

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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
И ПРОВЕРЕНО НА ОБЪЁМ
ЗАИМСТВОВАНИЯ

Заведующий кафедрой

д.п.н., профессор

И. Г. Захарова

29 июня 2018 г.

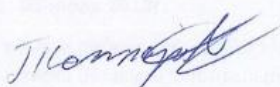


МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
РАЗРАБОТКА РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ВЫБОРА
СПЕЦИАЛЬНОСТИ НА ОСНОВЕ ПРОФИЛЕЙ АБИТУРИЕНТА В
СОЦИАЛЬНЫХ СЕТЯХ

02.04.03. Математическое обеспечение и администрирование
информационных систем

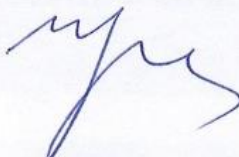
Магистерская программа «Высокопроизводительные вычислительные
системы»

Выполнил работу
Студент 2 курса
очной формы обучения



Токтарев Василий
Васильевич

Научный руководитель
Доктор педагогических
наук, профессор



Захарова Ирина
Гелиевна

Консультант
Ст. преподаватель



Аксёнов Михаил
Валерьевич

Рецензент
Старший преподаватель
(к.н.)



Пушкарев Александр
Николаевич

Тюмень, 2018

ОГЛАВЛЕНИЕ

Введение.....	2
Глава 1. Обзор существующих технологий.....	5
1.1 Обзор рекомендательных систем	5
1.2 Обзор технологий сбора данных из социальных сетей.....	7
1.3 Обзор алгоритмов выдачи рекомендаций	10
1.4 Предварительная обработка данных	12
1.5 Описание метода латентно-семантического анализа	14
1.6 Описание концепции системы	17
Глава 2. Программная реализация.....	18
2.1 Реализация сбора данных о пользователях социальной сети.....	18
2.2 Реализация предварительной обработки данных	21
2.3 Реализация сервиса выдачи рекомендации	23
2.4 Реализация метода выдачи рекомендации	26
Глава 3. Оценка качества работы системы.....	28
3.1 Описание метода оценки качества	28
3.2 Интерпретация результата оценки	30
Глава 4. Описание пользовательского интерфейса	31
Заключение	34
Список литературы	36

ВВЕДЕНИЕ

Тюменский государственный университет (ТюмГУ) насчитывает 11 институтов, и для абитуриента ТюмГУ выбор института является сложной задачей. Критерии выбора направления подготовки зависят не только от явных качеств абитуриента (такие как количество и качество приобретённых знаний, выбранные экзаменационные дисциплины, экзаменационный балл и др.), но и от неявных (области интереса). Поскольку от выбора направления абитуриентом зависит его дальнейшая карьера, он нуждается в рекомендации, которая учитывала бы его неявные качества. Одним из методов решения подобной проблемы может стать рекомендательная система поступления.

Рекомендательные системы — программы, сервисы или модули системы, которые пытаются предсказать, какие объекты (фильмы, музыка, книги, новости, и т.п.) будут интересны пользователю, имея определённую информацию о его профиле [1].

Рекомендательные системы присутствуют в большинстве сервисов и приложений. Среди них: онлайн-кинотеатры, новостные ленты, социальные сети, музыкальные сервисы и т. п. Следствием развития рекомендательных систем стала выдача более точных рекомендаций за счёт улучшения алгоритмов и обработки пользовательских данных.

Некоторые современные рекомендательные системы дают более точные рекомендации, чем человек.

Например, на сайте «imdb.com» пользователи оценивают фильмы по десятибалльной шкале. Система агрегирует эти оценки и из них получается средний рейтинг фильма. На этом же сайте есть блок с рекомендациями для конкретного пользователя. Если пользователь зашел на сайт и оценил несколько фильмов, «[imdb](http://imdb.com)» сможет порекомендовать этому пользователю еще какие-нибудь фильмы.

Аналогом для музыки является сервис «last.fm». Он рекомендует пользователю исполнителей, альбомы, мероприятия, которые могут быть ему

интересны. Еще одним примером рекомендательной системы может послужить популярный в США сервис «Pandora» является персональным радио, постепенно подстраивающимся под пользователя на основе его оценок.

Ещё одна область рекомендаций – рекомендация товаров. Так, сервис «Amazon» обладает системой рекомендации товаров на основе тех, что пользователь уже купил или оценил. Такая рекомендательная система обеспечивает определенную выгоду как покупателю, так и магазину.

В рамках данного исследования, используя информацию о пользователе, полученную из профиля социальной сети, система должна предоставить рекомендацию, какой из институтов ТюмГУ больше всего совпадает с интересами абитуриента. В качестве такой информации выступают анкетные данные (любимые книги, любимая музыка, любимые телешоу, любимые игры, любимые цитаты, деятельность, интересы), подписки (названия подписок, описание тематики групп подписок), тексты записей и комментарии, оставленные пользователем, тексты записей и комментариев, которые пользователь отметил отметкой «мне нравится».

