наук, а также классические подходы.
   21258,Графовые базы данных,Из этого курса вы узнаете об одной из разновидностей современных нереляционных баз данных: графовых. Графовые базы данных хранят данные, организованные в графы: структуры, состоящие из узлов, хранящих данные, и связей между ними. На графовые базы данных отлично ложатся социальные сети, рекомендательные системы, другие слабо структурированные данные с большим числом разнообразных связей.
   …
   Описание полей:
   - id – идентификатор дисциплины;

- title – название дисциплины;
- description – аннотация дисциплины (может отсутствовать).

Порядок выполнения работы

Краткий анализ предметных областей

Посчитайте количество разных предметных областей. В таблице с учебными сущностями некоторые предметные области являются частями других предметных областей. Например, есть предметная область "Математика" и есть "Теория графов (часть Математики)". Такие области следует объединить в область "Математика".

Задание 1

1. Выведите датафрейм, в котором будет указано название предметной области и количество разных учебных сущностей, относящихся к конкретной предметной области.
2. Сделайте визуализацию предметных областей.

Обработка дублирующихся и некорректных сущностей

В таблице учебных сущностей существуют дубликаты. На рисунке ниже приведен пример: "Матрица" и "Матрицы" – это на самом деле одна и та же учебная сущность.

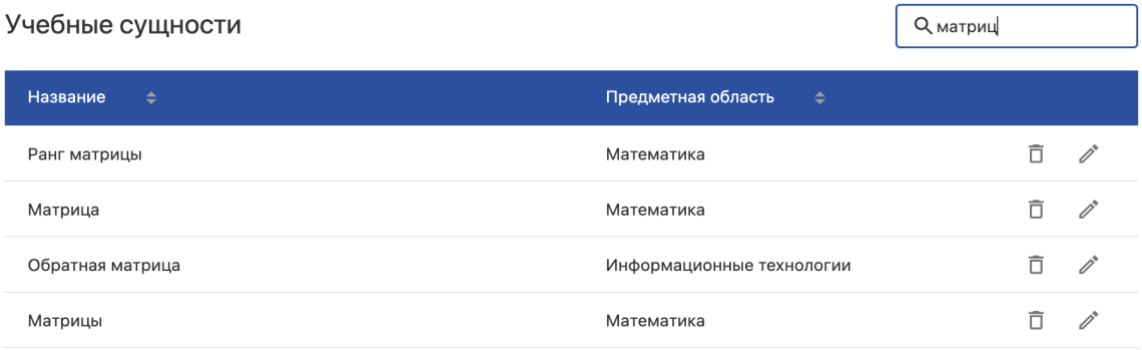


Рисунок 15 – Явные сущности-дубликаты на примере “Матрицы”

Также дубликаты могут быть неявными. В примере на следующем рисунке дублирующиеся сущности подчеркнуты красным цветом. "Git", "GIT", "Навыки работы с Git" и "Система контроля версий git" – это все об одной и той же сущности (рисунок 16).

