МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
Заведующий кафедрой, доцент, к. т. н.
_____ М.С. Воробьева
02.07. 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
магистерская диссертация

РАЗРАБОТКА РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ДЛЯ АБИТУРИЕНТОВ, ВЫБИРАЮЩИХ ОБРАЗОВАТЕЛЬНУЮ ПРОГРАММУ

02.04.03 Математическое обеспечение и администрирование информационных систем
Магистерская программа «Разработка технологий Интернета вещей и больших данных»

Выполнила работу
студентка 2 курса
очной формы обучения — *Подпись* — Беккер Анна Александровна

Научный руководитель
д. пед. н., профессор — *Подпись* — Захарова Ирина Гелиевна

Рецензент
ст. преп. кафедры
программного обеспечения ТюмГУ — *Подпись* — Павлова Елена Александровна

Тюмень
2021

ОГЛАВЛЕНИЕ

СПИСОК ТЕРМИНОВ

Бэкенд (англ. back-end) – программно-аппаратная часть сервиса.

Сериализация – процесс перевода какой-либо структуры данных в последовательность байтов.

Стемминг – это процесс, во время которого идет поиск основы заданного слова. В данном процессе основа слова не обязательно совпадает с морфологическим корнем слова.

Токенизация – процесс, во время которого текст разбивается на слова или на другие измеримые части текста (например, предложения).

Фреймворк – программная платформа, которая определяет структуру программной системы, программного обеспечения и объединяет в себе различные технологии за счёт API (описание способов, которыми одна компьютерная программа может взаимодействовать с другой)

Фронтенд (англ. front-end) – клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса.

«Get» запрос – запрос, в котором все данные передаются в строке запроса

«Post» запрос – запрос, в котором данные передаются в теле сообщения

ВВЕДЕНИЕ

Каждый год тысячи абитуриентов выбирают направления подготовки для поступления в различные университеты. При выборе направления обучения важными являются различные факторы и для каждого поступающего они могут разниться. Кто-то обращает внимание на проходной балл, кому-то важны определенные предметы в ходе обучения, а другие хотели бы изучать определенные технологии углубленно. Зачастую описание образовательных программ схожи друг с другом, а учебные планы содержат в себе большое количество изучаемых предметов. Кроме того, что изучение описаний дисциплин может занять огромное количество времени, эти описания содержат узконаправленные термины и детали, которые не всегда понятны школьнику или его родителям. При этом выпускники схожих по описанию программ при окончании обучения получают различные навыки и даже по одним и тем же дисциплинам различный уровень знаний, ведь на разных направлениях материал зачастую отличается объемом и уклон делается на разные дисциплины.

Как выбрать наиболее подходящее направление обучения человеку без специальной подготовки в определенной области знаний? Данное исследование направлено на решение актуальной и важной задачи выбора образовательной программы, наиболее подходящей конкретному абитуриенту. Предполагаемый подход к решению описанной задачи не был представлен ранее, что подтверждается анализом существующих публикаций в данной области.

Таким образом, целью данной работы стала разработка рекомендательной системы для абитуриентов, которые выбирают образовательную программу, на примере данных института математики и компьютерных наук (ИМиКН) Тюменского государственного университета.

Для достижения указанной цели было необходимо решить следующие задачи:

1. Изучить существующие решения
2. Изучить имеющиеся текстовые данные о программах обучения ИМиКН Тюменского государственного университета

3. Выделить критерии различий между образовательными программами
4. Разработать интерфейс рекомендательной системы, который позволит наглядно представить различия направлений обучения
5. Разработать форму запроса абитуриента к рекомендательной системе
6. Разработать алгоритм, выявляющий наиболее подходящие образовательные программы для запроса абитуриента

Данная система рекомендаций разрабатывалась в рамках научного проекта РФФИ №19-37-51028 и является частью общей системы «Цифровой след студента».

Основным источником информации о направлениях обучения и дисциплинах являются описания учебных программ, а также тексты рабочих программ дисциплин (РПД). Данные тексты хранятся в базе данных HBase. Кроме того, для взаимодействия абитуриента и системы рекомендации были выбраны слова, понятные выпускнику. Чтобы обеспечить доступность терминологии школьнику, было решено использовать школьные словари в качестве источника данных ключевых слов. Использование школьных словарей позволяет представить информацию из текстов РПД понятными для выпускника терминами. В качестве источника слов были взяты школьные словари терминов по информатике за 10 и 11 классы И. Г. Семакина, Е. К. Хеннер [1] и словарь по математике, предназначенный для молодежи и школьников [2].

Для подготовки и защиты выпускной квалификационной работы использовались поиск, анализ информации, системный подход для решения поставленных задач; приемы критического анализа проблемных ситуаций, а также средства и методы саморазвития и самореализации; методики межкультурного взаимодействия; умение расставлять приоритеты собственной деятельности при работе в общем проекте в соответствии с командной стратегией для достижения поставленной цели.

Формулирование выводов по итогам проведенной работы осуществлялись с учетом применения современных коммуникативных технологий (в том числе

на иностранном языке) для представления результатов на академических, профессиональных, экспертных ИТ-мероприятиях.

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ МЕТОДОВ ВИЗУАЛИЗАЦИИ РАЗЛИЧИЙ ОБРАЗОВАТЕЛЬНЫХ ПРОГРАММ

В настоящее время количество информации растет все быстрее. Для успешного анализа информации необходимо эффективно обрабатывать поступающие сырые данные и визуализировать их для дальнейшего анализа. В рамках данной работы необходимо было извлекать ключевые слова из текстов, подсчитывать частоту их повторения. Для этого были рассмотрены различные подходы и методы обработки текстовой информации, методы извлечения ключевых слов, методы визуализации текстов на естественном языке.

1.1 ОПРЕДЕЛЕНИЕ КЛЮЧЕВОГО СЛОВА

Объемы информации, которую необходимо анализировать растут с каждым годом. Но не всегда нужен анализ целого текста. Зачастую смысл текста может быть ясен из ключевых слов, которые наполняют текст. В связи с этим необходимо уметь выделять ключевые слова из текстов. Эта задача является одной из сложнейших задач лингвистики текста. В первую очередь были рассмотрены определения того, что в тексте считать ключевым словом. Существование различных подходов к определению самого понятия ключевое слово было выявлено при анализе предметной области. Общенаучное представление ключевого слова – это слово, которое определяет содержание текста и передает его основной смысл, но кроме этого существуют и другие определения. Так в работе Иванова В.К., Виноградовой Н.В. [3] определение ключевого слова объединяется из нескольких источников в одно, наиболее полное. Ключевые слова обладают следующими признаками:

1. Отражение темы текста.
2. Порядок ключевых слов может быть важен: в нем неявно отражается тема текста.
3. Емкое представление содержания текста. В наборе ключевых слов содержится самый важный смысл текста.

При этом термины «ключевое слово» и «ключевое словосочетание» рассматриваются как синонимы. Данное определение подходит для нашей работы наилучшим образом.

1.2 ПОДХОДЫ К ИЗВЛЕЧЕНИЮ КЛЮЧЕВЫХ СЛОВ

Были рассмотрены подходы к извлечению ключевых слов. Так, в работе Ванюшкина А.С., Гращенко Л.А. [4] представлен обзор различных методов и алгоритмов извлечения ключевых слов из текстов. Выделяют статистические и основанные на машинном обучении методы. По наличию элементов обучения выделяют необучаемые, обучаемые и самообучаемые методы извлечения ключевых слов. Обучаемые методы предполагают использование разнообразных лингвистических ресурсов для создания критериев принятия решения, что является ключевым словом. Если же обучение ведется без учителя или с подкреплением – метод классифицируют как самообучаемый. Необучаемые методы включают в себя выделение ключевых слов без опоры на контекст на основе моделей и правил. Для нашей работы использовался необучаемый метод извлечения ключевых слов. Это связано с тем, что такие методы хороши для расширяющихся корпусов текстов, таких как научные работы или нормативные акты. Также в работе представлена стандартная схема обработки текста для поиска ключевых слов. Согласно этой схеме, анализ делится на три этапа.

1. Сначала проводится предварительная предобработка или анализ данных: извлекаются тексты из различных источников их хранения, преобразуются в удобные для анализа структуры, создаются или применяются словари стоп-слов, для выделения графем, морфем применяются лемматизаторы, стеммеры и т.д. Возможно применение токенизации.
2. Затем следует этап распознавания или классификации информации. На этом этапе применяются определенные правила из базы эталонных правил, классификаторы. В результате получается расширенный список ключевых слов.

3. И третий этап – постобработка данных. Применяются индивидуальные пользовательские настройки и формируется итоговый список ключевых слов.

1.3 ХРАНЕНИЕ ИНФОРМАЦИИ

Современную систему рекомендаций трудно представить без какого-либо хранилища данных. Будь то облачное хранилище или локальная база данных. Данные в наиболее распространенных типах работающих сегодня баз данных обычно моделируются в виде строк и столбцов в серии таблиц, чтобы сделать обработку и запросы данных эффективными. После этого можно легко получить доступ к данным, управлять ими, изменять, обновлять, контролировать и систематизировать. Большинство баз данных используют язык структурированных запросов (SQL) для записи и запроса данных. Но актуально также и использование NoSQL хранилищ, в которых удобно хранить не структурированную информацию, а также разреженные данные. Когда говорят о NoSQL базах данных, то имеют в виду не всегда одинаковые по устройству системы. В связи с этим для этих разнородных систем много общих характеристик. Представим основные из них [5]:

Не используется SQL. Имеется в виду ANSI SQL DML – язык управления данными -, так как многие базы пытаются использовать языки запросов, похожие на общеизвестный SQL синтаксис.

Неструктурированные. В отличие от реляционных баз данных, в NoSQL базах структура данных не регламентирована (или слабо типизирована, если проводить аналогии с языками программирования) — в отдельной строке или документе можно добавить произвольное поле без предварительного декларативного изменения структуры всей таблицы. Такой подход позволяет легко расширять систему и модифицировать её – не нужно менять всю базу данных, необходимо лишь сделать изменения в коде программы.

Представление данных в виде агрегатов. Реляционная модель подразумевает нормализацию данных. Из-за этого бизнес-сущности приложения хранятся в такой модели в разных таблицах данных. Но в NoSQL хранилищах бизнес-сущности приложения являются целостными объектами.

Слабые ACID свойства (Atomicity — Атомарность, Consistency — Согласованность, Isolation — Изолированность, Durability — Прочность). Долгое время все реляционные базы данных гарантировали тот или другой уровень изоляции — либо за счет блокировок при изменении и блокирующего чтения, либо за счет логгирования. Но с ростом количества данных стало понятно, что нельзя обеспечить для них транзакционность набора операций с одной стороны и одновременно получить высокую доступность и быстрое время отклика с другой. Более того, даже обновление одной записи не гарантирует, что любой другой пользователь моментально увидит изменения в системе, но это время обновления данных стараются сделать как можно меньше. На самом деле слабые ACID свойства не означают, что их нет вообще. В большинстве случаев приложение, работающее с реляционной базой данных, использует транзакцию для изменения логически связанных объектов, что необходимо, так как это разные таблицы. Чтобы добиться такого же уровня изоляции, что и в реляционной базе данных, необходимо правильно проектировать модели данных в NoSQL базе.

Распределенные системы, без совместно используемых ресурсов. Данный пункт не относится к тем базам данных, структура которых плохо распределяется по разным узлам. Существует проблема вертикальной масштабируемости системы – она возникла из-за невероятно быстрого роста потока информации и необходимости обрабатывать её в установленное время. Даже кластеризованные (разделенные) на массиве дисков обычные реляционные базы данных не могут справиться с проблемами скорости, масштабируемости и

пропускной способности. Для решения данных проблем применяется горизонтальное масштабирование. Для этого быстрой сетью соединяют несколько независимых серверов и каждый сервер работает только лишь с часть данных и обрабатывает только часть запросов на чтение и обновление. При применении такой архитектуры расширение требует лишь добавление нового сервера в кластер. Это повышает время отклика хранилища, его пропускную способность и, конечно, емкость необходимо лишь добавить новый сервер в кластер. Кроме того, если новый узел добавляется, то NoSQL база данных самостоятельно производит процедуры шардинга (разделение данных по узлам), репликации (копирование данных на другие узлы при обновлении), а также обеспечивает отказоустойчивость (результат будет получен даже если один или несколько серверов перестали отвечать).

NoSQL базы в основном с открытым кодом и созданы в 21 столетии

Таким образом, развитие NoSQL баз данных идет гигантскими шагами и все больше разработчиков их используют. Одними из наиболее популярных примеров NoSQL баз данных являются Hadoop / HBase, Cassandra, Amazon SimpleDB, MongoDB [6]. Все вышеописанное не говорит о том, что обычные реляционные базы данных устаревают и становятся рудиментом. Напротив, они будут использоваться и развиваться все также активно. Но теперь совместно с ними будут использоваться и NoSQL базы данных, дополняя возможности реляционных.

В качестве базы данных в данной работе используется PostgreSQL - объектно-реляционная система баз данных с открытым исходным кодом с более чем 30-летним активным развитием, которая заслужила себе прочную репутацию за надежность, функциональность и производительность [7]. Её можно применять на множестве различных платформ. Данная СУБД включает следующие функциональные возможности:

Транзакции

Вложенные запросы

Представления

Ссылочная целостность - внешние ключи

Сложные блокировки

Типы, определяемые пользователем

Наследственность

Правила

Проверка совместимости версий

Кроме того, PostgreSQL постоянно поддерживается разработчиками, а производительность и надежность растут от версии к версии.

В качестве хранилища метаданных используется Apache HBase - это база данных Hadoop, распределенное, масштабируемое хранилище больших данных [8]. Данное хранилище является одной из самых популярных не реляционных баз данных в мире Big Data [9]. Данное хранилище является колоночно - ориентированным, то есть вся информация хранится в ячейках, которые группируются в колонки, а не в строки. HBase обладает следующими достоинствами:

HBase хорошо подходит для разреженных наборов данных из-за специфической модели данных, которая не ограничивает число столбцов, которые можно сгруппировать в группы или семейства (column families), а пустые значения хранить не требуется

Данные упорядочены по строковым ключам при динамическом секционировании (partitioning) диапазона строк

Встроенный механизм временных меток (timestamp), которые добавляются автоматически, но могут быть изменены вручную

Интеграция с различными внешними инструментами

Высокая производительность и быстрота работы, в том числе в режиме реального времени

Высокая доступность и отказоустойчивость, обеспечиваемые с помощью репликации через центр обработки данных, неделимые и согласованные

операции на уровне строк, а также автоматическое распределение нагрузки и балансировки таблиц с помощью механизмов регионализации. Распределенная файловая система Hadoop (HDFS) и Amazon S3 используются как отказоустойчивое хранилища данных

Способность к масштабированию

Недостатков у данного хранилища не так много, но все же есть:

Некоторые, особенно сложные запросы (примерно 1% от общего количества) могут выполняться медленнее (порядка 300 миллисекунд)

Индексация возможна только по одному полю (Row Key)

Выбор баз данных обусловлен в первую очередь интеграцией разработанной системы рекомендаций в основную систему «Цифровой след студента», где были выбраны именно эти базы данных.

1.4 ПРИЕМЫ И МЕТОДЫ ПРЕДОБРАБОТКИ ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

В настоящее время объемы новой информации растут каждую секунду. И всю эту информацию необходимо обрабатывать и анализировать. Но максимальную пользу из этой информации можно извлечь лишь при правильной обработке. Цель предобработки текста – превратить сырой текст в удобный для дальнейшей работы формат. Наиболее необходимые и потому часто встречающиеся этапы обработки это [10]:

Приведение всех букв к одному регистру. Это необходимо для сравнения слов между собой, поиска.

