ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о программном проекте

на тему **Веб-сервис кластеризации цифрового ассистента студента**
(промежуточный, этап 2)

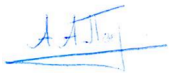
Выполнил:
студент группы БПМИ 1910  Смотрова Кристина Дмитриевна
Подпись И.О. Фамилия

15.04.2021
Дата

Принял:
руководитель проекта Паринов Андрей Андреевич
Имя, Отчество, Фамилия

Младший научный сотрудник
Должность, ученое звание

Департамент анализа данных и искусственного интеллекта
Место работы (Компания или подразделение НИУ ВШЭ)

Дата проверки 26.04 2021 10 
Оценка (по 10-тибалльной шкале) Подпись

Москва 2021

Содержание

1. Основные термины и определения

Веб-сервис – это любое программное обеспечение, которое доступно через Интернет и использует стандартизированную систему обмена сообщениями.

Кластеризация – это группировка множества объектов на непересекающиеся подмножества (кластеры), состоящие из схожих объектов.

Алгоритмы кластеризации, в частности, занимаются разбиением множеств данных на группы со схожими свойствами.

Цифровой ассистент является веб-сервисом или приложением для ПК и смартфонов. Является заменой помощника или «секретаря» для пользователя. В данном проекте цифровой ассистент предоставляет помощь студенту по подбору подходящих тем проектов, которые являются наиболее релевантными для него.

Базой данных называется файл или группа файлов стандартной структуры, служащая для хранения данных.

Для разработки программ, систем программ, работающих с базами данных, используются специальные средства – **системы управления базами данных** (СУБД).

Ассоциативные правила представляют собой механизм нахождения закономерностей между связанными элементами (событиями или объектами).

SQL - декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

REST - архитектурный стиль взаимодействия компонентов распределённого приложения в сети.

2. Введение

Во многих ВУЗах студентам часто предлагается сделать множество проектов, и студенты сталкиваются с трудностями выбора темы. Часто, ко второму курсу студент еще не может определиться, что конкретно ему нравится и чем бы он хотел заниматься. Данный проект создан помочь студенту в выборе темы проекта. Цифровой ассистент предназначен для отображения рекомендаций различных проектов, основанных на интересах студентов и на некоторых исторических данных. Иными словами, ассистент будет предоставлять релевантную информацию о студенческих проектах.

В данном проекте надо создать веб-сервис, предоставляющий нужные рекомендации. Для этого требуется реализовать некоторые алгоритмы кластеризации и нейросети.

Тема данного проекта актуальна, потому что ассистент облегчит выбор проекта студенту и «направит» его. Студент потеряет меньше времени на поиск нужной и интересной темы. Выбирая тему вручную, студент может пропустить некоторые проекты, а цифровой ассистент предложит все подходящие студенту темы.

Цель проекта – разработка мобильного приложения и сайта рекомендательной системы проектов. Команда проекта будет реализовывать задачи, представленные ниже.

Задачи проекта:

1) Реализация на Android, iOS
2) Разработка сайта
3) Веб-сервис кластеризации
4) Веб-сервис рекомендательной системы
5) Разработка архитектуры

3. Обзор и сравнительный анализ баз данных

Одним из самых часто используемых типов баз данных является реляционный тип баз данных. **Реляционные** системы управления базами данных поддерживают реляционную (= таблично-ориентированную) модель данных. Схема таблицы (= схема отношений) определяется именем таблицы и фиксированным числом атрибутов с фиксированными типами данных. Запись (= сущность) соответствует строке в таблице и состоит из значений каждого атрибута. Таким образом, отношение состоит из набора однородных записей. Табличные схемы генерируются путем нормализации в процессе моделирования данных.

Определены некоторые основные операции над отношениями:

- классические операции над множествами (объединение, пересечение и разность)

- выбор (выбор подмножества записей в соответствии с определенными критериями фильтрации значений атрибутов)

- проекция (выбор подмножества атрибутов / столбцов таблицы)

- соединение: специальное соединение нескольких таблиц как комбинация декартова произведения с выбором и проекцией.