Данная система должна удовлетворять следующим требованиям:

- Система должна автоматически собирать информацию о пользователе в обезличенном виде (для работы системы данные не обязаны быть персональными).
- Для сбора информации о пользователе система должна использовать профиль пользователя в социальной сети «ВКонтакте».
- В качестве результата должна предоставляться однозначная рекомендация в определённый институт ТюмГУ, либо сообщение о недостаточном объёме предоставленной информации о пользователе в таком случае.

Исходя из представленных требований бизнес-модель рекомендательной системы выглядит следующим образом:



Рисунок 1.1 – Бизнес-модель рекомендательной системы

Целью выпускной квалификационной работы является разработать систему рекомендаций поступления в ТюмГУ.

В соответствии с поставленной целью необходимо решить следующие **задачи**:

- изучить различные механизмы предоставления рекомендаций;
- разработать концепцию рекомендательной системы;
- разработать сервис сбора данных о пользователях из социальной сети;
- разработать рекомендательную систему;
- разместить рекомендательную систему в сети Интернет.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ ТЕХНОЛОГИЙ

1.1 ОБЗОР РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ

Поиск информации о подобных системах рекомендаций поступления в университет, удовлетворяющих требованиям, приведённым во введении, показал, что в общем доступе таких систем не было найдено.

Наиболее похожими системами являются рекомендательные системы выбора профессии на основе психологических факторов. Отличие таких систем заключается в явном сборе данных (пользователь должен пройти анкетирование, ответив на множество вопросов в тестовом виде или со свободным ответом).

По методу выдачи рекомендации рекомендательные системы делятся на множество типов. Среди всех типов выделяют два наиболее крупных и фундаментальных: фильтрация на основе содержания и коллаборативная фильтрация [2]. Рассмотрим их более подробно.

Фильтрация на основе содержания (контентная фильтрация). Рекомендательные системы такого типа основаны на описании пользователя и объектов, т.е. пользователю рекомендуются объекты, похожие на те, что этот пользователь уже употребил. В данном случае схожесть оценивается по признакам содержимого объектов.

Системой с таким типом фильтрации может быть, например, платформа для блога. В таком случае используя этот тип фильтрации будет использована информация о просмотренных записях блога и прочие характеристики блогов (автор, тематика и т.п.). Если определённый пользователь читает чаще всего читает блоги определённой тематики и часто оставляет под этими блогами свои комментарии, то контентная фильтрация может использовать эту информацию для определения похожего контента и сможет рекомендовать такой контент пользователю [3].

Коллаборативная фильтрация. Рекомендательные системы такого типа основаны на сборе и анализе большого объёма информации пользовательского поведения и действий. Это метод рекомендации, при котором анализируется только реакция пользователей на объекты. Реакцией может быть употребление объекта, оценивание объекта и прочие действия с объектом.

Примером такой фильтрации может быть сайт, предлагающий посетителям рекомендации относительно блогов. На основе информации от многих пользователей, которые подписываются на блоги и читают их, можно сгруппировать этих пользователей по их предпочтениям. Например, можно объединить в одну группу пользователей, которые читают несколько одних и тех же блогов. По этой информации идентифицируются самые популярные блоги среди тех, которые читают участники этой группы. Затем — конкретному пользователю этой группы — вы рекомендуете самый популярный блог из тех, на которые он еще не подписан и которые он не читает [3].

Используя комбинацию из этих двух типов фильтрации выделяют гибридную фильтрацию.

Гибридная фильтрация. Гибридная фильтрация — это такой тип фильтрации, которая для вычисления результата использует в определённой мере результаты контентной и коллаборативной фильтрации.

В данной работе данными, поступающими в систему являются тексты, описывающие пользователя, которые в свою очередь собираются из профиля пользователя в социальной сети (анкетные данные, названия групп, на которые подписан пользователь, тексты записей, оставленные пользователем).

Для обработки текстовых данных используется область искусственного интеллекта – обработка естественного языка.

Используя представленные технологии становится возможным реализовать рекомендательную систему поступления в ТюмГУ.

1.2 ОБЗОР ТЕХНОЛОГИЙ СБОРА ДАННЫХ ИЗ СОЦИАЛЬНЫХ СЕТЕЙ

Для получения данных из социальных сетей используется Web API.

API – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

Web API – интерфейс программирования для веб-служб.

Социальная сеть «ВКонтакте» обладает API, который называется «VK API». Доступ к этому API находится по адресу:

[https://api.vk.com/method/MN?P&access_token = AC&v = V](https://api.vk.com/method/MN?P&access_token=AC&v=V)

где *MN* – название метода API, к которому необходимо обратиться; *P* – входные параметры соответствующего метода API, последовательность пар ключ-значение, разделенных амперсандом; *AC* – ключ доступа приложения; *V* – используемая версия API (Использование этого параметра применяет некоторые изменения в формате ответа различных методов) [4].

Для получения информации о пользователях используется метод `users.get`. В качестве параметров он принимает:

- `user_ids` – перечисленные через запятую идентификаторы пользователей или их короткие имена.
- `fields` – список дополнительных полей профилей, которые необходимо вернуть.
- `name_case` – падеж для склонения имени и фамилии пользователя.