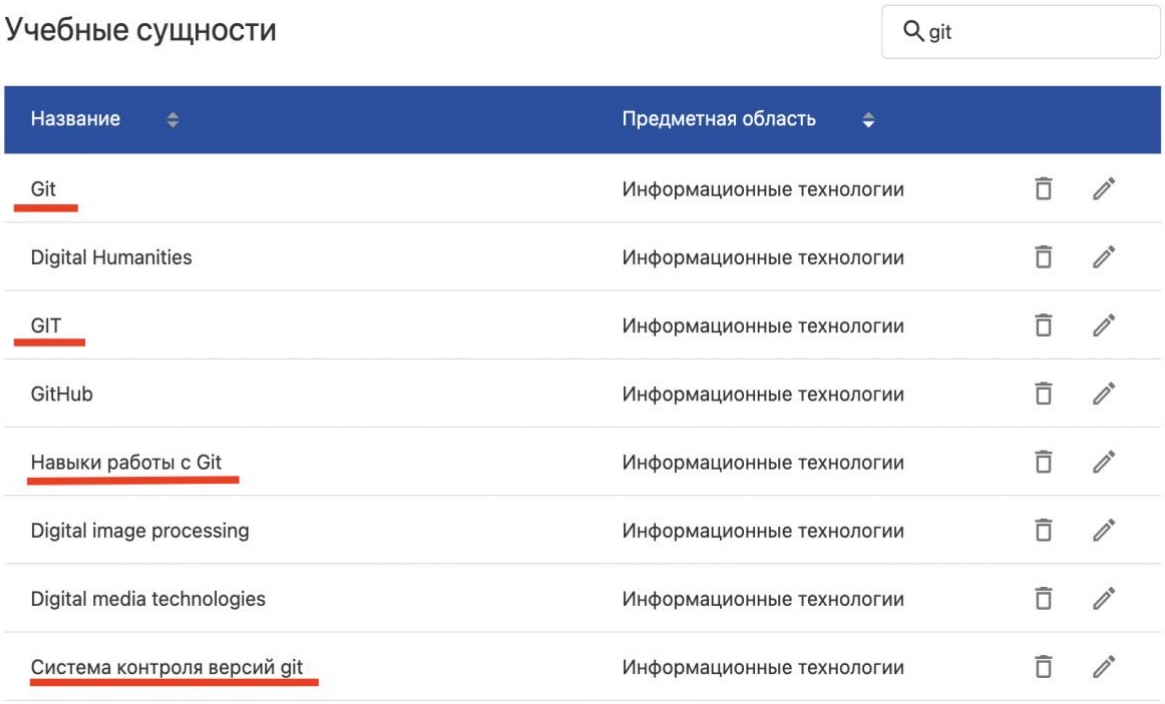


Рисунок 16 – Неявные сущности-дубликаты на примере “Git”

Выделим несколько групп критериев, по которым можно определить, что две или более учебных сущностей являются дубликатами.

Морфологические критерии:

1. Если у двух сущностей разные идентификаторы, и они относятся к разным предметным областям, но их названия полностью совпадают, то это одна и та же сущность.
2. Если есть две сущности, которые различаются только регистром букв, которые входят в их названия, то это одна и та же сущность.
3. Если две сущности совпадают после удаления из них всех знаков препинания, спецсимволов и повторяющихся пробелов в любом месте строки, то это одна и та же сущность.
4. Если две сущности совпадают после удаления из них опечаток, то это одна и та же сущность.
5. Если две сущности совпадают после их перевода на один язык, то это одна и та же сущность.
6. Если после лемматизации двух сущностей множества их лемм совпадают, то это одна и та же сущность.

Критерии, основанные на численных представлениях сущностей:

1. Если косинусное сходство векторов TF-IDF двух сущностей превышает заданный порог, то это одна и та же сущность.
2. Если косинусное сходство контекстных эмбеддингов двух сущностей превышает заданный порог, то это одна и та же сущность.

Критерии некорректности сущностей:

1. Если сущность начинается со строчной буквы, то эта сущность, скорее всего, некорректна.
2. Если сущность содержит более 6 слов, то эта сущность, скорее всего, некорректна.
3. Если сущность заканчивается знаком препинания, то эта сущность, скорее всего, некорректна.
4. Если сущность содержит союз "и", то эта сущность, скорее всего, некорректна.

Задание 2

1. Примените ко всем учебным сущностям перечисленные выше проверки на корректность.
2. Добавьте в датафтейм столбец с логическим значением, указывающим на корректность или некорректность сущности.
3. Постройте визуализацию, отражающую соотношение количества корректных и некорректных сущностей в каждой предметной области.