Обработка числовых данных. Обычно это либо замена на текстовый эквивалент, либо удаление цифр (чисел). Для этого зачастую используются регулярные выражения

Очищение текста от знаков препинания. Чаще всего удаление символов производится из заранее заданного набора. Этот этап обработки необходим для корректного подсчета слов, разделения предложений на

слова. С другой стороны, иногда пунктуация, наоборот, важна – для выделения смысловых частей, интонации и так далее.

Удаление пробельных символов (whitespaces) – этот этап необходим для удаления лишних пробелов

Токенизация (обычно реализуется на основе регулярных выражений) – разбиение текста на слова или предложения, т.е. на определенные единицы, удобные для дальнейшего анализа. Для этого шага создается словарь, в который сохраняют уникальные лексемы, встретившиеся в текстах. На этих этапах ученые сталкиваются с несколькими проблемами. Первая из них – это богатая морфология языка, т.е. развитая система склонений и спряжений слов. Из-за этого возникает сложность с составлением словаря: необходимо объединить и найти все словоформы одного и того же слово. Такая проблема была и в данной работе, так как составлялся словарь ключевых слов и осуществлялся поиск по нему. Вторая проблема – сложные, много коренные слова. В зависимости от задачи такие слова можно разбить на несколько, но можно учитывать и как самостоятельное слово. И третья проблема – определение границ слова. Обычно слова разделяются пробелами, но в некоторых случаях – дефисом, а в некоторых языках (например, японском) между словами вообще нет пробелов.

Удаление стоп-слов. Стоп-слова – это слова, знаки, символы, которые не несут определенной смысловой нагрузки в тексте, они являются шумом и мешают анализу данных. В современных библиотеках по обработке текстов на естественном языке существуют списки таких стоп-слов, но они часто бывают не полными и их необходимо расширять в зависимости от темы текстовых данных и поставленной задачи.

Стемминг. Процесс стемминга позволяет привести слово к его основной форме. Это необходимо для того, чтобы находить количество

корректных словоформ, значения которых схожи, но написания отличаются суффиксами, приставками, окончаниями и прочим. Суть метода заключается в том, чтобы найти основу слова, для чего, начиная с конца и начала слова, по очереди отсекаются его части. Правила вырезания для стеммера (инструмента, которым проводят стемминг) создаются заранее. Чаще всего это регулярные выражения, что делает этот метод очень трудоемким, потому что при подключении нового языка требуется новое лингвистическое исследование. Второй недостаток этого метода заключается в том, что информация может быть потеряна при вырезании частей, например, мы можем потерять информацию об определенной части речи - но этот недостаток имеет влияние в зависимости от задачи – иногда, часть речи не важна в анализе.

Лемматизация. Это подход используется как альтернатива стемминга – использовать их совместно бессмысленно. Суть лемматизации – это приведение слова к словарной форме — лемме. В русском языке, нормальная форма, например, для существительных — именительный падеж, единственное число, для прилагательных — именительный падеж, единственное число, мужской род, для глаголов, причастий, деепричастий — глагол в форме инфинитива несовершенного вида.

Векторизация. Цель векторизации – посчитать, насколько часто слово встречается в тексте. Для этого для документа формируется вектор, соответствующий размерности словаря, в котором каждому слову выделяется своя размерность. Такой подход называется мешком слов (bag-of-words). Данный подход обладает как плюсами, так и минусами. Плюсом мешка слов является простая реализация, однако данный метод теряет часть информации, например порядок слов. Для уменьшения потери информации можно использовать мешок N-грамм (добавлять не только слова, но и словосочетания), или использовать методы векторных представлений слов. Векторные произведения слов снижают

ошибку для слов, которые пишутся одинаково, но имеют разные значения.

1.4.1 ОБЗОР БИБЛИОТЕК ДЛЯ РАБОТЫ С ТЕКСТАМИ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

В ходе разработки модуля рекомендательной системы было принято решение использовать язык программирования Python. Python – интерпретируемый язык программирования с огромным количеством открытых библиотек для работы с различной информацией. Этот язык широко используется для научных проектов. Кроме того, он имеет простой синтаксис и прозрачную семантику языка. Также аргументов в пользу выбора языка Python стало то, что он уже используется для разработки системы «Цифровой след студента» в рамках гранта, в которую данную рекомендательную систему необходимо было внедрить. Существует множество библиотек для работы с текстами на естественном языке с помощью Python.

Первая рассмотренная библиотека – это библиотека nltk [11]. NLTK - ведущая платформа для создания программ на Python для работы с данными на человеческом языке. Библиотека была разработана Стивеном Бердом и Эдвардом Лопером в Пенсильванском университете. В области обработки естественного языка эта библиотека сыграла ключевую роль в различных прорывных исследованиях. Она предоставляет простые в использовании интерфейсы для более чем 50 корпусов текстов, а также набор библиотек обработки текста для классификации, токенизации, стемминга, тегирования, синтаксического анализа и семантического рассуждения. Данное средство используется для токенизации при обработке «сырых» текстов РПД, а также для выявления в текстах так называемых «стоп-слов» - т.е. слов, которые не несут в себе основной смысл текста, а лишь связывают слова между собой в единую речь. Кроме того, nltk поддерживает обработку текстов на русском языке.

Еще одна библиотека, позволяющая обрабатывать тексты – это pymorphy2 [12]. Библиотека pymorphy2 - это морфологический анализатор и генератор для русского и украинского языков. Библиотека реализована на языке

программирования Python, кроме того, некоторые её части написаны языке C++. Библиотека pymorphy2 распространяется с открытым исходным кодом. В данной работе она используется для приведения слов к нормальной форме. Это необходимо для сравнения слов между собой с разными окончаниями и формами слова. В зависимости от входных данных и требуемых операций pymorphy2 может предоставить скорость обработки от нескольких тысяч слов в секунду до более ста тысяч слов в секунду.

Полезным инструментом также является библиотека TextBlo [13]. Она имеет простой интерфейс и не требует долгого изучения для использования. В ней есть все базовые функции, такие как тегирование частей речи, анализ тональности, классификация. Минусом является её низкая скорость работы. Она не подходит для больших объемов данных.

Мощным инструментов для анализа текстов является библиотека Стэндфордского университета CoreNLP [14]. Она написана на Java и имеет высокую скорость работы. Также, её можно интегрировать с некоторыми инструментами библиотеки nltk. CoreNLP поддерживает различные языки, в том числе и русский язык. Но согласно исследованию [15], в котором рассматривались основные принципы работы библиотеки с текстовыми фрагментами, а также были разработаны способы взаимодействия CoreNLP с текстовыми данными на русскому языке, есть много не точностей при работе с русским языком. Кроме того, библиотека достаточно объемная и у нее высокий порог вхождения, что осложняет поддержку приложения с её с использованием.

Еще одна рассмотренная библиотека – Pattern [16]. В ней есть инструменты для:

Сбора данных: веб-сервисоы (Google, Twitter, Википедия), веб-искатель, HTML DOM-парсер

Обработки естественного языка: разметка частей речи, поиск по n-граммам, анализ настроений, WordNet

Машинного обучения: модель векторного пространства, кластеризация, классификация (KNN, SVM, персептрон)

Сетевой анализ: центральность графа и визуализация.

Она хорошо документирован, тщательно протестирована с помощью более чем 350 модульных тестов и поставляется в комплекте с более чем 50 примерами.

В результате обзора библиотек, были выбраны библиотеки nltk и pymorphy2 как наиболее подходящие и протестированные для русского языка, а также имеющие высокую производительность для работы с большими объемами данных.

1.4.2 ОБЗОР ФРЕЙМВОРКОВ ДЛЯ PYTHON

Сложно представить современную разработку приложений без использования какого-либо фреймворка. Фреймворки позволяют использовать комбинацию различных технологий в доступном и удобном формате. Кроме того, фреймворки повышают производительность и эффективность работы приложений. В ходе работы рассматривались фреймворки с открытым исходным кодом. Веб-фреймворки можно сгруппировать в две категории: фреймворк полного стека (множество модулей) и микро-фреймворк (небольшой и с ограниченными возможностями).

Django [17] — веб-фреймворк полного стека, подходящий для больших и сложных проектов. Этот фреймворк бесплатный и с открытым исходным кодом. Создан и поддерживается сотней опытных веб-разработчиков.

Flask [18] — фреймворк для создания веб-приложений на языке программирования Python. Относится к категории микро-фреймворков — он обеспечивает минималистичные каркасы веб-приложений, только с базовыми возможностями. Flask также очень полезен для разработки API (Application Programming Interface). По сравнению с Django более прост, лучше подходит для обучения или если необходимо больше контролировать те или иные вещи вручную. Для не нагруженных проектов Flask использовать удобнее [19].

Нельзя сказать, какой фреймворк лучше, так как в настоящее время фреймворков достаточно много и большинство из них подходят под одни и те же задачи, а скорость работы и функционал богат. В качестве фреймворка для данной системы рекомендаций использовался FastAPI, написанный на языке

программирования Python [20]. Он служит для создания лаконичных и довольно быстрых HTTP API-серверов со встроенными валидацией, сериализацией и асинхронностью. Выбор данного фреймворка обоснован не только тем, что он имеет весь необходимый функционал и поддержку разработчиков, но и главным образом тем, что он используется для общей системы «Цифровой след студента». То есть он нужен для интеграции с общим проектом. Кроме того, у FastAPI не высокий порог вхождения при изучении, что способствует удобному применению в такой модульной системе, как «Цифровой след студента», которая разрабатывается разными людьми и им требуется обучение.

1.5 ПОДХОДЫ К ВИЗУАЛИЗАЦИИ ДАННЫХ

Для удобной работы с данными необходимо представить данные наглядно. Визуальное представление данных может быть красивым, элегантным и информативным. Существует множество способов визуализировать информацию. Методы визуализации делят на несколько типов [21]:

Стандартные 2D/3D-образы. Наиболее простой способ представить - построить гистограммы, линейные графики и т.п. Он позволяет увидеть простые зависимости между разными признаками, проверить гипотезы. Минусом же являет его низкая информативность в случае большого количества информации.

Геометрические преобразования. Таковой является, например, диаграмма разброса данных. Геометрические преобразования позволяют осуществить трансформацию многомерных наборов данных для отображения их геометрических в различных пространствах - в декартовом и в не декартовом.

Отображение иконок. Основной смысл данного метода – отобразить значения многомерных данных в различные реальные образы - человеческие лица, стрелки, звезды и т.д. Для атрибутов элементов данных создаются свойства образов. Такие образы можно группировать для целостного анализа данных. Результирующая

визуализация представляет собой шаблон текстур, которые имеют различия, соответствующие характеристикам данных.

Методы, ориентированные на пиксели — рекурсивные шаблоны и т.п. Главным в этом подходе является отображение каждого измерения значения в цветной пиксель, а также проводится группировка по принадлежности к измерению. Так как один пиксель используется для отображения одного значения, то, следовательно, данный метод позволяет визуализировать большое количество данных.

Иерархические образы или диаграммы связей. Диаграмма связей реализуется в виде древовидной схемы, на которой изображены слова, идеи, задачи или другие понятия, связанные ветвями, отходящими от центрального понятия или идеи, отправной точкой или точкой приложения которых является центральный объект. Их удобно использовать для отображения данных, имеющих иерархию. Для нашей работы, например иерархию представляет Учебный план и дисциплины – дисциплины являются частью учебного плана, а также каждая дисциплина относится к одной или нескольким областям знаний. На рисунке 1 представлен пример визуализации данным методом.

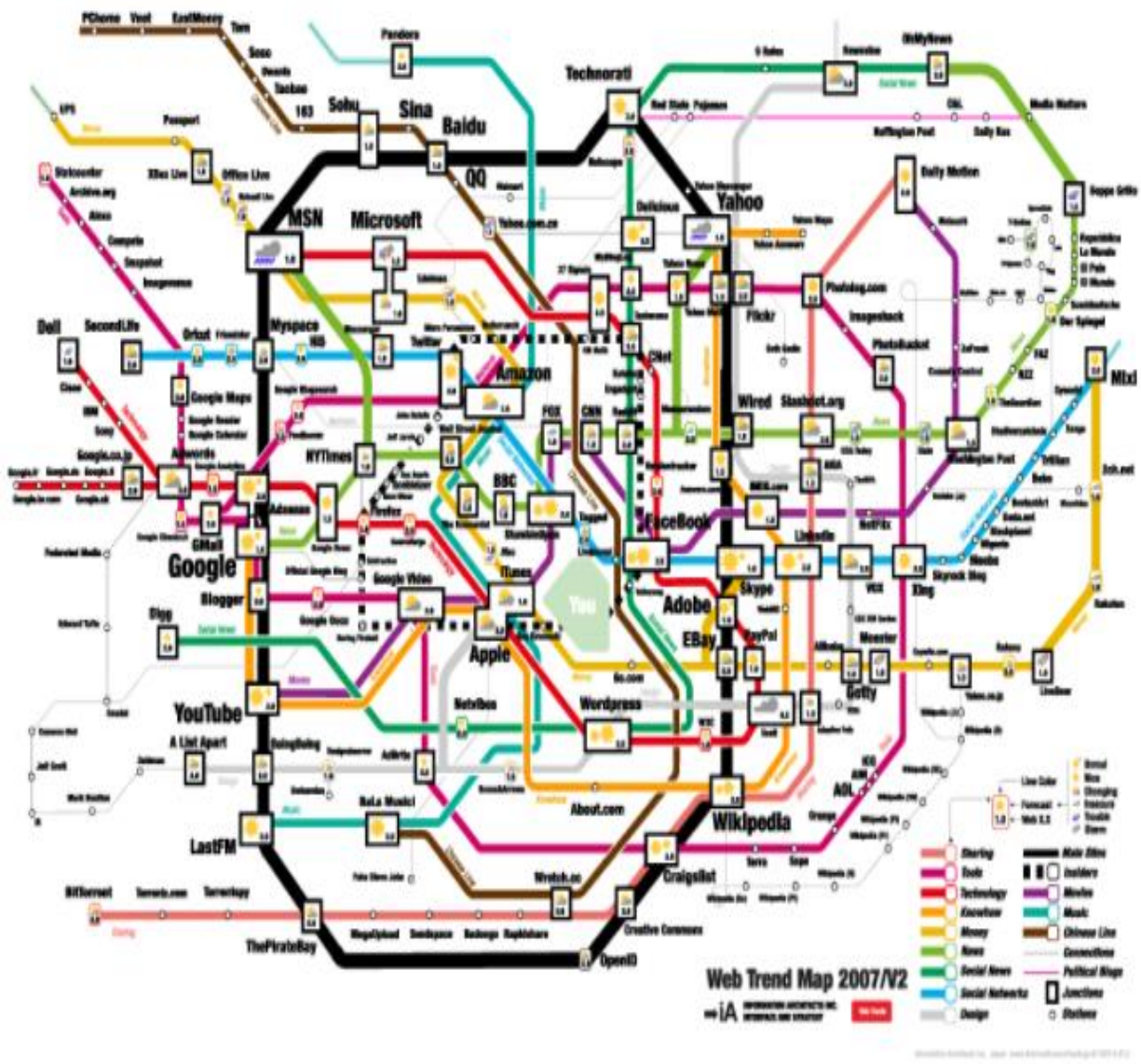

Рис. 1. Пример визуализации с помощью диаграммы связей. Карта наиболее популярных веб-сайтов в сети Интернет [22]

Treemapping – один из относительно новых способов визуализации информации при помощи вложенных прямоугольников, меняющих соотношение сторон в зависимости от исходных данных. Преимуществом является использование пространства при построении – оно на столько эффективно, что на одном экране могут отображаться тысячи данных одновременно и будут различимы. На

рисунке 2 представлен пример визуализации распределения голосов избирателей на президентских выборах 2012 года в США.