Для получения списка записей со стены пользователя или сообщества используется метод `wall.get`. В качестве параметров он принимает:

- `owner_id` - идентификатор пользователя или сообщества, со стены которого необходимо получить записи.
- `domain` – короткий адрес пользователя или сообщества.
- `offset` – смещение, необходимое для выборки определенного подмножества записей.

- `count` – определяет, какие типы записей на стене необходимо получить. Возможные значения:
 - `suggests` — предложенные записи на стене сообщества.
 - `postponed` — отложенные записи.
 - `owner` — записи владельца стены.
 - `others` — записи не от владельца стены.
 - `all` — все записи на стене.
- `extended` – флаг, при значении 1 в ответе будут возвращены дополнительные поля `profiles` и `groups`, содержащие информацию о пользователях и сообществах.
- список дополнительных полей для профилей и сообществ, которые необходимо вернуть.

Для получения информации о подписках используется метод `users.getSubscriptions`. В качестве параметров он принимает:

- `user_id` – идентификатор пользователя, подписки которого необходимо получить.
- `extended` – флаг, при значении 1 возвращает объединенный список, содержащий объекты `group` и `user` вместе.
- `offset` – смещение необходимое для выборки определенного подмножества подписок.
- `count` – количество подписок, которые необходимо вернуть.
- `fields` – список дополнительных полей для объектов `user` и `group`, которые необходимо вернуть.

Платформа «VK API» накладывает ограничения на частоту вызова методов API в 3 запроса в секунду. Для того, чтобы сократить количество вызываемых методов предполагается использовать хранимые процедуры, которые позволяют исполнять код на стороне сервера API, при этом в одном вызове хранимой процедуры допускается осуществлять до 25 вызовов API-методов.

Доступ к хранимым процедурам находится по адресу:

https://api.vk.com/method/execute.PN?P

где *PN* – название вызываемой процедуры, *P* – список параметров.

Используя эту возможность предполагается разработать хранимую процедуру `getUserInfo`, в которой объединены вызовы методов `users.get`, `wall.get` и `users.getSubscriptions`.

Таким образом, используя Web API платформы «VK API» предоставляется возможность собирать нужные данные о пользователях «ВКонтакте».

1.3 ОБЗОР АЛГОРИТМОВ ВЫДАЧИ РЕКОМЕНДАЦИЙ

Для составления вектора интереса пользователя социальной сети «ВКонтакте», требуются следующие признаки:

- анкетные данные:
 - любимая музыка,
 - любимые фильмы,
 - любимые телешоу,
 - любимые книги,
 - любимые игры,
 - любимые цитаты,
 - описание деятельности,
 - интересы,
 - «о себе»
- текст записей на стене пользователя;
- текст названий подписок пользователя;
- текст описания тематики подписок.

Из подписок следует брать именно текст названий, а не уникальный идентификатор по той причине, что подписки пользователя часто изменяются (подписка на новые, отписка от старых), при этом интересы пользователя не изменяются. Названия подписок точнее характеризуют интересы пользователя.

В этом случае необходимо обратиться к направлению компьютерного анализа «обработка естественного языка».

Обработка естественного языка — общее направление искусственного интеллекта, которое изучает проблемы компьютерного анализа и синтеза естественных языков. Среди задач обработки естественного языка присутствует задача тематического моделирования, т.е. анализа взаимосвязи между коллекцией документов и терминами в них встречающимися. Если взять в качестве документов вышеперечисленные признаки пользователя

«ВКонтакте»), вычислить на их основе вектор интересов пользователя и найти наименьшее расстояние до векторов интересов, полученными на основе обучающей выборки, то получается алгоритм, который позволяет предоставить рекомендацию поступления на основе интересов пользователя.

Следовательно, алгоритм должен выполнять следующие задачи:

- строить тематическую модель коллекции документов;
- вычислять вектор интересов пользователя;
- вычислять расстояние между вектором интересов пользователя и векторами интересов пользователей, которые окончили обучение в ТюмГУ.

Среди алгоритмов, выполняющих эти задачи, можно выделить следующие:

- латентно-семантический анализ (ЛСА);
- **Латентное размещение Дирихле**. Это часто применяемый метод тематического моделирования в машинном обучении и информационном поиске, который объясняет результаты наблюдений с помощью неявных групп. Это позволяет выявить зависимость некоторых частей данных. Например, если наблюдениями являются слова, собранные в некоторые документы, то утверждается, что каждый документ представляет собой смесь небольшого количества тем и что появление каждого слова связано с одной из тем документа.

Среди этих алгоритмов используем латентно-семантический анализ, т.к.:

- ЛСА является наилучшим методом для выявления латентных зависимостей внутри множества документов.
- Частично снимает полисемию (наличие у слова двух и более значений) и омонимию (одинаковые по написанию, но разные по значению единицы языка). [5]

1.4 ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ

Предварительная обработка данных подразумевает:

- лемматизацию;
- стэмминг;
- отбрасывание стоп-слов;
- отбрасывание редких слов;
- выделение ключевых фраз.