Задание 3

1. Реализуйте минимум 4 морфологических критерия определения дубликатов. Подсчитайте количество дубликатов, которые удалось найти с помощью этого критерия.
2. Реализуйте минимум 1 критерий выявления дубликатов, основанный на численных представлениях сущностей. Подсчитайте количество дубликатов, которые удалось найти с помощью этого критерия.
3. Добавьте в датафрейм с учебными сущностями столбец duplicates. Поместите в этот столбец список дубликатов для каждой учебной сущности.
4. Покажите на графике количество дубликатов, которые удалось найти с помощью каждого из реализованных критериев.

Граф учебных сущностей

Граф ($G(V, E)$) – математическая модель представления системы объектов, обладающих парными связями. Представляет собой совокупность двух множеств: множества объектов (вершин V) и множества связей между объектами (ребер E).

Теория графов применяется в самых разных областях технических наук. В анализе данных к известным задачам, решаемыми алгоритмами теории графов, можно отнести:

- анализ соцсетей;
- анализ структуры текста.

В общем случае при помощи графов можно визуализировать любую систему объектов, попарно связанных между собой.

Некоторые определения:

- Порядок – количество вершин графа.
- Размер – количество ребер графа.
- Смежные вершины – две вершины, соединенные одним ребром.
- Смежные ребра – ребра, инцидентные одной вершине.
- Степень вершины – количество инцидентных вершине ребер (т.е. сколько ребер связано с вершиной).
- Изолированная вершина – вершина, не связанная ни с одним ребром.
- Висячая вершина – вершина всего с одним ребром.
- Псевдограф – граф, имеющий петли.
- Мультиграф – граф с кратными ребрами (кратные ребра – те, у которых обе концевые вершины совпадают).
- Ориентированный граф – имеет дуги вместо ребер, одна из двух концевых вершин является начальной, вторая – конечной.
- Связный граф – граф, в котором есть путь для любых двух вершин.
- Плотность – отношение числа ребер графа к числу ребер в соответствующем полном графе.
- Полный граф – граф, в котором каждая пара вершин смежна.
- Клика – подмножество вершин, любые две из которых соединены ребром.

Популярной задачей при построении графа является выделение центральных вершин. Подходов к определению центральности вершины существует несколько:

- Центральность по посредничеству (Betweenness centrality) – показывает, через какие вершины проходит наибольшее количество кратчайших путей
- Для каждой пары вершин (s,t) вычисляются кратчайшие пути между ними.
- Для каждой пары вершин (s,t) определяется доля кратчайших путей, которые проходят через рассматриваемую вершину.
- Суммируем эти доли по всем парам вершин (s,t).
- Центральность по собственному вектору или влиятельности (Eigenvector centrality) – показывает, насколько тесно связана вершина с другими

вершинами, определяет централизацию всей структуры графа по собственным векторам

- Центральность по влиятельности означает влияние вершины в пределах графа. Назначаются относительные показатели всем вершинам графа на основе концепции о том, что связи с вершинами с высоким показателем вкладывают больше в показатель рассматриваемой вершины, чем такая же связь с вершиной с низким показателем.
- На этом показателе основан алгоритм PageRank.
- Центральность по степени (Degree centrality) – показывает, сколько ребер исходит из вершины, самая простая оценка важности вершины
- Центральность по степени близости (Closeness centrality) – обратная величина суммы длин кратчайших путей между вершиной и всеми другими вершинами графа; чем более центральна вершина, тем ближе она ко всем другим вершинам $C(x) = \frac{N}{\sum d(y,x)}$, где N – количество вершин графа.

Другой популярной задачей является поиск сообществ на графе. Сообщество на графе – группа вершин, которые теснее связаны между собой, чем с остальным графом. Выделение сообществ является задачей кластеризации. Качество разбиения на сообщества обычно оценивается по модулярности – скалярной величине из отрезка [−1;1], которая показывает, насколько при заданном разбиении на сообщества плотность связей внутри сообщества больше плотности связей между сообществами.

networkx – это модуль для построения и анализа графов. Подробно с модулем можно ознакомиться в документации.