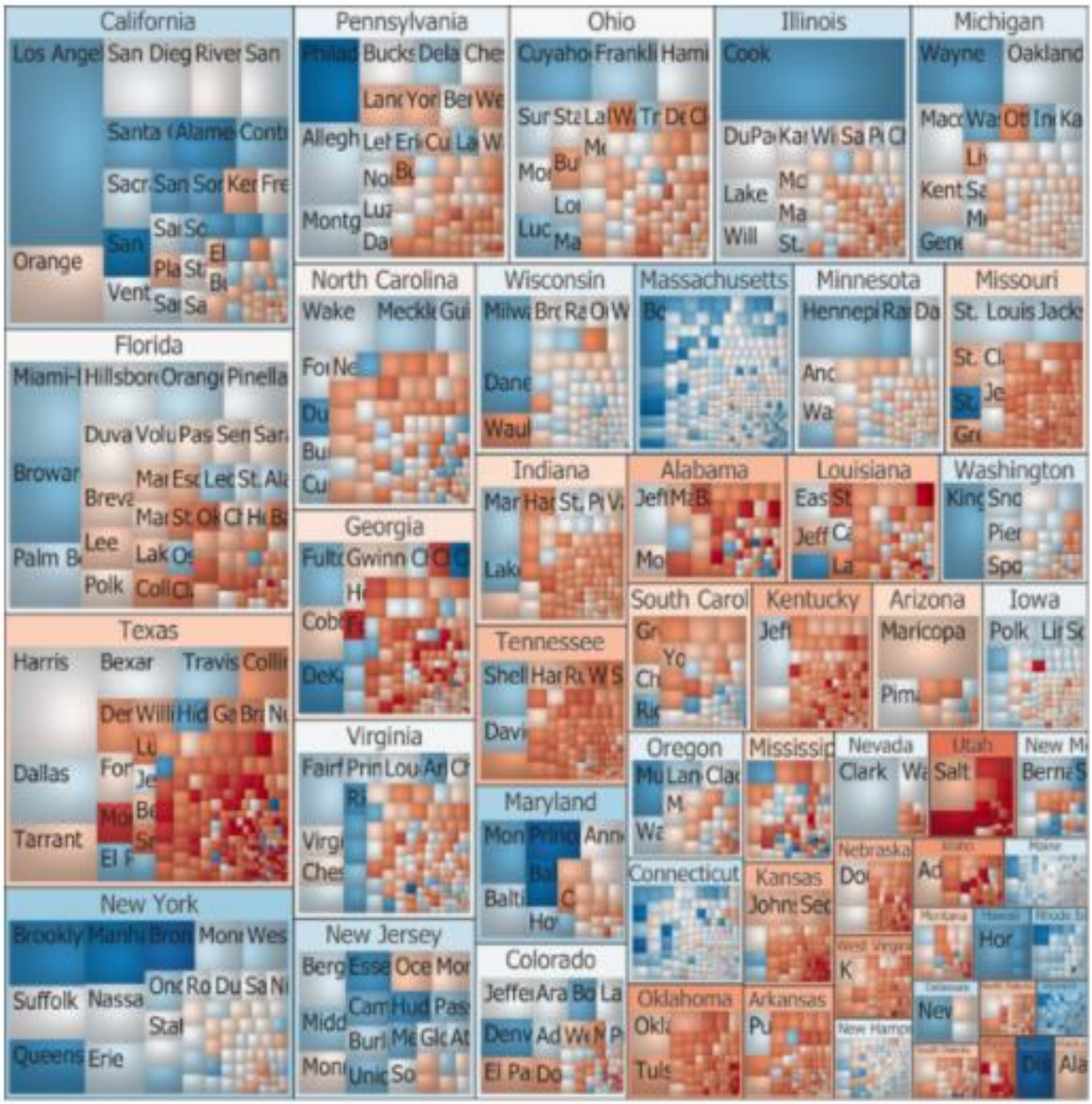

Рис. 2. Распределение голосов избирателей на президентских выборах 2012 года в США [23]

Все эти подходы очень популярны и активно используются аналитиками. Были также изучены подходы к визуализации данных по учебным планам и дисциплинам.

Так в работе Takada, S., Cuadros-Vargas, E., Impagliazzo, J [24] представлен способ визуализации компьютерных дисциплин с целью предоставить

возможность людям задавать различные вопросы относительно изучаемых программ интерактивно с помощью набора параметров визуализации. Авторы предлагают показывать различия учебной программы на основе компетенций. Для этого необходимо рассмотреть следующие параметры учебной программы:

Знание - владение основными понятиями, содержанием и применением обучения к новым ситуациям. Список тем часто представляет собой знания, которые охватывает компьютерный курс.

Навыки – они относятся к способностям и стратегиям, которые развиваются с течением времени, через практику и взаимодействие с другими. Она включает в себя способность производить результаты. Примерами развития навыков могут быть задания на основе проблем и лабораторные работы.

Диспозиции - включают в себя социально-эмоциональные навыки, поведение и установки, которые характеризуют склонность к выполнению задач и чувствительность к знанию того, когда и как заниматься этими задачами, например, уверенность в работе со сложностями, ответственность.

Именно эти три показателя формируют компетенции выпускника. Такой подход, в том числе, позволяет понять, что в учебной программе необходимо согласовать с требованиями работодателей в соответствующей индустрии. Интерфейс средства для абитуриентов предполагает выбор категорий из области знаний компьютерных наук, представленных к выбору, а также набор диспозиции из списка. Данные наборы диспозиций и категорий были выявлены в ходе различных опросов предполагаемых пользователей. В итоге, абитуриенту будет представлена таблица из 3 направлений обучения, для которых показаны наборы из шести параметров визуализации в соответствии со сделанным выбором.

Данный подход интересен тем, что проводился опрос аудитории на знание области компьютерных технологий, что позволило не только разбить курсы по категориям из области компьютерных наук, но и понять, какими знаниями обладают абитуриенты на момент поступления в среднем. Это помогает определить наиболее точные параметры визуализации учебных программ.

Кроме того, в работе представлен обзор других средств для визуализации учебных программ: это и системы визуализации на основе графов, и графики. Но все они направлены по большей части на взаимодействие с уже действующими студентами для помощи в выборе курсов в ходе обучения либо же на работу с администрацией университета для корректирования учебных программ.

1.6 ОБЗОР ИНСТРУМЕНТОВ ДЛЯ ВИЗУАЛИЗАЦИИ ДАННЫХ

Для визуализации данных есть много специальных инструментов. Инструменты могут быть как простыми, которые требуют лишь загрузку данных и настройку отображения, так и сложными — таким инструменты требуют настройки, предварительного анализа данных или знаний какого-либо языка программирования. Выбор средства визуализации зависит от поставленной задачи, а также от навыков, которыми владеют участники проекта. Для данной работы были проанализированы средства визуализации для языков программирования Python и JavaScript, а также те, которые не требую знаний языков программирования. Это обосновано тем, что «back-end» сторона разрабатывалась на языке Python, «front-end» сторона на языке JavaScript, а общий анализ данных и визуализацию для целей предварительного анализа перед предобработкой текста можно проводить с помощью любого средства, подходящего под задачу.

Первый инструмент – это Plotly [25]. Он позволяет быстро и удобно строить различные графики. В нем доступны гистограммы, диаграммы, графики в 3D и в 2D. Доступен для различных языков программирования, в том числе для Python и JavaScript. Есть различные варианты для импорта данных, построение графиков онлайн. Данный инструмент прост в освоение и хорошо подходит для обучения студентов.

Следующий инструмент DataHero [26]. Он удобен тем, что не требует дополнительных знаний языков программирования. Он также имеет функции построения различных графиков и в него можно интегрировать облачные сервисы, что удобно, когда информация располагается в разных местах.

Также для визуализации используется бесплатная библиотека Chart.js для языка JavaScript [27]. Он подойдет для небольших проектов. Для построения диаграмм программа использует HTML5 Canvas - элемента HTML5, предназначенного для создания растрового двухмерного изображения при помощи скриптов, и создает быстро реагирующий на изменения простой дизайн. В нашем проекте данный инструмент также задействован.

Список инструментов можно продолжать. Он постоянно пополняется и выбор инструмента зависит от задачи – на сколько быстро необходимо обрабатывать данные, нужно ли отображать в 3D пространстве или хватит 2D отображения и так далее.

ГЛАВА 2. ОПИСАНИЕ АЛГОРИТМОВ ОБРАБОТКИ ДАННЫХ

В работе использовались различные «сырые» текстовые данные: тексты РПД, тексты словарей, а также структурированная информация из базы данных: об учебных планах, направлениях обучения. В таблице 1 представлено описание исходных проанализированных данных.

Таблица 1

Описание используемых данных

Тип данных	Описание	Количество
Текст РПД	Хранятся в базе данных HBase. Текстовые данные описания РПД, тексты хранятся не предобработанными.	675 файлов
Информация о дисциплине	Хранится в отдельной таблице в базе данных PostgreSQL. Каждая строка, описывающая дисциплину, содержит следующие данные: Название дисциплины id	675 строк
Мета информация РПД	Хранится в отдельной таблице в базе данных PostgreSQL. Каждая строка, описывающая метаинформацию РПД содержит следующие данные:	675 строк

	направлении обучения id название дисциплины год обучения id доступа к тексту РПД в базе HBase (guid)	
Текст школьного словаря	Текстовые данные школьных словарей, тексты не предобработаны. После обработки информация сохраняется в HBase.	2 файла
Данные учебных планов	Хранятся в отдельной таблице в базе данных PostgreSQL. Каждая строка, описывающая учебный план содержит следующую информацию: направление обучения название учебного плана id год поступления	31 строка

	год окончания обучения.	
Данные направлений обучения	Хранятся в отдельной таблице в базе данных PostgreSQL Каждая строка, описывающая направление обучения содержит следующую информацию: название направления id форма обучения – очная/заочная.	20 строк

Был проведен также анализ количесвта ключевых слов в текстах РПД. В результате анализа оказалось, что среднее количество разных ключевых слов в каждом из текстов РПД равно 57. Кроме того, среднее количество ключевых слов в общем равно 155050.

2.1 ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

Тексты, которые необходимо было анализировать не предобработаны. Поэтому, для работы с текстами необходимо было провести предварительную подготовку текстов. Так как многие слова в тексте не несут смысловой нагрузки (это так называемые «служебные слова»), то необходимо было исключить их из общего подсчета слов. Исключение слов производилось на основе стандартного набора слов стоп-слов, предлагаемого в библиотеке nltk (описана в главе 1, пункт 1.4.1). Таким образом, ниже описана работа, проведенная с текстами.

1. Для всех текстов словарей производятся следующие действия:
   1.1. Приведение всех букв к нижнему регистру
   1.2. Очищение текста от запятых

1.3. Вычленение слов, определение которых дано в словаре

1.4. Приведение слов к нормальной форме

2. Для всех текстов РПД производятся следующие действия:

2.1. Приведение всех букв к нижнему регистру

2.2. Очищение текста от запятых

2.3. Токенизация текста

2.4. Приведение слов к нормальной форме

В результате предобработки данных в базу данных HBase сохраняются следующие данные:

Для каждого текста РПД - все слова, после предобработки в виде текстового файла. Порядок слов соответствует порядку слов в исходных текстах.

Для каждого текста РПД – общее количество слов после предобработки

Для каждого текста РПД – информация о словах в виде json файла, в котором содержится список структур со следующими полями:

Ключевое слово в нижнем регистре и нормальной форме

Число – сколько раз ключевое слово встретилось в тексте

Для каждого словаря – информация о словах в виде json файла, в котором содержится список структур со следующими полями:

Ключевое слово в нижнем регистре и нормальной форме

Число – сколько раз встретилось слово во всех текстах РПД

Пример json файла для словарей и текстов РПД представлен на рисунке 3.

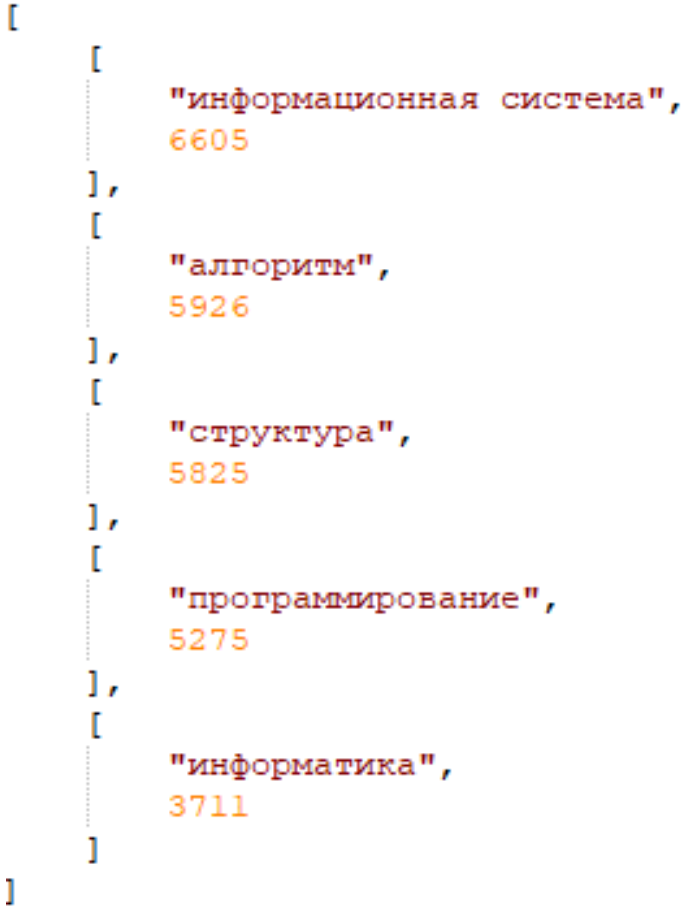

Рис. 3. Пример файла с данными о словах

При появлении новой РПД необходимо обновлять информацию для каждого сохраненного словаря. Для этого находится пересечение множества ключевых слов новой РПД и множества ключевых слов словарей. Для слов из пересечения данных множеств увеличивается количество раз, встретившихся во всех текстах РПД. Кроме того, для новой РПД будет найден и сохранен список ключевых слов, соответствующих добавленному тексту. Данные обновления необходимо проводить при добавлении новой РПД в базу данных.

2.2 ОПИСАНИЕ АЛГОРИТМА ПОИСКА НАИБОЛЕЕ ПОДХОДЯЩИХ ОБРАЗОВАТЕЛЬНЫХ ПРОГРАММ ДЛЯ АБИТУРИЕНТА

Описание алгоритма реализовано с помощью псевдокода. В таблице 2 представлены переменные, которые используются в алгоритме поиска наиболее подходящих образовательных программ.

Таблица 2

Переменные алгоритма поиска наиболее подходящих образовательных программ

Идентификатор переменной	Описание
	Ключевые слова абитуриента, s Z, s [0, S)
	Учебные планы, i Z, i [0, I). Учебный план содержит: name – название year – год dir_id – id направления description – описание
	Дисциплины для , k Z, k [0, K). Дисциплина содержит: subj_id – id дисциплины subj_name – название rpd_words – словарь пар { – ключевое слово, – частота повторений в тексте РПД }, j Z, j [0, J) all_word_count – общее количество слов в тексте РПД
valid_rpd_count	Количество дисциплин с валидными данным в базе данных
op_words_count	Общее количество слов в образовательной программе
op_rat	Сколько раз слова из input_words встретились в текстах РПД

op_ball	Балл схожести образовательной программы
programs	Итоговый список структур «Образовательная программа» (Рисунок 5)
rpd_words_count	Словарь пар {ключевое слово, частота повторений в тексте РПД}
rpd_rats	Список информации о дисциплинах. Содержит: subj_id из subj_name из rpd_words_count

В таблице 3 представлены ключевые слова, используемые при написании псевдокода. Ключевые слова соответствуют рекомендациям по написанию псевдокода из статьи Д. Ллойда [28].