Лемматизация – это приведение каждого слова в документе к его нормальной форме. Разработка хорошего лемматизатора требует составления грамматического словаря со всеми формами слов, либо аккуратной формализации правил языка со всеми исключениями, что является трудоёмким проектом. Известные лемматизаторы совершенствуются постепенно. Их недостатком является неполнота словарей, особенно по части специальной терминологии и неологизмов, которые во многих приложениях как раз и представляют наибольший интерес.

Стэмминг – это более простая технология, которая состоит в отбрасывании изменяемых частей слов, главным образом, окончаний. Она не требует хранения словаря всех слов и основана на правилах морфологии языка. Недостатком стемминга является большее число ошибок. Стемминг хорошо подходит для английского языка, но хуже подходит для русского.

Отбрасывание стоп-слов. Слова, встречающиеся во многих текстах различной тематики, бесполезны для тематического моделирования, и могут быть отброшены. К ним относятся предлоги, союзы, числительные, местоимения, некоторые глаголы, прилагательные и наречия. Число таких слов обычно варьируется в пределах нескольких сотен. Их отбрасывание почти не влияет на длину словаря, но может приводить к заметному сокращению длины некоторых текстов.

Выделение редких слов. Слова, встречающиеся в длинном документе слишком редко, например, только один раз, следует выделять, полагая, что данное слово более весомо характеризует тематику данного документа.

Выделение ключевых фраз. При обработке коллекций научных, юридических или других специальных текстов вместо отдельных слов выделяют ключевые фразы, словосочетания, являющиеся терминами предметной области. Это отдельная довольно сложная задача, для решения которой используются тезаурусы, составленные экспертами, либо методы машинного обучения, при этом для формирования обучающих выборок всё равно приходится привлекать экспертов [6].

В данной работе в качестве предварительной обработки данных имеет смысл использовать лемматизацию, отбрасывание стоп-слов, выделение редких слов. Не имеет смысл брать стэмминг, так как этот способ плохо подходит для русского языка.

1.5 ОПИСАНИЕ МЕТОДА ЛАТЕНТНО-СЕМАНТИЧЕСКОГО АНАЛИЗА

Латентно-семантический анализ (ЛСА) – это метод обработки информации на естественном языке, анализирующий взаимосвязь между коллекцией документов и терминами в них встречающимися, сопоставляющий некоторые факторы (тематики) всем документам и терминам. В основе метода латентно-семантического анализа лежат принципы факторного анализа, в частности, выявление латентных связей изучаемых объектов. ЛСА часто используется для поиска информации, классификации документов, моделях понимания и других областях, где требуется выявление главных факторов из массива информационных данных [7].

Алгоритм состоит из следующих шагов:

- составление терм-документной матрицы;
- сокращение числа переменных операцией сингулярного разложения терм-документной матрицы;
- вычисление метрик.

Составление терм-документной матрицы. В качестве исходной информации используется терм-документная матрица, которая представляет собой математическую матрицу, описывающую частоту терминов, которые встречаются в коллекции документов. Строки соответствуют документам в коллекции, а столбцы – терминам. Элементами этой матрицы являются веса

Сокращение числа переменных операцией сингулярного разложения. Сингулярное разложение – это математическая операция, раскладывающая терм-документную матрицу на три составляющих, т.е. представление в виде:

$$M = U \times W \times V^t$$

где U и V^t – ортогональные матрицы, а W – диагональная матрица, значения на диагонали которой называются сингулярными коэффициентами

матрицы M . Сингулярное разложение позволяет выделить ключевые составляющие исходной матрицы.

Основная идея ЛСА состоит в том, что если в качестве матрицы M использовалась терм-документная матрица, то матрица M^* , содержащая только k первых линейно независимых компонент, отражает основную структуру различных зависимостей, присутствующих в исходной матрице. Структура зависимостей определяется весовыми функциями термов.

Как правило, выбор k зависит от поставленной задачи и подбирается эмпирически. Если выбранное значение k слишком велико, то метод теряет свою мощьность и приближается по характеристикам к стандартным векторным методам. Слишком маленькое значение k не позволяет улавливать различия между похожими термами или документами. Если необходимо выбирать значение k автоматически, то можно, например, установить пороговое значение сингулярных коэффициентов и отбрасывать все строки и столбцы, соответствующие сингулярным коэффициентам, не превышающим данного порогового значения [8].

Преимущество сингулярного разложения состоит в том, что оно выделяет ключевые составляющие матрицы, позволяя игнорировать шумы [5].

Результатом алгоритма становится набор терминов, объединённые в группы (документы). Для того чтобы вычислить рекомендацию, необходимо найти наименьшее расстояние между состоящими из k линейно независимых компонент вектором искомого документа и документами из выборки, сгруппированными по институтам выпуска. Группа документов с наименьшим расстоянием является рекомендацией поступления в институт, по которому эти документы были сгруппированы.

Вычисление метрик. Для вычисления расстояния используют два метода: вычисление скалярного произведения или вычисление Евклидовой метрики. В случае рекомендательной системы используется вычисление

Евклидовой метрики, так как она обеспечивает большую точность. Евклидова метрика r определяется следующим образом:

$$r = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

где p и q – векторы линейно независимых компонент, для которых находится расстояние.