На протяжении многих лет многие СУБД были расширены нереляционными понятиями, такими как пользовательские типы данных, а не атомарные атрибуты, наследование и иерархии, поэтому их иногда называют объектно-реляционными СУБД.

Примеры: MySQL, Oracle, Microsoft SQL Server

Документо-ориентированные базы данных также являются очень популярными среди пользователей. В них используется бессхемная организация данных. Иными словами:

- записи не обязательно должны иметь единую структуру, то есть, например, разные записи могут иметь разные столбцы

- столбцы могут иметь более одного значения (массивы)

- записи могут иметь вложенную структуру.

В данном типе баз данных часто используют внутренние обозначения, которые могут быть обработаны непосредственно в приложениях, в основном JSON. Документы JSON, конечно, также могут храниться в виде чистого текста в хранилищах ключей и значений или в системах реляционных баз данных. Однако это потребует обработки структур на стороне клиента, что

имеет свой недостаток: функции, предлагаемые хранилищами документов (например, вторичные индексы), будут недоступны.

Примеры: MongoDB, Microsoft Azure Cosmos DB.

Графовые СУБД, также называемые графоориентированными СУБД или графовыми базами данных, представляют данные в графовых структурах в виде вершин и ребер, которые являются отношениями между вершинами. Они позволяют легко обрабатывать данные в таком виде и просто вычислять конкретные свойства графа, такие как количество шагов, необходимых для перехода от одной вершины к другой.

Графовые СУБД обычно не предоставляют индексы на всех вершинах, прямой доступ к вершинам на основе значений атрибутов в этих случаях невозможен.

Примеры: Microsoft Azure Cosmos DB, Neo4j.

Базы данных **«ключ – значение»**, вероятно, являются самой простой формой систем управления базами данных. Они могут хранить только пары ключей и значений, а также извлекать значения, когда ключ известен.

Эти простые системы обычно не подходят для сложных приложений. С другой стороны, именно эта простота делает такие системы привлекательными в определенных обстоятельствах. Например, ресурс эффективные хранилища ключей и значений часто применяются во встроенных системах или в качестве высокопроизводительных внутри процессных баз данных.

Расширенная форма хранилищ ключ-значение способна сортировать ключи и, таким образом, позволяет выполнять запросы диапазона, а также упорядоченную обработку ключей.

Примеры: Redis, Amazon DynamoDB.

СУБД **временных рядов** — это система управления базами данных, оптимизированная для обработки данных временных рядов: каждая запись связана с меткой времени.

Например, данные временных рядов могут быть получены датчиками, интеллектуальными счетчиками или RFID в так называемом Интернете вещей или могут изображать биржевые тикеры высокочастотной биржевой торговой системы.

СУБД временных рядов предназначены для эффективного сбора, хранения и запроса различных временных рядов с высокими объемами транзакций. Хотя данными временных рядов можно управлять с помощью других категорий СУБД (от хранилищ ключевых значений до

реляционных систем), для решения конкретных задач часто требуются специализированные системы.

Примеры: InfluxDB, Prometheus, Graphite.

The Resource Description Framework (RDF) или **среда описания ресурса** - это методология описания информации, первоначально разработанная для описания метаданных IT-ресурсов. Сегодня он используется гораздо шире, часто в связи с семантической сетью, но также и в других приложениях.

Модель RDF представляет информацию в виде троек в форме субъект-предикат-объект. СУБД, которые способны хранить и обрабатывать такие тройки, называются хранилищами RDF или тройными хранилищами.

Хранилища RDF можно рассматривать как подкласс графовых СУБД, интерпретируя предикат как связь между субъектом и объектом в приведенной выше нотации. Однако хранилища RDF предлагают конкретные методы, выходящие за рамки общих графовых СУБД. Например, SPARQL, SQL-подобный язык запросов для данных RDF, поддерживается большинством хранилищ RDF.