Пример построения графа по тексту

Построим граф для слов некоторого текста. Будем считать, что между двумя словами есть связь, если они идут в тексте подряд. Будем использовать все части речи, но опять же, можно построить граф только с существительными, например.

```
# импортируем необходимые для выполнения задачи модули
from collections import Counter
import matplotlib.pyplot as plt
import networkx as nx
import re
import pymorphy2
```

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')

from nltk.tokenize import word_tokenize
# текст для построения графа
text = """
Белеет парус одинокой
В тумане моря голубом!..
Что ищет он в стране далекой?
Что кинул он в краю родном?..
Играют волны – ветер свищет,
И мачта гнется и скрыпит…
Увы! он счастия не ищет
И не от счастия бежит!
Под ним струя светлей лазури,
Над ним луч солнца золотой…
А он, мятежный, просит бури,
Как будто в бурях есть покой!
"""
# очистка текста от пунктуации
doc = text.replace("\n", " ")
doc = re.sub(r"[^А-Яа-яЁё ]", "", doc).lower()
# токенизация текста
doc = word_tokenize(doc)

# удаление стоп-слов
sw = nltk.corpus.stopwords.words('russian')
doc = [word for word in doc if word not in sw]

# лемматизация текста
analyzer = pymorphy2.MorphAnalyzer()
doc = [analyzer.parse(word)[0].normal_form for word in doc]

# создание зависимых пар слов – биграмм
pairs = list(nltk.bigrams(doc))
print(f"Количество пар слов в тексте: {len(pairs)}")  # 34
# создание и подсчет уникальных пар слов в тексте
pairs = [tuple(sorted(pair)) for pair in pairs]
word_pairs = dict(Counter(pairs))
```

```
print(f"Количество уникальных пар слов в тексте: {len(word_pairs)}")  # 33
# формирование пар слов для графа
edges = [(pair[0], pair[1], val) for pair, val in word_pairs.items()]
# создание графа
G = nx.Graph()
# добавление ребер на граф
G.add_weighted_edges_from(edges)
# увеличение размера вершин для визуализации
node_sizes = [deg*150 for node, deg in G.degree()]