Таким образом, становится возможным предоставление рекомендации, используя информацию профиля пользователя на основе предварительно собранной информации профилей пользователей, уже закончивших ТюмГУ.

1.6 ОПИСАНИЕ КОНЦЕПЦИИ СИСТЕМЫ

Система должна обладать следующим функционалом:

- Автоматически формировать корпус профилей студентов, окончивших ТюмГУ на основе профиля в соцсети «ВКонтакте».
- Использовать ссылку на профиль «ВКонтакте» пользователя для предоставления рекомендации.
- Результатом является однозначная рекомендация в определённый институт ТюмГУ.

Исходя из перечисленных требований и состава системы, концепция выглядит следующим образом:

1. При первом запуске сервис начинает сбор и обработку данных пользователей «ВКонтакте», окончивших ТюмГУ. На этом этапе при обращении на сайт сервиса показывается сообщение о просьбе подождать окончания запуска сервиса.
2. Сервис выполняет предобработку данных.
3. Пользователь заходит на страницу сервиса и вводит ссылку на свой профиль в «ВКонтакте», нажимает кнопку «Продолжить».
4. Сервис загружает информацию о профиле пользователя в «ВКонтакте», выполняет вычисление рекомендации.
5. Сервис отображает однозначную рекомендацию пользователю о поступлении в определённый институт ТюмГУ.

Таким образом реализованная концепция позволяет решать поставленную перед системой задачу.

ГЛАВА 2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1 РЕАЛИЗАЦИЯ СБОРА ДАННЫХ О ПОЛЬЗОВАТЕЛЯХ СОЦИАЛЬНОЙ СЕТИ

Для реализации сбора данных о пользователях социальной сети «ВКонтакте» требуется реализовать процедуру `GetUserInfo`, рассмотренную в главе 1.2. Платформа «VK API» предоставляет реализацию хранимых процедур на языке программирования JavaScript. Код реализованной процедуры представлен в листинге 2.1.

```
var id = Args.id;
var result = {};
result = API.users.get({
    user_ids: id,
    fields: "verified, sex, bdate, city, country, home_town,
domain, has_mobile, contacts, site, education, universities,
schools, status, last_seen, followers_count, common_count,
occupation, nickname, relatives, relation, personal,
connections, exports, wall_comments, activities, interests,
music, movies, tv, books, games, about, quotes, can_post,
can_see_all_posts, can_see_audio, can_write_private_message,
can_send_friend_request, is_favorite, is_hidden_from_feed,
timezone, screen_name, maiden_name, crop_photo, is_friend,
friend_status, career, military, blacklisted, blacklisted_by_me"
})[0];
result.wall = API.wall.get({
    owner_id: id,
    filter: "owner"
});
result.subscriptions = API.users.getSubscriptions({
    user_id: id,
    extended: 1,
    count: 200
});
return result;
```

Листинг 2.1 – Процедура `GetUserInfo`

Для вызова хранимой процедуры `GetUserInfo` используется библиотека «VkNet». Эта библиотека используется для вызовов методов «VK API».

Прежде всего, нужно получить уникальные идентификаторы пользователей для каждого института ТюмГУ. Для этого необходимо вызвать

метод `Users.Search(UserSearchParams @params)`, который возвращает список пользователей в соответствии с заданным критерием поиска. Параметр `@params` это объект с параметрами поиска, среди них есть параметры:

- **`university_faculty`**, представляет идентификатор института;
- **`university_year`**, представляет год окончания института.

Идентификаторами института являются целочисленные константы.

Программный код, реализующий получение списка уникальных идентификаторов пользователей представлен в листинге 2.2.

```
var userIds = new List<long>();
foreach (var year in Years)
    foreach (int facultyId in Faculties.Enumerate())
    {
        var response = vkApi.Users.Search(new UserSearchParams
        {
            University = UniversityId,
            UniversityFaculty = facultyId,
            UniversityYear = year,
            Count = 1000
        });
        userIds.AddRange(response.Select(u => u.Id));
    }
```

Листинг 2.2 – Получение списка уникальных идентификаторов пользователей

Затем необходимо получить список полной информации. Для этого необходимо реализовать метод `FetchRemote`, где для каждого полученного идентификатора пользователя вызовем процедуру `GetUserInfo` с параметром `userId` равным идентификатору получаемого пользователя. Код получения полной информации пользователей представлен в листинге 2.3.

```
public enum Faculties : int
{
    ИПиП = 165153,
    ИФК = 190438,
    ИМиКН = 2158741,
    ФТИ = 2158742,
    ИНБИО = 2158743,
    ИнЗем = 2158744,
    ИФИЖ = 2158745,
    ИИПН = 2158746,
    ИГиП = 2158747,
    ФЭИ = 2158748,
```

```

    ИнХим = 2175698
}

List<FetchedUser> FetchRemote()
{
    var userIds = new List<long>();
    foreach (var year in Years)
        foreach (int facultyId in Faculties.Enumerate())
        {
            var response = vkApi.Users.Search(new UserSearchParams
            {
                University = UniversityId,
                UniversityFaculty = facultyId,
                UniversityYear = year,
                Count = 1000
            });
            userIds.AddRange(response.Select(u => u.Id));
        }
    var results = new List<FetchedUser>();
    foreach (var userId in userIds)
    {
        var infoResponse = vkApi.Call(
            "execute.getUserInfo",
            new VkNet.Utills.VkParameters(
                new Dictionary<string, string>
                {
                    ["id"] = userId.ToString()
                }
            ));
        var fetchedResponse = JsonConvert.DeserializeObject(
            <FetchedResponse>(infoResponse.RawJson));
        results.Add(fetchedResponse.Response);
    }
    return results;
}