Примеры: MarkLogic, Apache Jena – TDB, Virtuoso.

Хранилища событий — это системы управления базами данных, реализующие концепцию поиска событий. Они сохраняют все события изменения состояния объекта вместе с меткой времени, создавая тем самым временные ряды для отдельных объектов. Текущее состояние объекта может быть выведено путем воспроизведения всех событий для этого объекта с момента 0 до текущего времени. В отличие от этого другие типы СУБД хранят текущее состояние объекта (и не сохраняют историю, если она явно не смоделирована).

Например, для объекта корзины покупок каждая вставка продукта (название продукта, количество, цена) будет сохранена как новое событие.

Поддерживаемые операции с хранилищами событий - добавление новых событий и запрос временных рядов событий для объектов. Не поддерживаются модификации или удаления уже сохраненных событий. Это упрощает поддержание согласованности в распределенных системах.

Из-за соображений производительности многие хранилища событий реализуют концепции хранения snapshot (моментальных снимков или состояний объектов в определенный момент времени).

4. Краткий обзор библиотеки pandas

Pandas – программная библиотека на языке Python для обработки и анализы данных. Работа с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня.

Для работы с данной библиотекой была использована база данных на основе таблицы Excel, где содержалась информация об оценках студентов за проекты. Таблица была приведена к «красивому» виду для удобства работы с ней в pandas.

С помощью методов данной библиотеки можно взаимодействовать с таблицей. А именно: можно удалять столбцы по именам, по номерам столбцов, удалять строки по номерам, удалять строки и столбцы с определенного номера, копировать данные из ячейки в ячейку, определять количество столбцов и строк в таблице, удалять строки по определенным данным, которые содержатся в них, добавлять строки и столбцы различными методами. Все изученные операции представлены на Github в репозитории, который будет прикреплен далее.

5. Http REST Requests

Актуальность рассмотрения протокола HTTP связана с тем, что современные распределенные системы используют его для обмена данными. Методы протокола используется для задач CRUD (Create, Update, Delete).

Свойства протокола:

- Текстовый.
- Поддержка разных типов сообщений. Тело сообщения может содержать любой тип данных.
- Без поддержки соеднинения. Веб-клиент посылает http-запрос (http-request) веб-серверу и отключается от него. Веб-сервер обрабатывает запрос и посылает веб-клиент http-ответ (http-respose).
- Без поддержки состояния (stateless). HTTP не сохраняет данные о клиенте или сервере. Данные могут храниться на сервере или на клиенте. HTTP/1.0 создает новое соединение для каждой запрос\ответ сессии. HTTP/1.1 может использоваться для нескольких сессий.

Основные методы:

- GET
  - Используется для получения информации от сервера.
- HEAD
  - Получение только строки статуса и заголовка без тела сообщения
- POST
  - Используется для отправки сообщений
- PUT
  - Используется для замены текущих данных на сервере
- DELETE
  - Используется для удаления.

После загрузки Requests (библиотеки Python) можно начинать выполнять API-запросы. Для этого надо выполнить следующие пункты:

1. Сформировать URL и тело GET или POST-запроса.
2. Добавить аутентификационные параметры.
3. Непосредственно выполнить запрос.
4. Распарсить результат

6. Основные выводы по этапу и план дальнейшей работы по проекту

Было выполнено:

- Рассмотрены различные типы баз данных, их отличия и преимущества каждого вида.
- Была подробно изучены библиотека Pandas и ее методы работы с базами данных. Кроме того, было рассмотрено как сохранять в базу данных данные из Excel. После этого методы pandas были применены к таблице. Подробнее об этом можно прочитать в драфте проекта, который будет прикреплен далее.
- Было изучено http REST Requests, а именно основные методы работы, которые были применены к рассмотрению веб-сервиса МИЭМа.