# визуализация графа
plt.figure(figsize=(15,10))
pos = nx.layout.spring_layout(G)
edges, weights = zip(*nx.get_edge_attributes(G,'weight').items())
nx.draw(G, pos, node_color='orange',
      node_size=node_sizes,
      edge_color=range(len(G.edges())), width=2.0,
      with_labels=True, edge_cmap=plt.cm.Blues)
plt.show()
```

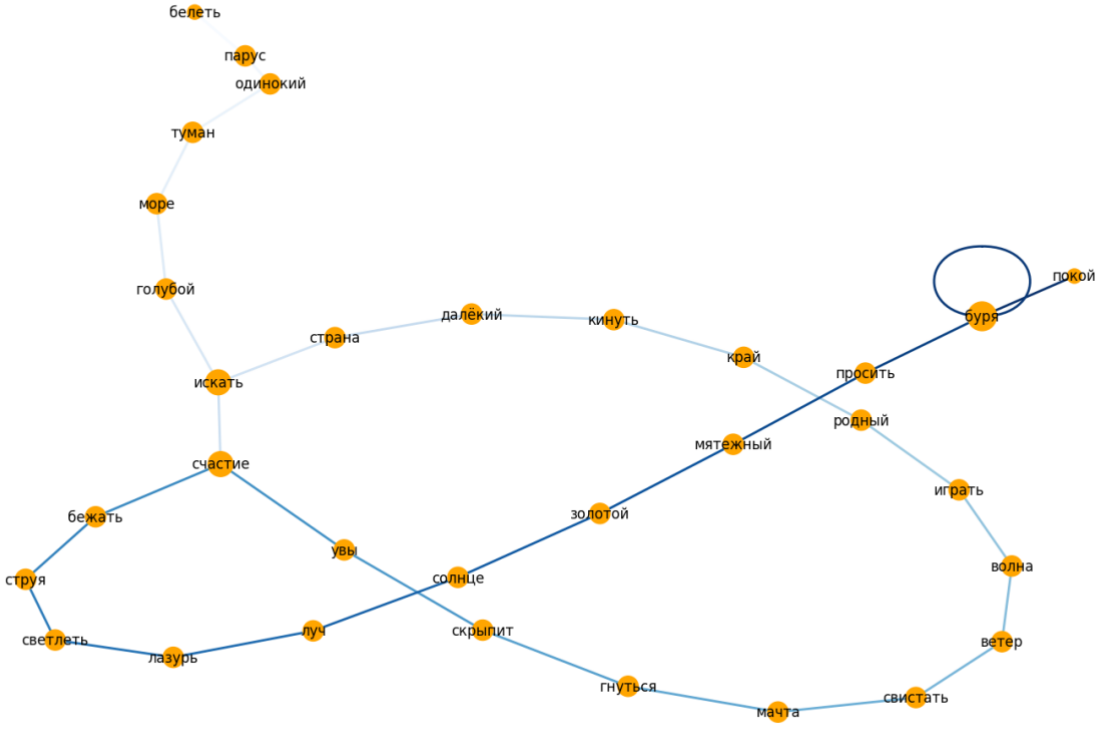


Рисунок 17– Граф слов, отражающих их последовательность и частоту встречаемости пар в тексте

Получился граф, по которому можно отследить связи между словами в тексте и увидеть в целом структуру документа.

Задание 4

Постройте граф ключевых (наиболее частотных по корпусу) слов для аннотаций из таблицы descriptions.xlsx. Для этого необходимо сделать предобработку, включающую токенизацию, лемматизацию и очистку текста. При построении графа учтите следующие условия:

- ключевым словом может быть только существительное;
- ключевое слово не может быть стоп-словом;
- общее количество вершин должно быть в диапазоне от 50 до 100;
- 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;
- при визуализации настройте размер вершины (по степени) и толщину ребер (по весу).

Задание 5

Постройте граф предметных областей для таблицы items.xlsx по следующим правилам:

- считайте, что между двумя предметными областями есть связь, если учебные сущности из них встретились в названии и/или аннотации одной дисциплины;
- учебная сущность входит в название и/или аннотацию дисциплины, если после предобработки, включающей очистку и лемматизацию, строка или ее дубликат полностью входит в текст названия и/или аннотации, прошедший точно такую же предобработку.

Ответьте на вопросы:

1. Есть ли на вашем графе висячие вершины?
2. Какая вершина является центральной по степени?
3. Какое ребро (если ребра есть) имеет наибольший вес?

Порядок защиты и представление результатов выполнения лабораторной работы

Для защиты лабораторной работы студенту необходимо:

1. Предоставить результаты выполнения заданий в виде ссылки на Google Colab.
2. Выполнить дополнительное задание в процессе сдачи работы. Дополнительное задание представляет собой обработку данных лабораторной работы, по сложности аналогичную тем, что требовалось выполнить в рамках работы.

Примеры дополнительных заданий:

1. Проверьте, всегда ли сущности-дубликаты относятся к одной предметной области. Если нет, покажите несколько примеров, когда это не так.
2. Найдите три вершины на графе ключевых слов из задания 4, обладающих максимальными значениями центральности по посредничеству, а также три вершины с наибольшими степенями. Есть ли в полученных списках общие вершины?
3. Придумайте один критерий, не приведенный в списке критериев к заданию 3, и реализуйте его.

Дополнительные материалы

1. NetworkX. Network Analysis in Python [Электронный ресурс]. URL: https://networkx.org/ (дата обращения: 02.04.2024).

Хлопотов Максим Валерьевич
Чернышева Анастасия Вадимовна

Методические указания к выполнению лабораторных работ по обработке и визуализации данных на Python

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49, литер А