```

Листинг 2.3 – Получение полной информации о пользователях

Результатом метода `FetchRemote` является список пользователей с идентификаторами института и всеми необходимыми данными для проведения дальнейшего анализа.

После получения полной информации о пользователях можно приступить к предварительной обработке данных.

2.2 РЕАЛИЗАЦИЯ ПРЕДВАРИТЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ

Предварительная обработка данных в рекомендательной системе поступления включает в себя:

- лемматизацию,
- отбрасывание стоп-слов,
- выделение редких слов.

Для реализации лемматизации используется библиотека ПО «Грамматический словарь русского языка» фирмы Solarix. Это ПО компьютерной грамматики русского языка, которое используется для склонения, спряжения, поиска синонимов, антонимов, лемматизации, морфологического и синтаксического анализа текста. Данное ПО обладает API для лемматизации слов. [9]

Для использования этого API необходимо подключить к проекту рекомендательной системы поступления файлы `lemmatizator.dll` (предоставляющий API) и `lemmatizator_fx.dll` (обёртка для использования в платформе .NET).

Прежде необходимо загрузить базу данных лемматизатора, которая поставляется вместе с дистрибутивом (файл `lemmatizator.db`). Для загрузки используется метод `sol_LoadLemmatizator8(string filePath, int Flags)`, где `filePath` – путь к файлу базы лемматизатора. Метод возвращает ссылку на лемматизатор. Код загрузки лемматизатора представлен в листинге 2.4

```
var engine = solarix.sol_LoadLemmatizator8(  
    @"..\..\lemmatizator.db", 0);
```

Листинг 2.4 – Загрузка базы данных лемматизатора

Для получения леммы слова используется метод `sol_GetLemma8(IntPtr hEngine, string word, string result, int bufSize)`, где `hEngine` – ссылка на загруженный лемматизатор, `word` – слово, для которого необходимо выполнить лемматизацию, `result` – результат лемматизации, `bufSize` – длина

результата в байтах. Метод возвращает число нормальных форм слова, чаще всего это 1. Если число нормальных форм больше 1, то в result вернётся первая найденная нормальная форма в базе лемматизатора. Если число нормальных форм равно 0, то result вернёт значение, равное word.

Для отбрасывания стоп-слов используется грамматический словарь ПО «Грамматический словарь русского языка», который включает в себя 205 тысяч словарных статей.

В рекомендательной системе отбрасываются слова служебных частей речи: предлоги, союзы и частицы. Слова служебных частей речи не имеют лексического значения, поэтому не имеет смысл держать их в корпусе слов.

Для выделения редких слов осуществляется поиск слова в массиве общеупотребительных слов. Если искомое слово не является общеупотребительным, то число употреблений этого слова в документе умножается на коэффициент $k = 3$.

Корпус общеупотребительных слов извлекается из грамматического словаря ПО «Грамматический словарь русского языка». Для поиска слова в грамматической базе используется метод `sol_FindWord(IntPtr hEngine, string Word)`, где `hEngine` – ссылка на загруженный лемматизатор, `word` – искомое слово. Метод возвращает 1, если слово найдено и 0, если слово не найдено.

Используя приведённые методы API ПО «Грамматический словарь русского языка» в результате получается предобработанный корпус слов документа. Код с реализацией предобработки данных приведён в листинге 2.5.

```
var lemmed = await body.ForEach(w =>
{
    string normalized = null;
    int bufSize;
    var normalized = sol_GetLemma8(
        engine, w, normalized, bufSize);
    return sol_FindWord(engine, normalized) > 0
        ? normalized
        : null;
}).Where(w => w != null).ToArrayAsync();
```

Листинг 2.5 – Предобработка данных

2.3 РЕАЛИЗАЦИЯ СЕРВИСА ВЫДАЧИ РЕКОМЕНДАЦИИ

Рекомендательная система автоматизирует следующие процессы предоставления рекомендации:

1. Загрузка информации о пользователях, выполнение предобработки и группировки.
2. Загрузка информации для потребовавшего рекомендацию пользователя из социальной сети «ВКонтакте».
3. Предоставление рекомендации на основе полученной информации.