Таблица 3

Ключевые слова, используемые в псевдокоде

Ключевое слово	Описание
ИЗВЛЕЧЬ	Запросить из базы данных информацию
СОЗДАТЬ	Проинициализировать переменную начальными данными
ДЛЯ	Цикл со счетчиком
ЕСЛИ	Условный оператор
ПЕРЕЙТИ	Перейти к следующей итерации цикла
УВЕЛИЧИТЬ	Увеличить количество
ДОБАВИТЬ	Сохранить значение в список или словарь

ОТСОРТИРОВАТЬ	Сортировка значений по какому-либо параметру
ВЕРНУТЬ	Вернуть результат работы алгоритма

Далее приведен алгоритм расчета рекомендаций.

ИЗВЛЕЧЬ учебные планы , i Z, i [0, I) из базы данных

СОЗДАТЬ пустой выходной список рекомендованных программ programs

ДЛЯ , i Z, i [0, I):

valid_rpd_count = 0

op_rat = 0

op_words_count = 0

СОЗДАТЬ пустой список для выходной информации о РПД rpd_rats

ИЗВЛЕЧЬ информацию о дисциплинах , k Z, k [0, K) для

ДЛЯ , k Z, k [0, K):

ЕСЛИ 0 == из

ПЕРЕЙТИ к следующей РПД

СОЗДАТЬ пустой список rpd_words_count пар {слово, частота его вхождения в текущий текст РПД }

УВЕЛИЧИТЬ op_words_count на из (счетчик количества слов)

valid_rpd_count += 1

ДЛЯ , s Z, s [0, S):

ДЛЯ , , j Z, j [0, J):

ЕСЛИ

УВЕЛИЧИТЬ op_rat на (счетчик частоты встречаемости слов в текстах РПД)

ДОБАВИТЬ в rpd_words_count пару { , }

ДОБАВИТЬ в rpd_rats

{ из , из , rpd_words_count}

ЕСЛИ valid_rpd_count == 0

ПЕРЕЙТИ к следующей РПД

op_ball = ————————

ДОБАВИТЬ в programs информацию о текущем плане

{ из , op_ball, из , rpds_rats }

ОТСОРТИРОВАТЬ programs по убыванию op_ball

ВЕРНУТЬ первые L элементов programs

ГЛАВА 3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ ФОРМИРОВАНИЯ РЕКОМЕНДАЦИЙ

3.1 ОБЩЕЕ ОПИСАНИЕ

Система «Цифровой след студента», представленная в статье [29], состоит из различных модулей, которые взаимодействуют на основе общей модели и схемы данных. Модуль – это клиент-серверное приложение, состоящее из двух частей: «front-end» и «back-end». Сторона «front-end» представляет собой визуальную составляющую, с которой взаимодействует пользователь. Эта часть не участвует в расчетах и все данные, которые ввел пользователь, передаются на сторону «back-end» по заданному протоколу. Сторона «back-end» полностью отвечает за расчеты и взаимодействие с данными из баз данных. Разработанная в рамках данной работы система рекомендаций представляет собой один из модулей информационной системы «Цифровой след студента».

3.2 ОСОБЕННОСТИ ДЕЙСТВИЙ ПОЛЬЗОВАТЕЛЯ

В основе разработки системы лежит удобство для пользователя и доступность работы с ней. В связи с этим проектирование системы начиналось с продумывания интерфейса взаимодействия абитуриента с системой рекомендации. Представим взаимодействие пользователя и рекомендательной системы. Как может абитуриент правильно сформулировать свой запрос при выборе образовательной программы? На какие ключевые критерии стоит обратить внимание? Он может сказать, какие области знаний в школе его интересовали и/или давались легко, а в каких-то он предпочитает более конкретные направления. Пользователю не придется составлять законченный вопрос в формате предложения, для удобного формирования запроса к сервису. Для этого потребуется только выбрать из облаков слов наиболее близкие слова, описывающие его предпочтения в каждой из областей знаний.

В рамках данной работы были выбраны области знаний, соответствующие основным направлениям института математики и компьютерных наук – это области математики и область информатики. На рисунке 4 представлен пример

облака слов, в котором пользователь может выбрать одно или несколько ключевых слов, по которым система будет подбирать наиболее подходящее

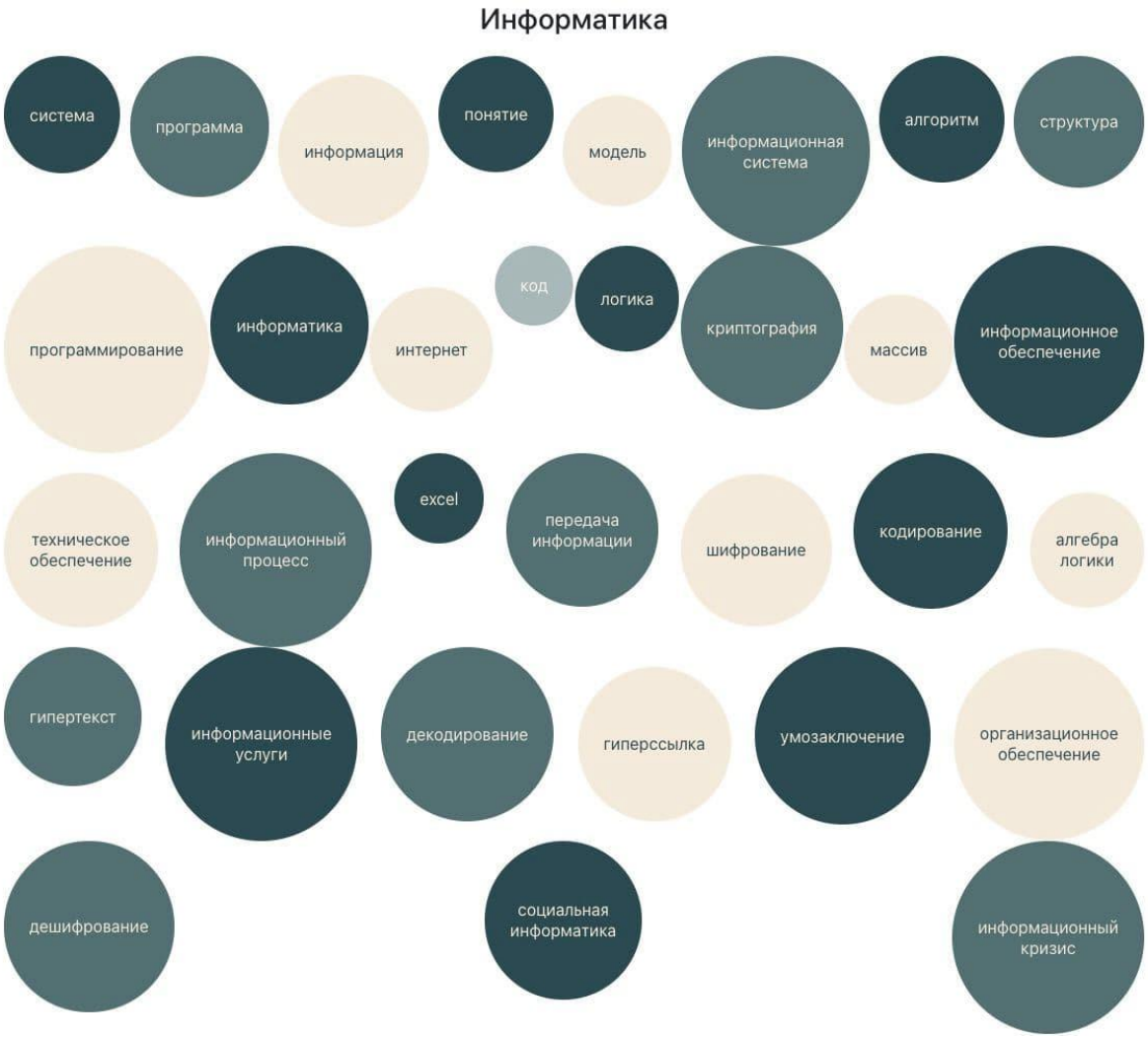

Рис. 4. Облако слов для области знаний «Информатика»

направление. Облако слов для области знаний «Математика» выглядит аналогично.

После выбора слов пользователь выбирает, сколько образовательных программ он хочет получить в результате расчетов системы и нажимает на кнопку «Получить рекомендации». Система начинает расчеты на основе алгоритмов, представленных выше (глава 2, пункт 2.2).

В результате расчетов будет получена информация по образовательным программам. Образовательные программы будут отсортированы по рейтингу схожести с выбранными пользователем словами. Длина списка рекомендованных программ зависит от выбора абитуриента. Если

рекомендованных программ меньше, чем заданное количество, то выводится максимально возможное количество образовательных программ.

3.3 АРХИТЕКТУРА СИСТЕМЫ

Архитектура программного продукта представлена на рисунке 5. Система состоит из трех частей – это хранилище данных, сервисы «front-end» и сервисы «back-end».

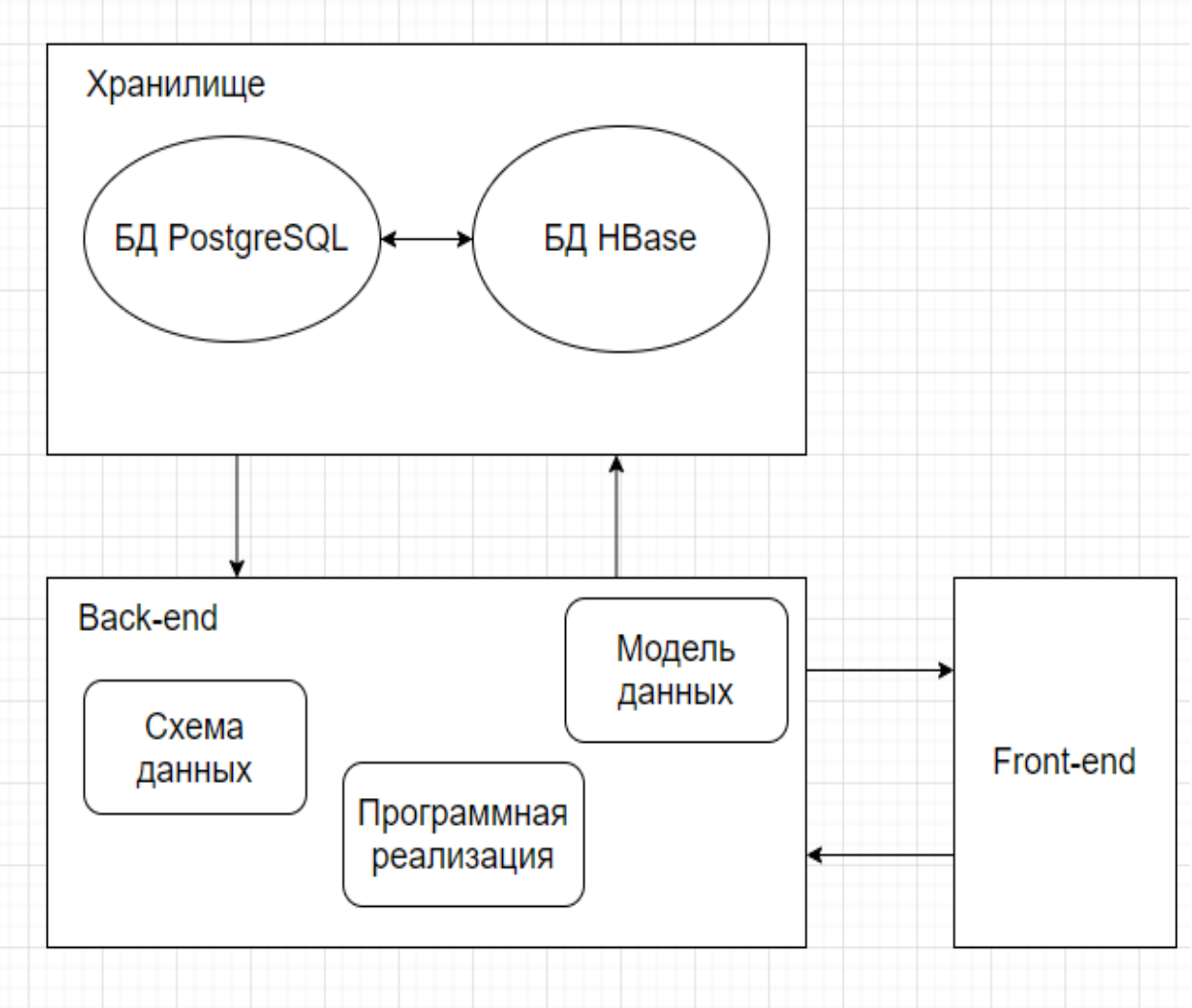

Рис. 5. Архитектура программного продукта

Хранилище включает в себя базу данных PostgreSQL и HBase. Хранилище является общим на всю систему «Цифровой след студента». Таким образом, хранилище представляет собой объединение двух баз данных, в которых хранится не только структурированная и обработанная информация, но и сырые данные. На рисунке 6 представлена схема хранилища данных в системе «Цифровой след студента».

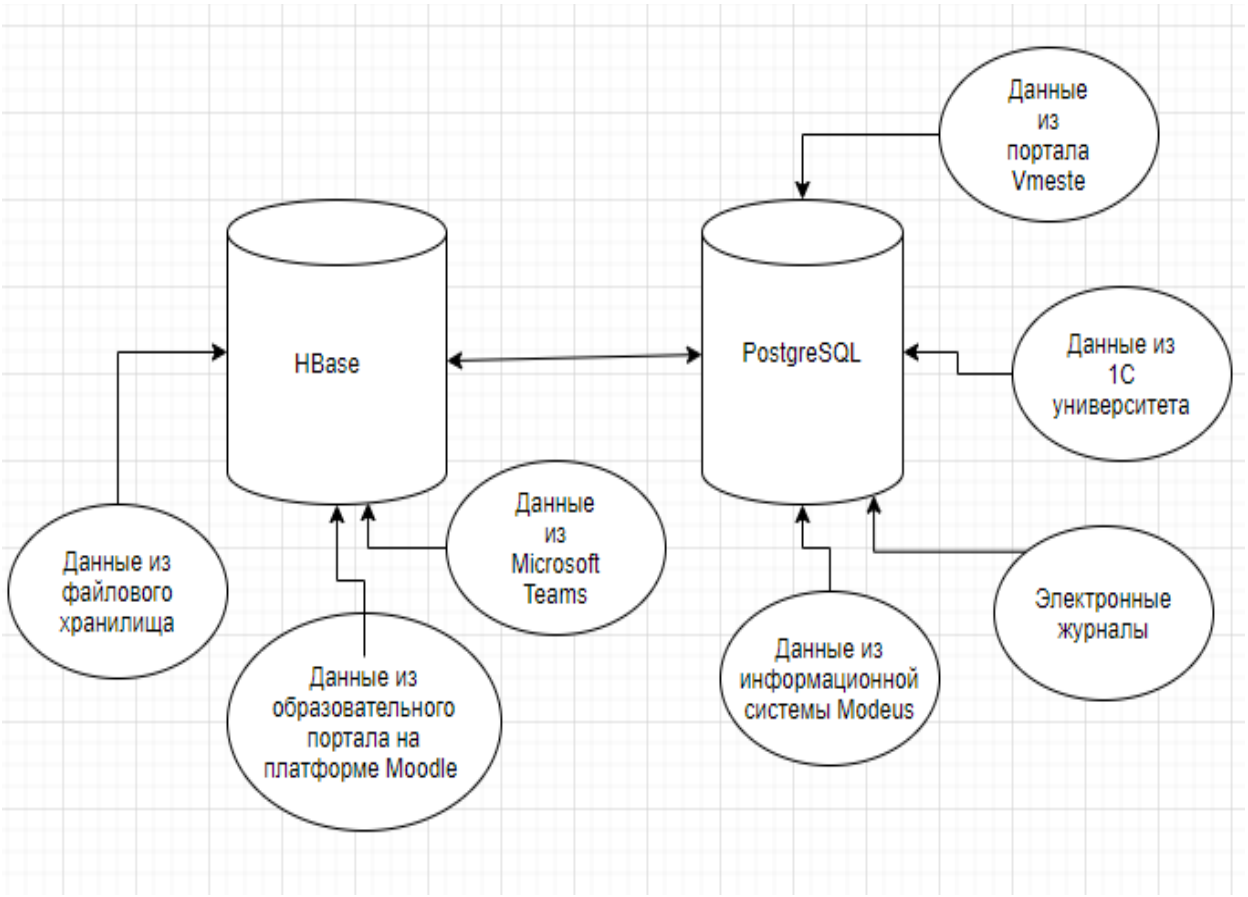

Рис. 6. Схема хранилища данных в системе «Цифровой след студента»

Запросы к базам данных осуществляются через сторону «back-end». В рамках данной работы разрабатывалась сторона «back-end». Опишем её более подробно.