В план дальнейшей работы входит:

- Изучение процесса OAuth2 на примере vk.com
- Создание базы данных студентов, для дальнейшей работы с ней на основе веб-сервиса МИЭМа
- Соединить все этапы работы участников команды для получения итогового веб-сервиса

7. Драфт проекта

https://github.com/chrismotrova/DigitalAssistant_project

По ссылке выше находится репозиторий, куда добавлены данные, с которыми я работала.

8. Архитектура программной системы

Архитектура системы состоит из следующих частей:

1. База данных (информация об объектах, информация о пользователях)
    a. Подсистема интеграции
2. Подсистема аналитики
    a. Алгоритмы (классификации)
3. REST веб-сервис
4. Клиент (веб-сайт - django, flask, vue.js, desktop, mobile)

9. Описание функциональных и нефункциональных требований к программному проекту

9.1.Функциональные требования

1. Возможность авторизации посредством логина и пароля.
2. Возможность запуска и остановки работы программы.
3. Предоставление информации о подключенных и о носимых устройствах.
4. Поддержка загрузки конфигурации из файла конфигурации.
5. Поддержка различных операционных систем.

9.2.Нефункциональные требования

1. Предоставление непрерывного доступа к веб-сервису пользователю.
2. Устранение возможности непредвиденной остановки работы веб-сервиса.
3. Надежное хранение информации о пользователях.
4. На ПК/мобильном устройстве, где производится эксплуатация программы необходимо обеспечить регулярные проверки оборудования и программного обеспечения на наличие сбоев и неполадок. Обеспечить защиту устройства от воздействия шпионских программ, программ-шуток, троянских программ и других видов вирусов.

10. Использованные источники

1. DB-Engines – Knowledge Base of Relational and NoSQL Database Management Systems [Электронный ресурс] Режим доступа: https://db-engines.com/en/, свободный. (дата обращения: 06.02.2021)

2. SQL for Data Science | Coursera [Электронный ресурс] Режим доступа: https://www.coursera.org/learn/sql-for-data-science, свободный. (дата обращения: 04.02.2021)

3. Learn R, Python & Data Science Online | DataCamp [Электронный ресурс] Режим доступа: https://www.datacamp.com/, свободный. (дата обращения: 31.01.2021)

4. 2.3 Clustering [Электронный ресурс] Режим доступа: https://scikit-learn.org/stable/modules/clustering.html#birch, свободный. (дата обращения: 31.01.2021)

5. SQL Tutorial [Электронный ресурс] Режим доступа: https://www.w3schools.com/sql/, свободный. (дата обращения: 31.01.2021)

6. Quickstart: Connect and query SQL Server – Azure Data Studio | Microsoft Docs [Электронный ресурс] Режим доступа: https://docs.microsoft.com/en-us/sql/azure-data-studio/quickstart-sql-server?view=sql-server-ver15, свободный. (дата обращения: 31.01.2021)

7. 10 minutes to pandas – pandas 1.2.4 documentation [Электронный ресурс] Режим доступа: https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html#min , свободный. (дата обращения: 14.03.2021)

8. openpyxl – A Python library to read/write Excel 2010 xlsx/xlsm files [Электронный ресурс] Режим доступа: https://openpyxl.readthedocs.io/en/stable/ , свободный. (дата обращения: 14.03.2021)

9. pandas.DataFrame.to_sql – pandas 1.2.4 documentation [Электронный ресурс] Режим доступа: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_sql.html , свободный. (дата обращения: 28.03.2021)

10. Http REST Requests Flask [Электронный ресурс] Режим доступа: ipython notebook, свободный. (дата обращения: 30.03.2021)

11. Календарный план

1. Ноябрь – февраль (КТ1): алгоритмы кластеризации, типы СУБД, алгоритмы ассоциативных правил. Обзор инструментов.
2. 28.02.2021: Реализация таблицы для хранения данных о пользователях.
3. Апрель (КТ2): разработка программы для выполнения подзадачи проекта, делаем эксперименты на данных.
4. Май-июнь (КТ3): Интеграция различных подсистем в единую систему.