Причём пункт 1 представляет из себя продолжительный процесс. Например, число профилей студентов, окончивших любой институт ТюмГУ за последние 5 лет примерно равно 8000. С учётом ограничения платформы «VK API», которая ограничивает число вызовов максимум 3 в секунду, время на загрузку этих пользователей составит 45 минут без предварительного поиска идентификаторов этих пользователей.

Эта особенность подразумевает разработку системы, используя микросервисную архитектуру, т.е. работа целого сервиса будет разделена на взаимосвязанной работе отдельных сервисов, называемых микросервисами.

[10]

В рекомендательной системе выделены следующие микросервисы:

- сервис загрузки данных из «ВКонтакте»;
- сервис предобработки данных;
- сервис выдачи рекомендации.

При первом запуске системы будет инициирован сервис выдачи рекомендации, который в свою очередь инициирует сервис загрузки данных из «ВКонтакте». Сервис выдачи рекомендации не будет выполнять вычисление и выдачу рекомендации пока не получит результаты от сервиса загрузки данных из «ВКонтакте». Сервис загрузки данных из «ВКонтакте» в

свою очередь использует сервис предобработки данных после получения всей информации о профилях студентов, окончивших ТюмГУ.

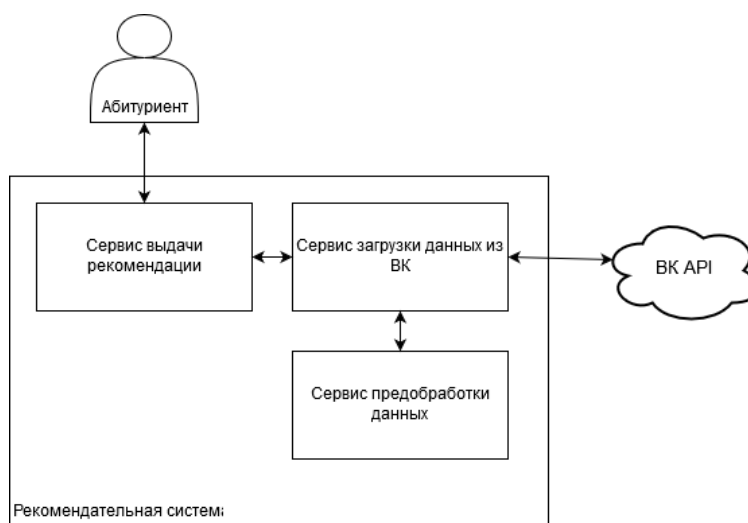


Рисунок 2.1 – Взаимодействия сервисов системы

Исходя из перечисленных особенностей платформа разработки должна предоставлять возможность разработки веб-сервиса, используя микросервисную архитектуру и обладать наличием инструментария для реализации алгоритма латентно-семантического анализа.

В качестве платформы разработки используется «ASP.Net Core». Это свободный открытый веб-фреймворк, использующий платформу «.Net», который обладает всеми необходимыми средствами для разработки рекомендательной системы. [11]

На этой платформе микросервисами являются классы, которые зарегистрированы в методе конфигурации веб-приложения. Такие классы можно вызывать из HTTP-контекста веб-приложения, например, из контроллера. Для конфигурации используются методы `WebHostBuilder`, которые влияют на время жизни экземпляра класса:

- **AddSingleton()**. В этом случае для зарегистрированного класса будет создан один экземпляр класса на всё время жизни веб-приложения.

- **AddTransient()**. В этом случае для зарегистрированного класса будет создан один экземпляр класса каждый раз, когда к нему выполняется обращение.
- **AddScope()**. В этом случае для зарегистрированного класса будет создан один экземпляр класса в рамках одного веб-запроса.

В рекомендательной системе поступления экземпляры сервисов загрузки данных из «ВКонтакте», предобработки данных и выдачи рекомендаций требуются один раз за всё время жизни веб-приложения, поэтому их необходимо зарегистрировать с помощью метода `AddSingleton()`;

Для получения доступа к микросервису из контроллера веб-приложения используется метод `HttpContext.GetRequiredService<TType>()`, где `TType` – тип класса микросервиса. Метод создаст и вернёт экземпляр класса в зависимости от способа его регистрации.

Реализация регистрации микросервисов представлена в листинге 2.6

```
var vkApi = new VkApi();
services.AddSingleton(s => vkApi);
services.AddSingleton(s => new AnalyzingService(
    config["vkLogin"],
    config["vkPassword"],
    vkApi));
```

Листинг 2.6 – Регистрация микросервисов

2.4 РЕАЛИЗАЦИЯ МЕТОДА ВЫДАЧИ РЕКОМЕНДАЦИИ

Рекомендация о поступлении формируется с помощью метода латентно-семантического анализ, описанного в главе 1.5.

Метод подразумевает следующие действия:

1. Формирование терм-документной матрицы.
2. Сокращение числа переменных операцией сингулярного разложения терм-документной матрицы.
3. Вычисление метрики.