Модель данных определяет классы доступа к данным из базы данных PostrgreSQL с помощью SQLAlchemy. Для данного модуля были разработаны классы для метаданных РПД, данных о дисциплинах, направлениях обучения и учебных планах. Добавлены основные необходимые поля для работы модуля. Модель может быть расширена для переиспользования в случае необходимости. В классах модели хранятся идентификаторы доступа к некоторым данным из базы данных HBase. Таким образом, с помощью данной модели можно получать информацию и из базы данных HBase. Модель данных представлена на рисунке 7. Код, описывающий модель данных приведен в приложении 1.

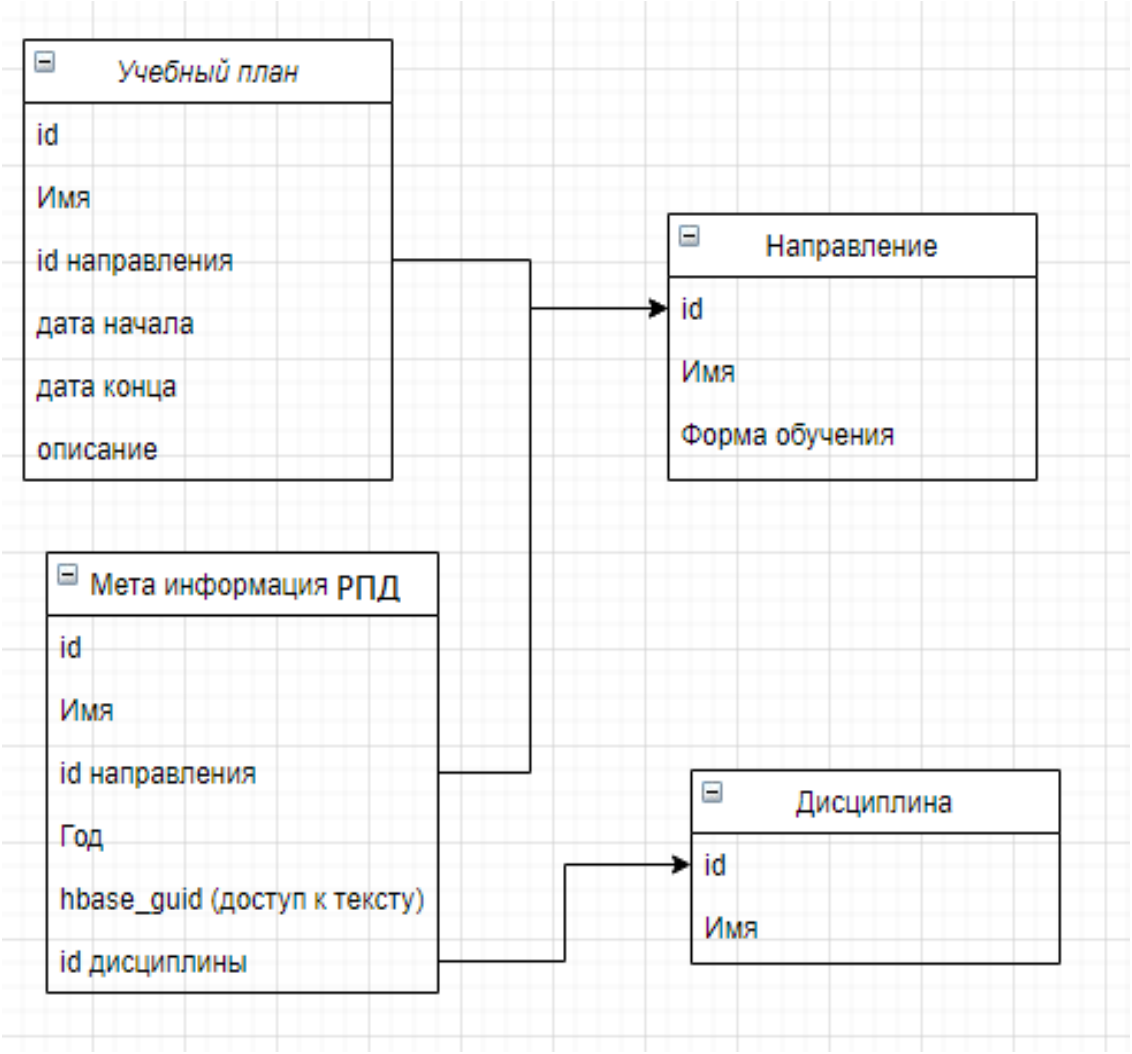

Рис. 7. Модель данных

Модель данных системы рекомендации является частью общей модели системы «Цифровой след студента» и является её расширением.

Схема данных показывает классы, позволяющие взаимодействовать стороне «front-end» и «back-end». Для данного модуля были разработаны классы для получения стороной «front-end» данных об образовательных программах и дисциплинах. Схема данных представлена на рисунке 8. Данная схема позволяет взаимодействовать не только с расчетной инофрмацией, но и с информацией непосредственно из базы данных (таблица «Информация РПД» на рисунке 8). Данная схема также, как и модель данных, расширяема. Код, описывающий схему данных приведен в приложении 2.

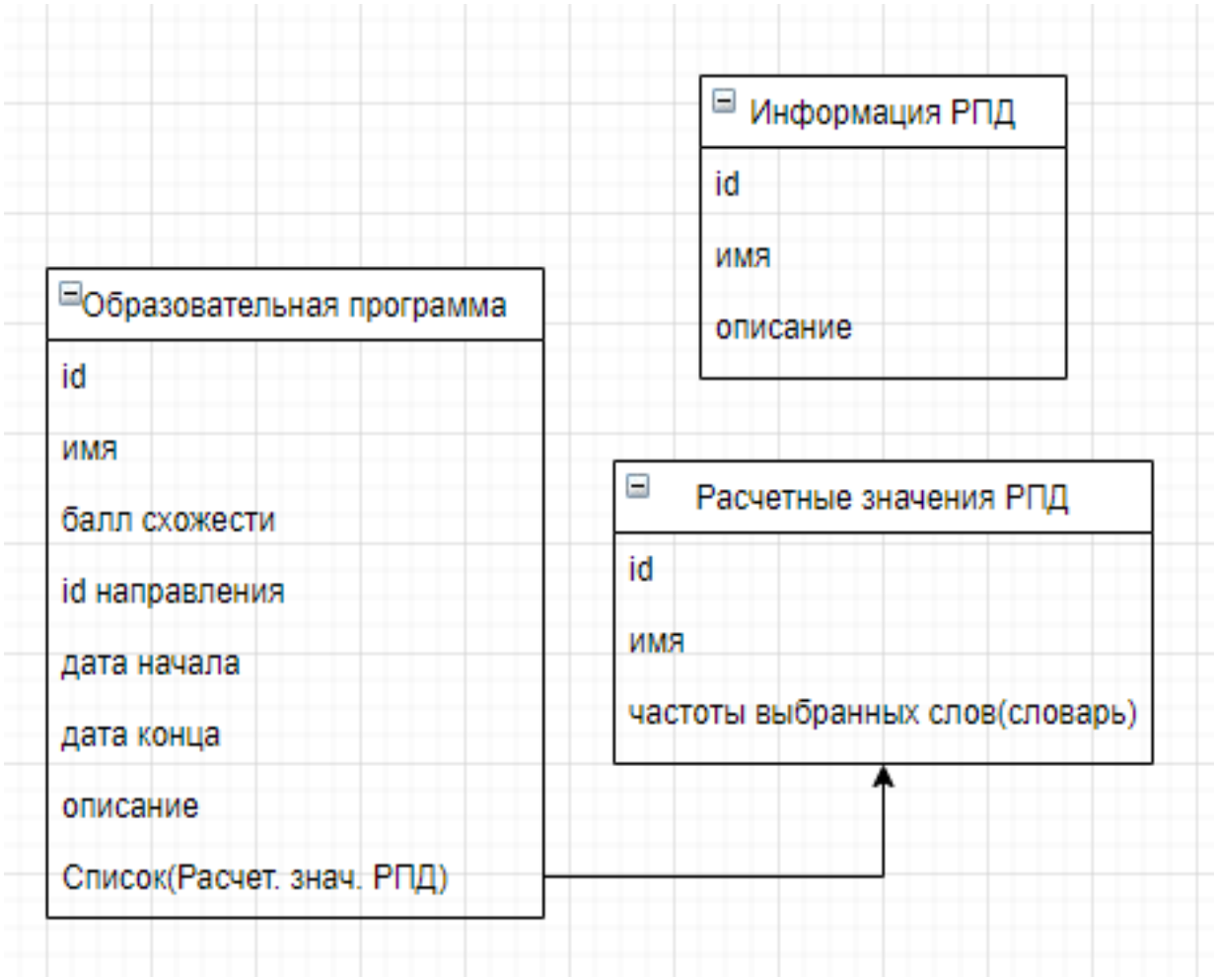

Рис. 8. Схема данных

Отдельно стоит отметить точки подключения стороны «front-end» к стороне «back-end» (Endpoint). Они определяют адрес доступа, а также форматы запросов от стороны «front-end» к стороне «back-end». Запросы представлены в приложении 3.

Алгоритмическая часть (программная реализация алгоритма) содержит функции, реализующие алгоритм поиска подходящих образовательных программ, а также запросы и формирование необходимых данных из баз данных для отображения на стороне «front -end» — это структуры с информацией об образовательных программах, а также о дисциплинах, входящих в рекомендованные образовательные программы. Полный код алгоритма и функций доступа к данным приведен в приложении 4.

На рисунке 9 представлена схема интеграции модели и схемы данных в общую систему «Цифровой след студента». На схеме видно, что модель и схема

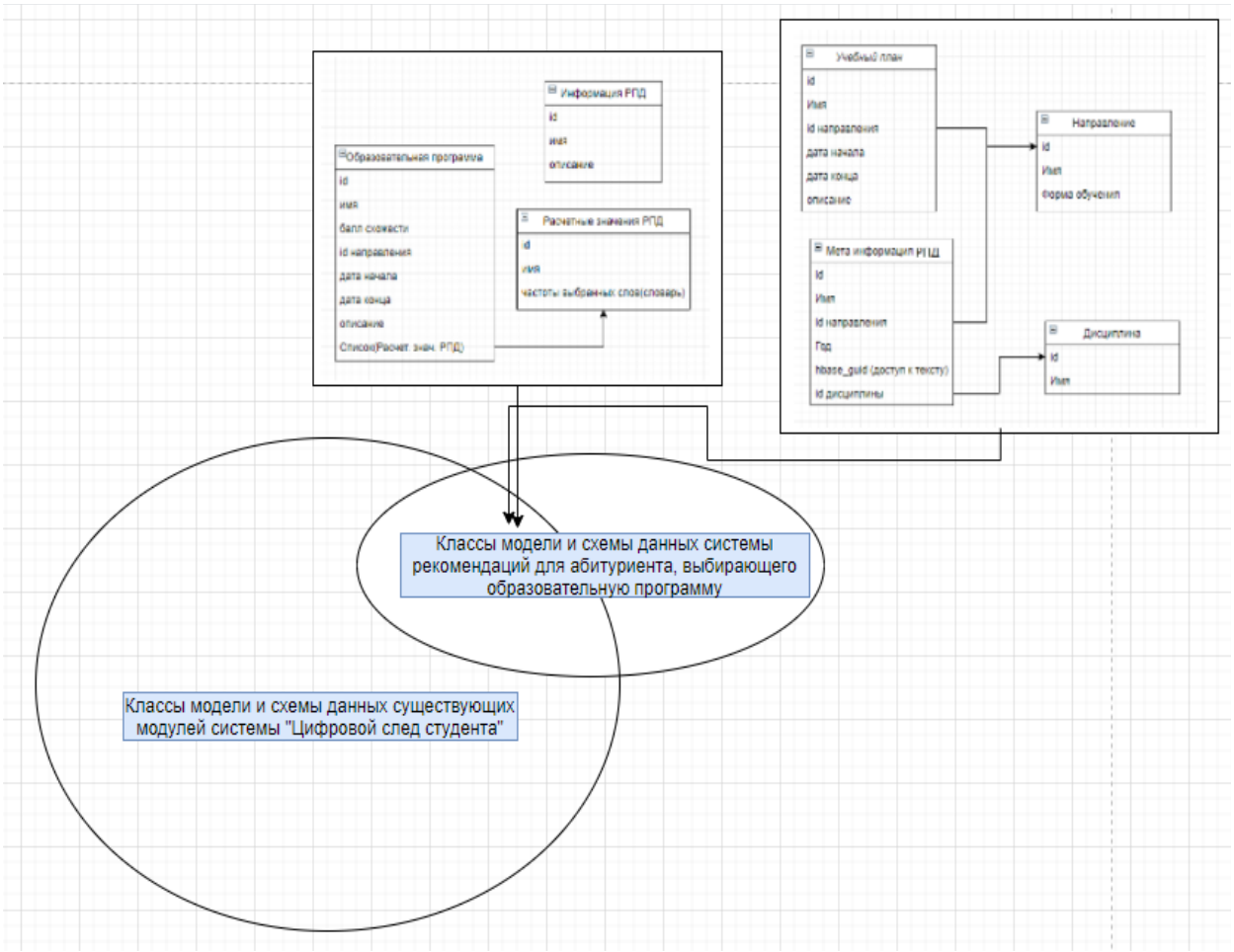

Рис. 9. Интеграция модели и схемы данных системы рекомендаций в общую систему «Цифровой след студента»

являются расширениями существующих модели и схемы общей системы «Цифровой след студента» и могут быть использованы для других модулей этой системы.

3.4 ОРГАНИЗАЦИЯ ОБЩЕНИЯ СТОРОН «BACK-END» И «FRONT-END»

Сторона «front-end» отправляет запросы к стороне «back-end». Запросы могут быть как формата «get» - то есть без аргументов, так и формата «post», то есть позволяющие добавить входные данные для стороны «back-end». Сторона «back-end» постоянно ожидает запросов и отвечает на них с помощью структур данных, в соответствии со схемой, описанной в пункте 3.3 данной главы (Рисунок 8). Запросы от стороны «front-end» следующие:

1. «Post» запрос для получения рекомендации направления обучения для абитуриента, с аргументом в виде списка строк, где строки – это ключевые слова, выбранные абитуриентом в облаке слов. В ответ на данный запрос сторона «back-end» выдает список структур «Образовательная программа» (Рисунок 6)
2. «Get» запрос для получения информации об изучаемых дисциплинах. В ответ на данный запрос сторона «back-end» выдает структуру «Информация РПД» (Рисунок 8)

3.5 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ РЕКОМЕНДАЦИЙ

Программную реализацию можно описать схематично, выделив основные блоки и связи между ними. На рисунке 10 представлена схема программной реализации системы рекомендаций.

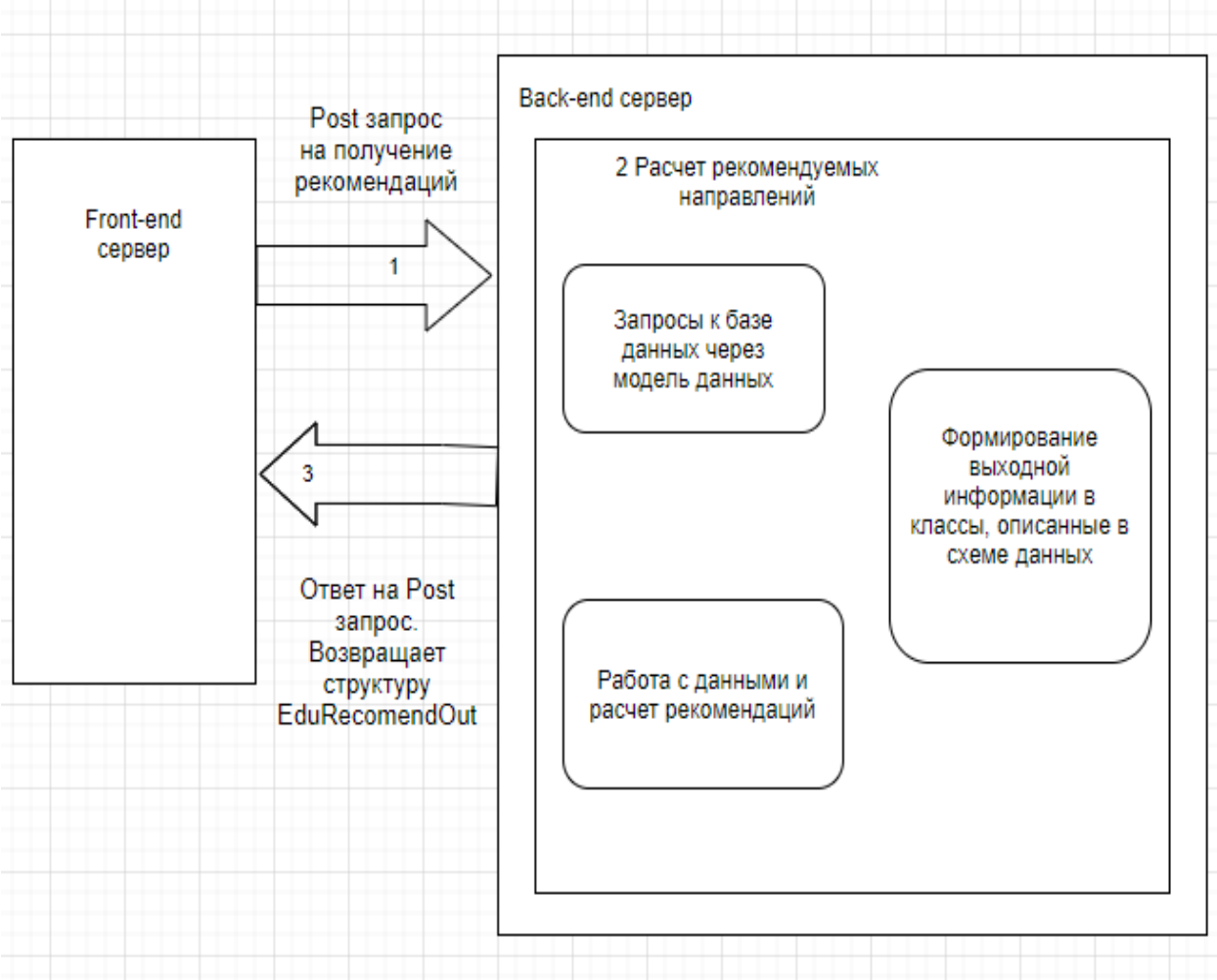

Рис. 10. Схема программной реализации рекомендательной системы

Абитуриент работает с «front-end» стороной системы рекомендаций. После нажатия на кнопку «Получить рекомендации», сторона «front-end» отправляет запрос на «back-end» сервер (стрелка 1 на рисунке 10). Параметры запроса описаны в приложении 3, функция get_recommend_edus(db: Session, hbase: Connection, input_words: List[str], limit: int = 100, offset: int = 0). На вход поступает список ключевых слов, выбранных абитуриентом, и после применения алгоритма системы рекомендации (объект 2 на рисунке 10), «back-end» сервер отправляет ответ на запрос с информацией о рекомендуемых направлениях (стрелка 3 на рисунке 10). Представим подробнее взаимодействие программы с моделью и со схемой данных во время работы алгоритма.

Во время расчетов необходимо получать информацию об учебных планах и изучаемых дисциплинах. Для этого используются функции, в которых отправляются запросы к базе данных, функции описаны в приложении 4:

получение наиболее актуального учебного плана get_last_academic_plans(db: Session, hbase: Connection, limit: int = 100, offset: int = 0)

получение исходной информации о РПД определенного учебного плана get_rpds_by_edu(db: Session, hbase: Connection, dir_id: int, year_start: int, year_end: int, limit: int = 100, offset: int = 0).

получение информации о текстах РПД определенного учебного плана get_rpds_recommend_info(db: Session, hbase: Connection, dir_id: int, year_start: int, year_end: int, limit: int = 100, offset: int = 0)

Для доступа к базе данных эти функции используют модель данных, представленную в приложении 1. Основная функция, реализующая алгоритм также представлена в приложении 4: get_recommend_edus(db: Session, hbase: Connection, input_words: List[str], limit: int = 100, offset: int = 0). Кроме работы с информацией из базы данных, в ней также формируется выходная информация о рекомендуемых направлениях. Согласно схеме данных, представленной в приложении 2, функция возвращает информацию в виде класса EduRecomendOut.

3.6 ТЕСТИРОВАНИЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ

Алгоритм тестировался на данных об образовательных программах ИМиКН ТюмГУ. Желаемое количество рекомендованных программ задавалось равным трем. В таблице 4 приведены примеры рекомендаций, на основе списка ключевых слов, поданного на вход системе рекомендаций.

Таблица 4

Результаты расчетов в ходе тестирования алгоритма

Входные данные: список ключевых слов	**Выходные данные: список названий образовательных программ**
'кодирование', 'шифрование', 'криптография'	1. "Компьютерная безопасность" 2. "Информационная безопасность" 3. "Информационные системы и технологии"
'алгоритм', 'программирование', 'модель'	1. "Математическое обеспечение и администрирование информационных систем" 2. "Механика и математическое моделирование" 3. "Информационные системы и технологии"
'дифференцирование', 'аналитическая геометрия', 'модель', 'стереометрия'	1. "Механика и математическое моделирование" 2. "Математическое обеспечение и администрирование информационных систем" 3. "Информационные системы и технологии"

3.7 ПРИМЕР ВЗАИМОДЕЙСТВИЯ ПОЛЬЗОВАТЕЛЯ С СИСТЕМОЙ РЕКОМЕНДАЦИЙ

Представим пример работы с рекомендательной системой. На рисунке 11 представлен пример выбора ключевых слов абитуриентом.

С помощью облака слов визуализируются слова из соответствующей области знаний. Максимальное количество слов, которое видно в облаке слов, можно настроить в соответствующем поле. Представим пример выбора слова из области знаний «Информатика». Изначально слова в облаке отображаются в

зависимости от встречаемости их в текстах РПД. При вводе слова в строку поиска, производится фильтрация слов по началу ввода. Так, на рисунке 6 можно видеть, что при выборе буквы «п», в списке слов, предложенных на выбор абитуриенту, остаются лишь те слова, которые начинаются только на данную букву.

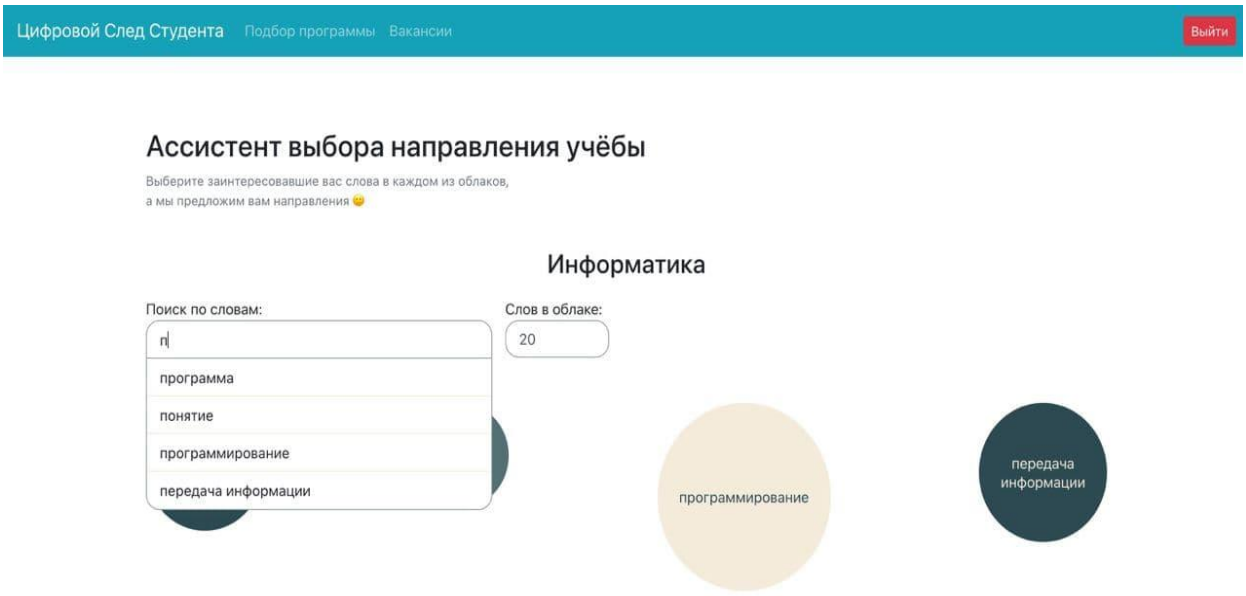

Рис. 11. Фильтрация ключевых слов при запросе

Кроме того, в облаке слов также остаются лишь те слова, которые соответствуют началу запроса. Выбирать слова можно как из списка, так и кликом на слово в облаке слов.

Выбранные слова отображаются рядом со строкой поиска и могут быть удалены из списка путем нажатия на крестик около нужного слова. На рисунке 12 показано, как выглядит конечный выбор слов пользователем: это слова «алгоритм», «программирование» и «модель». После выбора слов пользователь нажимает на кнопку «Получить рекомендации», расположенную в правом верхнем углу окна системы рекомендаций и система начинает рассчитывать наиболее подходящие направления обучения для данного запроса.

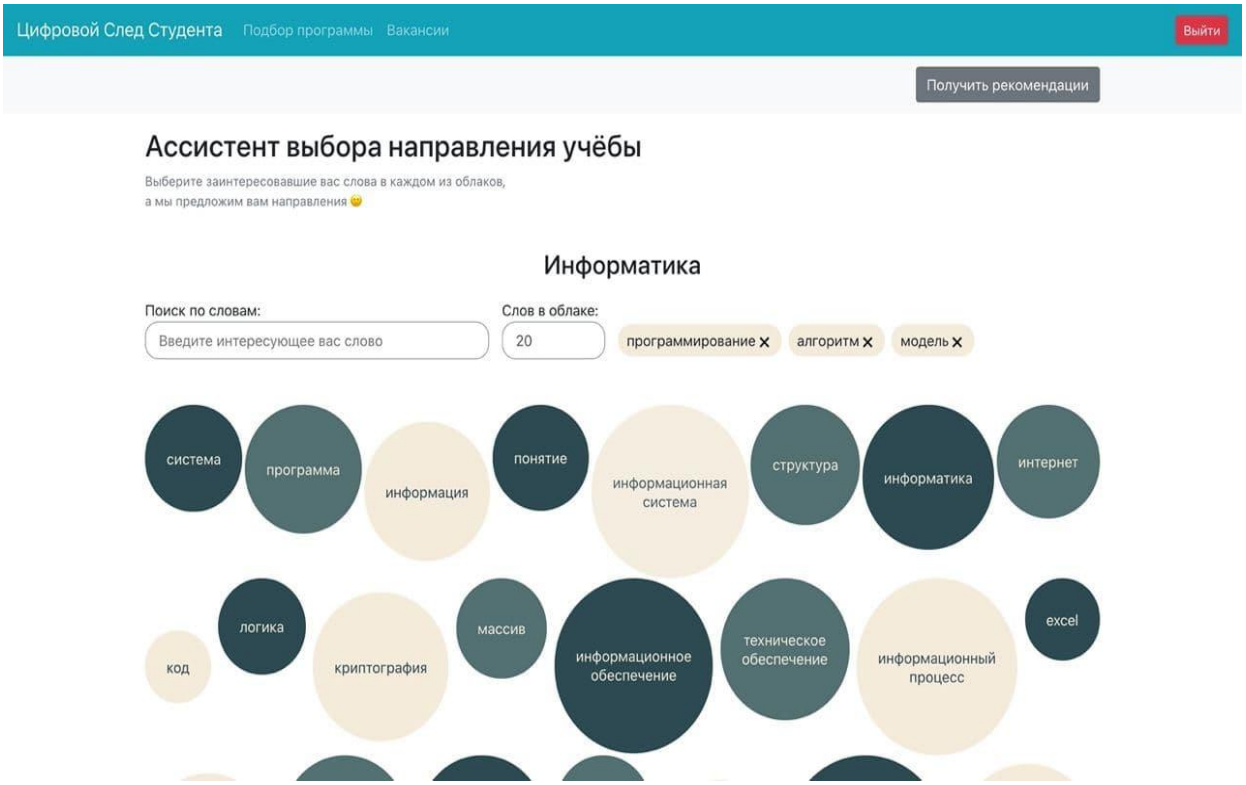

Рис. 12. Результат выбора слов абитуриентом

На рисунке 13 представлен результат, рассчитанный системой рекомендаций. Мы видим список рекомендованных образовательных программ на основе расчета. Список отсортирован по баллу схожести с запросом пользователя. Рядом отображается, на сколько процентов направление соответствует введенному запросу пользователя. Количество рекомендованных программ,

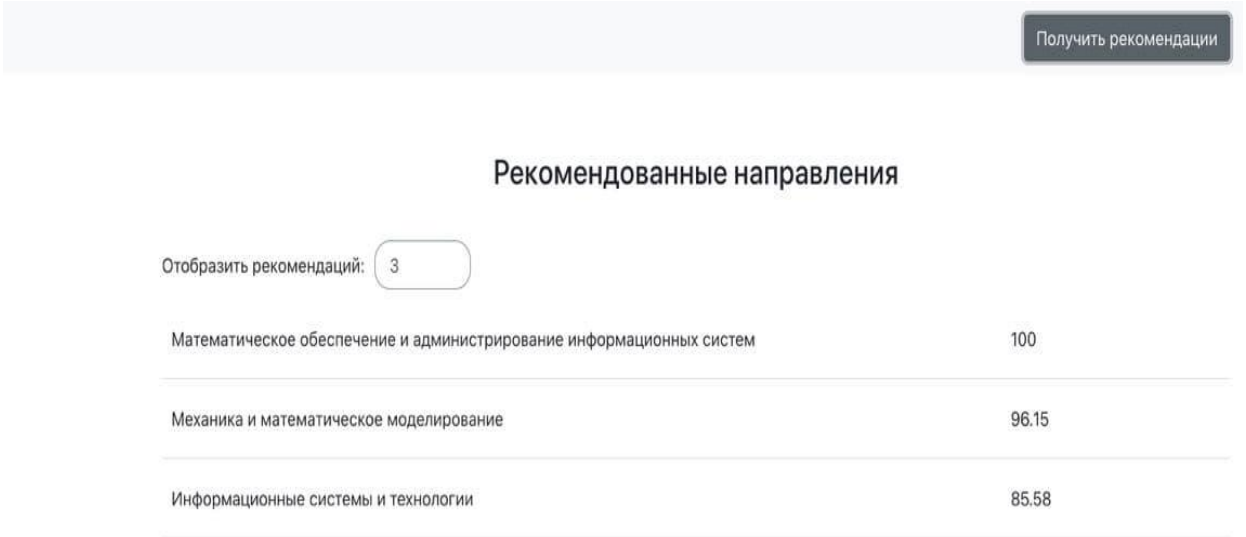

Рис. 13. Результат расчета системы рекомендаций

которое необходимо вывести пользователю, задается в соответствующем поле

перед список слов. В данном примере было указано найти 3 лучшие рекомендации – это значение задано изначально, при открытии страницы.

Для анализа ключевых слов, вошедших в каждое из рекомендованных направлений, реализовано отображение гистограмм частот встречаемости выбранных абитуриентом ключевых слов в дисциплинах рекомендованных направлений. При клике на рекомендованном направлении отображаются ключевые слова из списка абитуриента, которые входят в проанализированные тексты выбранного направления. При клике на определенном ключевом слове пользователь видит гистограмму распределения данного ключевого слова в текстах РПД из выбранного направления. Далее приведены примеры гистограмм для каждого из рекомендованных направлений в примере для слова «алгоритм». На рисунке 14 приведен пример для слова «алгоритм» и направления «Математическое обеспечение и администрирование информационных систем».

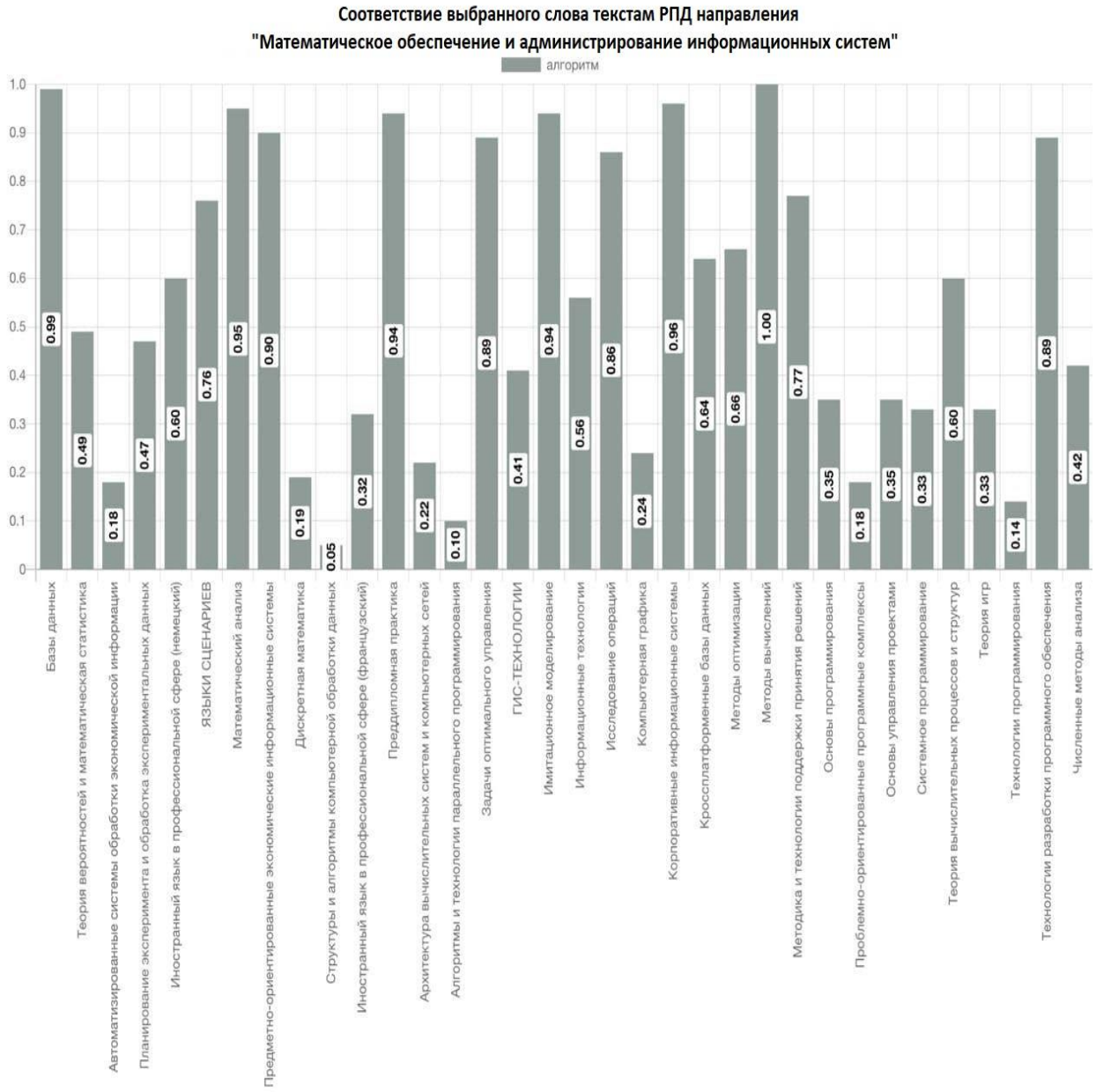

Рис. 14. Пример визуализации выбранного ключевого слова «алгоритм» в текстах РПД направления «Математическое обеспечение и администрирование информационных систем»

Представим результаты для этого же слова «алгоритм» для двух других направлений. На рисунке 15 приведен пример для слова «алгоритм» и направления «Механика и математическое моделирование».

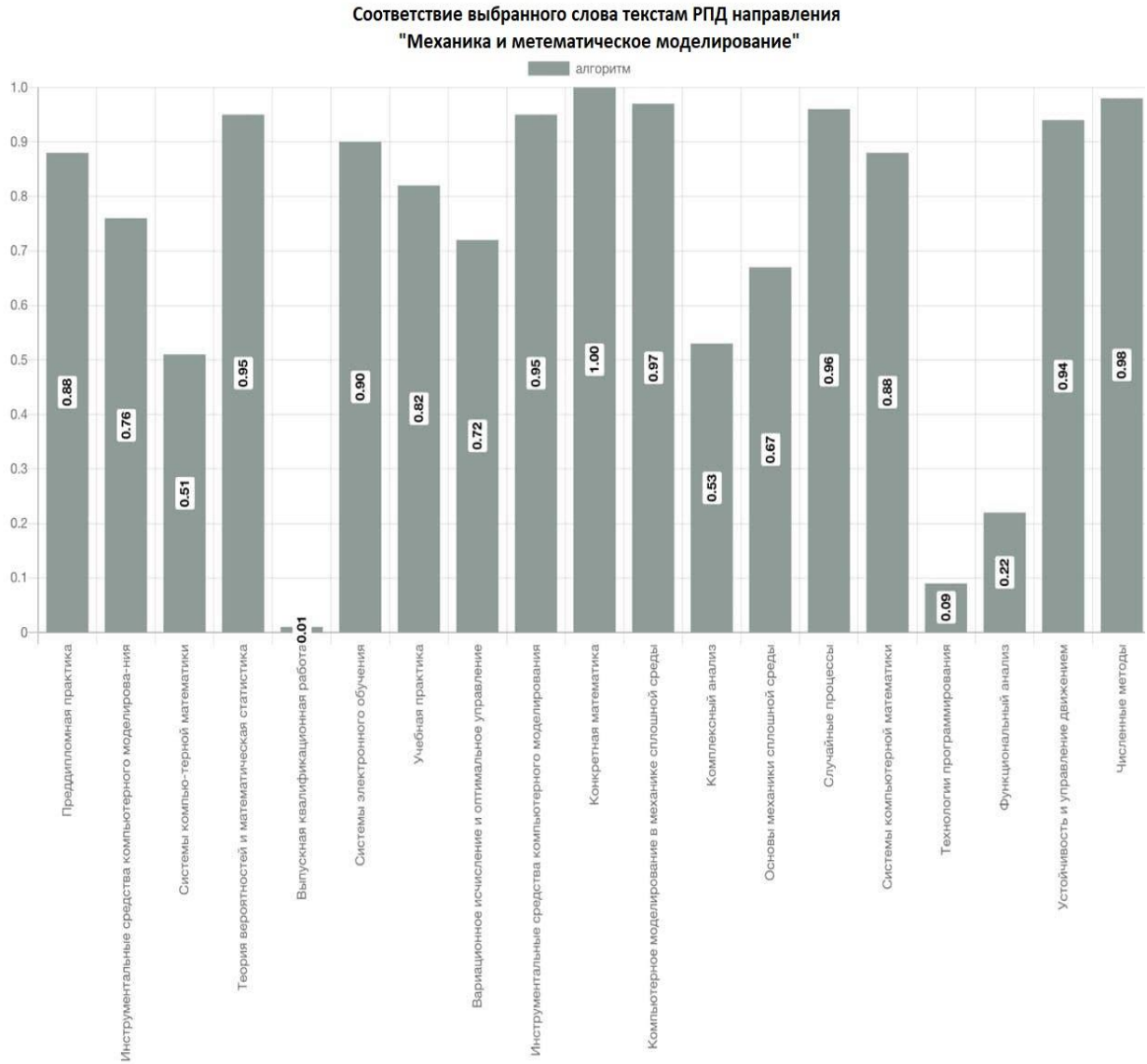

Рис. 15. Пример визуализации выбранного ключевого слова «алгоритм» в текстах РПД направления «Механика и математическое моделирование»

На рисунке 16 приведен пример для слова «алгоритм» и направления «Информационные системы и технологии».

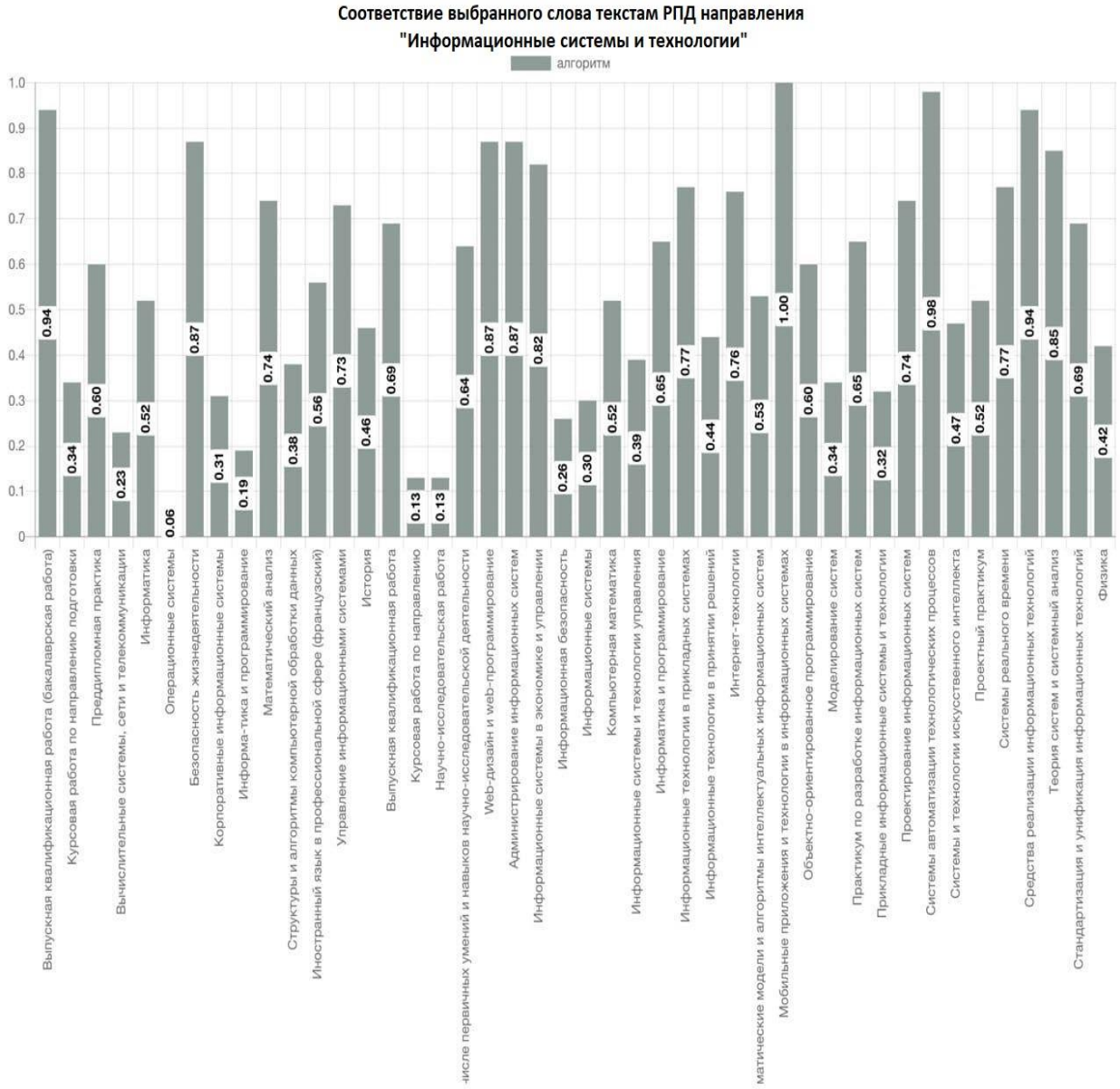

Рис. 16. Пример визуализации выбранного ключевого слова «алгоритм» в текстах РПД направления «Информационные системы и технологии»

ЗАКЛЮЧЕНИЕ

В диссертационной работе достигнута поставленная цель исследования – разработана рекомендательная система для абитуриентов, которые выбирают образовательную программу, на примере данных института математики и компьютерных наук (ИМиКН) Тюменского государственного университета.

При выполнении задач исследования были получены следующие основные результаты. Во-первых, были изучены существующие решения выявленной проблемы. Исследование показало, что проблема актуальна и выбранный подход еще не был использован при её решении. Во-вторых, были проанализированы данные об учебных планах и дисциплинах ИМиКН Тюменского государственного университета и на основе анализа была проведена обработка сырых данных для их дальнейшего использования в данной рекомендательной системе, а также в других модулях информационной системы «Цифровой след студента». В-третьих, были выделены критерии различий между образовательными программами и на их основе разработан интерфейс рекомендательной системы, который наглядно представляет различия направлений обучения, а также предоставляет возможность абитуриенту давать запрос систему рекомендации в удобной форме.

Планируется протестировать данную рекомендательную систему в период приёмной кампании ИМиКН ТюмГУ и выявить, какие детали образовательных программ остаются не учтенными при данном подходе, что абитуриенту остается не понятным после получения рекомендации от системы. Кроме того, планируется расширить данную систему на другие институты Тюменского государственного университета. Благодаря разработанной структуре, данная возможность заложена в систему и требует лишь новых текстов РПД и данных из школьных словарей в соответствующих институтам областей знаний.

Работа выполнена в рамках научного проекта РФФИ №19-37-51028

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Семакин И.Г., Хеннер Е.К. Словарь терминов по информатике за 10 и 11 класс по учебнику И.Г. Семакина, Е.К. Хеннер URL: https://infourok.ru/slovar-terminov-po-informatike-klass-po-uchebniku-i-g-semakina-e-k-henner-2339662.html (Дата обращения: 13.03.2021)
2. Математика от А до Я: Справочное пособие. Издание третье с дополнениями / А.М. Романов, А.С. Попова, Г.Н. Леонов, Р.В. Дегтерева: Барнаул 2003
3. Иванов В.К., Виноградова Н.В. Современные методы автоматизированного извлечения ключевых слов из текста, Информационные ресурсы России, 2016. URL: https://cyberleninka.ru/article/n/metody-i-algoritmy-izvlecheniya-klyuchevyh-slov (Дата обращения: 13.03.2021)
4. Ванюшкин А.С., Гращенко Л.А. Методы и алгоритмы извлечения ключевых слов // Новые информационные технологии в автоматизированных системах, 2016. URL: https://cyberleninka.ru/article/n/metody-i-algoritmy-izvlecheniya-klyuchevyh-slov (Дата обращения: 13.03.2021)
5. Лисовенко А. NoSQL базы данных: понимаем суть, 2012. URL: https://habr.com/ru/post/152477/ (Дата обращения: 13.03.2021)
6. NoSQL Archive [сайт] URL: https://hostingdata.co.uk/nosql-database/ (Дата обращения: 13.03.2021)
7. PostgreSQL: The World's Most Advanced Open Source Relational Database [сайт] URL: https://www.postgresql.org/ (Дата обращения: 13.03.2021)
8. Apache HBsae [сайт] URL: https://hbase.apache.org/ (Дата обращения: 13.03.2021)
9. The basic facts for the Cassandra vs HBase comparison, 2019 [сайт] URL: https://www.intellectsoft.net/blog/hbase-vs-cassandra/(Дата обращения: 27.05.2021)
10. Близнюк Б.О., Васильева Л.В., Стрельников И.Д., Ткачук Д.С Современные методы обработки естественного языка // Вестник Харьковского национального университета имени В.Н. Карзина. 2017. С 14-26. URL:

https://docplayer.ru/83047890-Sovremennye-metody-obrabotki-estestvennogo-yazyka.html (Дата обращения: 27.05.2021)

11. Natural Language Toolkit [сайт] URL: Natural Language Toolkit — NLTK 3.5 documentation (Дата обращения: 13.03.2021)
12. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts, 320-332, 2015. URL: https://link.springer.com/chapter/10.1007%2F978-3-319-26123-2_31 (Дата обращения: 13.03.2021)
13. TextBlob: Simplified Text Processing [сайт] URL: https://textblob.readthedocs.io/en/dev/(Дата обращения: 27.05.2021)
14. CoreNLP [сайт] URL: https://stanfordnlp.github.io/CoreNLP/(Дата обращения: 27.05.2021)
15. Андреева О.В., Багиров М. Б., Данькина А.А., Федорова Т.О., Шевелёва М.М. Интеллектуальный анализ данных на базе Stanford CoreNLP для определения частей речи в русском языке // Системы и средства информ., 2018, том 28, выпуск 2, 145–153
16. Pattern [сайт] URL: https://github.com/clips/pattern (Дата обращения: 27.05.2021)
17. Django [сайт] URL: https://www.djangoproject.com/foundation/ (Дата обращения: 27.05.2021)
18. Flask [сайт] URL: https://flask.palletsprojects.com/en/2.0.x/ (Дата обращения: 27.05.2021)
19. G. Dwyer, Flask vs. Django: Why Flask might be better [сайт] 13.02.2017 URL: https://www.codementor.io/@garethdwyer/flask-vs-django-why-flask-might-be-better-4xs7mdf8v/(Дата обращения: 27.05.2021)
20. FastAPI [сайт] URL: https://fastapi.tiangolo.com/ (Дата обращения: 16.04.2021)
21. Берченко Д.А., Круг П.Г. Аналитический обзор методов визуализации данных // Евразийский Научный Журнал №5 2017, 2017 URL: http://journalpro.ru/articles/analiticheskiy-obzor-metodov-vizualizatsii-dannykh/ (Дата обращения 01.06.2021)

22. Карта наиболее популярных веб-сайтов в сети Интернет [сайт] URL: https://ia.net/ (Дата обращения: 27.05.2021)
23. Шурига Л. Современные подходы к визуализации данных, 2015 URL: http://datareview.info/article/sovremennyie-podhodyi-k-vizualizatsii-dannyih/ (Дата обращения: 27.05.2021)
24. Takada, S., Cuadros-Vargas, E., Impagliazzo, J. et al. Toward the visual understanding of computing curricula // Educ Inf Technol 25, 4231–4270, 2020). URL: https://doi.org/10.1007/s10639-020-10127-1 (Дата обращения: 13.03.2021)
25. Plotly [сайт] URL: https://plotly.com/ (Дата обращения: 27.05.2021)
26. DataHero [сайт] URL: https://datahero.com/(Дата обращения: 27.05.2021)
27. Chartjs [сайт] URL: https://www.chartjs.org/ (Дата обращения: 27.05.2021)
28. Jack Lloyd How to Write Pseudocode URL: How to Write Pseudocode: 15 Steps (with Pictures) - wikiHow (Дата обращения: 27.05.2021)
29. Иваненко О.А., Бакланов И.А., Чемакин Т.А., Воробьев А.М. Информационные сервисы для сопровождения индивидуальных образовательных траекторий студентов вуза // Информатизация образования и методика электронного обучения: цифровые технологии в образовании. Материалы IV Международной научной конференции. В двух частях. Красноярск: Сибирский федеральный университет, 2020, с. 115-118. URL: https://elibrary.ru/item.asp?id=44034502 (Дата обращения: 27.05.2021)
30. Шереметьева С.О., Осминин П.Г. Методы и модели автоматического извлечения ключевых слов // Южно-уральский государственный университет, г. Челябинск, 2015. URL: https://cyberleninka.ru/article/n/metody-i-modeli-avtomaticheskogo-izvlecheniya-klyuchevyh-slov (Дата обращения: 13.03.2021)

Приложение 1

МОДЕЛЬ ДАННЫХ

```
from sqlalchemy import Column, BigInteger, String, ForeignKey, Date
from sqlalchemy.orm import relationship
from .base import Base

#класс описания учебного плана

class AcademicPlan(Base):
    __tablename__ = 'academic_plans'
    id = Column(BigInteger, primary_key=True, index=True)
    name = Column(String(255))
    direction_id = Column(BigInteger, ForeignKey('directions.id'))
    direction = relationship('Direction', backref='plans')
    start_year = Column(Date)
    end_year = Column(Date)
```

```
from sqlalchemy import Column, BigInteger, String, ForeignKey
from .base import Base

#класс описания направления обучения

class Direction(Base):
    __tablename__ = 'directions'
    id = Column(BigInteger, primary_key=True, index=True)
    name = Column(String(255))
    form_id = Column(BigInteger, ForeignKey('forms.id'))  # форма обучения
```

```
from sqlalchemy import Column, BigInteger, String
from .base import Base

#класс описания дисциплины учебного плана AcademicPlan

class Subject(Base):
    __tablename__ = 'subjects'
    id = Column(BigInteger, primary_key=True, index=True)
    name = Column(String(255))
```

```
from sqlalchemy import Column, BigInteger, Integer, String, ForeignKey, Date
from .base import Base
from sqlalchemy.orm import relationship

#Класс описания метаданных РПД для дисциплины Subject
#из учебного плана AcademicPlan

class RpdMeta(Base):
    __tablename__ = 'rpd_metas'
    rpd_id = Column(BigInteger, primary_key=True, index=True)
```

```
    name = Column(String(255))
    year = Column(Date)
    subject_id = Column(BigInteger, ForeignKey('subjects.id'))
    subject = relationship('Subject', backref='rpds')
    direction_id = Column(BigInteger, ForeignKey('directions.id'))
    direction = relationship('Direction', backref='rpds')
    hbase_guid = Column(String(255))
```

Приложение 2

СХЕМА ДАННЫХ

```python
from typing import Optional
from datetime import date
from pydantic import BaseModel

#класс описания учебного плана

class AcademicPlan(BaseModel):
    id: int
    name: str
    direction_id: int
    dir_text: Optional[str]
    start_year: date
    end_year: date
```

```python
from typing import Optional
from pydantic import BaseModel

#класс описания направления обучения

class Direction(BaseModel):
    id: int
    name: Optional[str]
    form_id: int  #форма обучения
```

```python
from typing import Optional
from pydantic import BaseModel

#класс описания дисциплины учебного плана AcademicPlan

class Subject(BaseModel):
    id: int
    name: Optional[str]
```

```python
from typing import Optional, Dict
from datetime import date
from pydantic import BaseModel

#Класс описания метаданных РПД для дисциплины Subject
#из учебного плана AcademicPlan

class RpdMeta(BaseModel):
    id: int
    hbase_guid: str
    name: str
    year: date
```

```
    subject_id: int
    direction_id: int
#класс описания данных РПД из базы данных для дисциплины Subject
#из учебного плана AcademicPlan

class RpdOut(BaseModel):
    id: int
    name: str
    description: str

#класс описания расчетных данных РПД для дисциплины Subject
#из учебного плана AcademicPlan

class RpdInfo(BaseModel):
    id: int
    name: str
    words_stat: Dict
```

```
from typing import Optional, List
from datetime import date
from pydantic import BaseModel
from schemas.rpd import RpdInfo

#класс описания расчетных данных образовательной программы
#класс содержит основную информацию, содержащуюся на выходе
#из алгоритма рекомендательной системы

class Edu(BaseModel):
    name: str
    rating: float
    start_year: date
    end_year: date
    description: str
    rpds_info: List[RpdInfo]  #список строк: название РПД - частота

#список из образовательных программ Edu

class EduRecommendOut(BaseModel):
    recommended: Optional[List[Edu]]
```

Приложение 3

ВЗАИМОДЕЙСТВИЕ СО СТОРОНОЙ FRONT-END

```
from typing import Any, List
from datetime import date
from fastapi import APIRouter, Depends
from happybase import Connection
from sqlalchemy.orm import Session

from api import utils
from app import edu_recommendations, rpd
from schemas.edu_recommendations import EduRecommendOut
from schemas.rpd import RpdOut

router = APIRouter()

#запрос от стороны front на расчет рекомендуемых направлений
#входные данные – список ключевых слов
#выходные данные – список рекомендованных программ

@router.post("/edu", response_model=EduRecommendOut)
def get_recommend_edus(
        keywords: List[str],
        limit: int = 100,
        offset: int = 0,
        db: Session = Depends(utils.get_db),
        hbase: Connection = Depends(utils.get_hbase),
        current_user: dict = Depends(utils.get_current_user),
) -> Any:
    return edu_recommendations.get_recommend_edus(db, hbase, keywords, limit,
offset)

#запрос от стороны front на получение исходной информации
#об изучаемых дисциплинах в конкретном направлении
#входные данные – даты начала и конца обучения,
#номер направления
#выходные данные – список информации о дисциплинах –
#название и описание

@router.get("/rpd", response_model=List[RpdOut])
def get_rpds_by_edu(
        direction_id: int,
        start_year: date,
        end_year: date,
        limit: int = 100,
        offset: int = 0,
        db: Session = Depends(utils.get_db),
        hbase: Connection = Depends(utils.get_hbase),
        current_user: dict = Depends(utils.get_current_user),
) -> Any:
    return rpd.get_rpds_by_edu(db, hbase, direction_id, start_year.year,
end_year.year, limit, offset)
```

Приложение 4

АЛГОРИТМЫ И ЗАПРОСЫ ДАННЫХ

```
from models.academic_plan import AcademicPlan
from sqlalchemy import text
from happybase import Connection
from sqlalchemy.orm import Session

#функция получения наиболее актуального учебного плана

def get_last_academic_plans(db: Session, hbase: Connection, limit: int = 100,
offset: int = 0):
    sqlstr = text('SELECT id, name, direction_id, start_year, end_year FROM
ACADEMIC_PLAN WHERE start_year IN (SELECT MAX(start_year) FROM ACADEMIC_PLAN GROUP
BY direction_id)')
    plans_list = db.query(AcademicPlan).from_statement(sqlstr).all()
    return plans_list
```

```
from models import RpdMeta
from happybase import Connection
from sqlalchemy.orm import Session
from datetime import date

#функция получения информации о РПД дисциплин в определенном учебном плане

def get_rpds_by_edu(db: Session, hbase: Connection, dir_id: int, year_start: int,
year_end: int, limit: int = 100, offset: int = 0):
    rpd_list = db.query(RpdMeta).filter(RpdMeta.direction_id == dir_id)
    result = []

    for text in rpd_list:
        guid = text.hbase_guid
        if guid is None:
            continue

        row = hbase.table('rpds').row(guid.encode(), columns=(b'text:text',))

        if not row:
            continue

        result.append({'id': text.rpd_id, 'name': text.name, 'description':
row[b'text:text'].decode()})
    return result
```

```
from app.academic_plans import get_last_academic_plans
from models import RpdMeta
from typing import List
from happybase import Connection
from sqlalchemy.orm import Session
from datetime import date
```

```
#функция получения информации об учебном плане из хранилища

def get_rpds_recommend_info(db: Session, hbase: Connection, dir_id: int, year_start:
int, year_end: int, limit: int = 100, offset: int = 0):
    rpd_list = db.query(RpdMeta).filter(RpdMeta.direction_id == dir_id)
    result = list()
    for text in rpd_list:
        guid = text.hbase_guid
        if guid is None:
            continue

        row = hbase.table('rpdwords').row(guid.encode(), columns=(b'text:text',))

        if not row:
             continue

        word_dict = json.loads(row["char_int:words"].decode())
        count = -1
        tb = hbase.table('numchars')
        query_str = "SingleColumnValueFilter ('text_guid', 'text_guid', =,
'substring:" + str(guid)+ "')"

        for key, data in tb.scan(filter=query_str, limit=1):
            count = int(data[b'char_int:numwords'])

        if count == -1:
          continue

        result.append([text.rpd_id, count, text.name, word_dict])

    return result

#функция получения списка рекомендованных образовательных программ

def get_recommend_edus(db: Session, hbase: Connection, input_words: List[str],
limit: int = 100, offset: int = 0):
    academic_plans = get_last_academic_plans(db, hbase, limit, offset)
    programs = []

    for row in academic_plans:
        op_name = row.name  # name
        op_rat = 0
        op_words_count = 0
        rpd_count = 0
        rpds_rats = []  # РПД - сколько раз встретилось слово
        metas = get_rpds_recommend_info(db=db, hbase=hbase, dir_id=row.direction_id,
year_start=row.start_year.year, year_end=row.end_year.year,limit=limit,
offset=offset)

        for met in metas:
            all_word_count = met[1]
            if all_word_count == 0:
                continue
            op_words_count += all_word_count
```

```
            rpd_count += 1
           rpd_words = met[3]
            rpd_words_counts = {}

            for in_word in input_words:
                for word, count in rpd_words.items():
                    if in_word == word:
                        op_rat += count
                        rpd_words_counts[in_word] = count

            rpds_rats.append({'id': met[0], 'name': met[2], 'words_stat':
rpd_words_counts})

        if rpd_count == 0:
            continue

        op_ball = op_rat / op_words_count
        programs.append({"name": op_name, "start_year": row.start_year,"end_year":
row.end_year, "rating": op_ball, "description": "", "rpds_info": rpds_rats})

    if len(programs) > 0:
        programs = sorted(programs, key=lambda val: val["rating"], reverse=True)

    return {'recommended': [programs[0], programs[1], programs[2]] if len(programs)
> 3 else programs }
```