Код формирования терм-документной матрицы представлен в листинге

2.7

```
var docs = new List<Faculties>();
docs.AddRange(Faculties.Enumerate());
var body = users.SelectMany(u => u.GetWords()
    .Where(w => w != string.Empty && w.Length > 2));

var terms = users.SelectMany(u => u.GetWords())
    .Distinct()
    .ToArray();

var dtMat = new double[docs.Count, terms.Count()];
for (int i = 0; i < dtMat.GetLength(0); i++)
{
    IEnumerable<string> facultyBody;
    if (dtMat.GetLength(0) - 1 == i)
        facultyBody = user.GetWords();
    else
    {
        facultyBody = users
            .Where(u => u.Universities != null && u.Universities
                .Select(uni => uni.FacultyId)
                .Contains((int)docs[i]))
            .SelectMany(u => u.GetWords());
    }
    for (int j = 0; j < dtMat.GetLength(1); j++)
        dtMat[i, j] = facultyBody.Count(w => w == terms[j]);
}
```

Листинг 2.7 – Формирование терм-документной матрицы

В качестве инструмента для реализации операции сингулярного разложения используется фреймворк «Accord.NET». Это фреймворк для машинного обучения на платформе «.Net». Среди возможностей этого

фреймворка имеются методы, необходимые инструменты для реализации операции [12]. Код операции сокращения числа переменных представлен в листинге 2.8.

```
svd = new SingularValueDecomposition(dtMat);
var lowRankedMat = Matrix.Dot(
    svd.LeftSingularVectors,
    svd.DiagonalMatrix);

var indexModel = new Dictionary<Faculties, double[]>();
for (int i = 0; i < docs.Count; i++)
{
    var vector = new double[svd.Rank];
    for (int j = 0; j < vector.Length; j++)
        vector[j] = lowRankedMat[i, j];
    indexModel.Add(docs[i], vector);
}
```

Листинг 2.8 – Сокращение числа переменных

Затем, получив векторы для каждого института, вычислим метрики расстояния до каждого института. Код вычисления метрик представлен в листинге 2.9.

```
var target = indexModel[Faculties.Target];
var facs = FacultiesExtensions.EnumerateReal();
var result = new Dictionary<Faculties, double>();
foreach (var faculty in facs)
{
    var vector = indexModel[faculty];
    double sum = 0;
    for (int i = 0; i < svd.Rank; i++)
        sum += Math.Pow(vector[i] - target[i], 2);
    var range = Math.Sqrt(sum);
    result.Add(faculty, range);
}
result = result
    .OrderBy(r => r.Value)
    .ToDictionary(ks => ks.Key, vs => vs.Value);
```

Листинг 2.9 – Вычисление метрики

Результатом является словарь, в котором ключом является институт, а значением является величина метрики. Институт с наименьшей метрикой является институтом рекомендации о поступлении.

ГЛАВА 3. ОЦЕНКА КАЧЕСТВА РАБОТЫ СИСТЕМЫ

3.1 ОПИСАНИЕ МЕТОДА ОЦЕНКИ КАЧЕСТВА

Для оценки качества работы системы используется перекрёстная проверка (англ. Cross-Validation).

Это один из наиболее распространённых и используемых методов для проведения полноценного тестирования и оценки качества работы различных алгоритмов машинного обучения. Для независимой выборки данный метод позволяет получить несмещённую оценку вероятности ошибки в отличие от средней ошибки на обучаемой выборке, которая может являться смещённой оценкой вероятности ошибки из-за переобучения алгоритма. Другим достоинством данной процедуры является способность получения оценки вероятности ошибки алгоритма, при отсутствии специально разработанной для тестирования контрольной выборки.

В перекрёстной проверке исходный набор данных разбивается на k блоков, причём каких-то определенных рекомендаций по выбору числа блоков нет. Из k блоков один оставляется для тестирования модели, а остающиеся $k - 1$ блока используются как тренировочный набор. Процесс повторяется k раз, и каждый из блоков используется один раз как тестовый набор. Получаются k результатов, по одному на каждый блок, они усредняются или комбинируются каким-либо другим способом, и дают одну оценку. [13]

В данной работе перекрёстная проверка применяется со следующими ограничениями:

- число блоков $k = 10$,
- размер блоков примерно одинаков,
- в качестве результирующей оценки используется среднее арифметическое оценок каждого блока.

Для реализации перекрёстной проверки используется класс библиотеки «Accord.NET» CrossValidation.

Этот класс содержит статические методы для выполнения перекрёстной проверки.

Для создания валидатора модели перекрёстной проверки используется метод `CrossValidation.Create<TModel, TLearner<Tinput, TOutput>>()`, где `TModel` – тип модели машинного обучения, `TLearner` – тип обучающего алгоритма, который используется в `TModel`, `Tinput` – тип входных данных, `TOutput` – тип выходных данных. Метод возвращает валидатор, т.е. объект, который будет выполнять перекрёстную проверку и выдавать результат.

Этот объект имеет следующие важные свойства:

- `K`. Число блоков.
- `Learner`. Алгоритм обучения.
- `Loss`. Способ сравнения ожидаемой и настоящей модели.

Для вычисления результата проверки используется метод `CrossValidation.Learn(TInput data)`, где `data` – данные, для которых необходимо провести проверку. Метод возвращает объект `CrossValidationResult`, который содержит результаты проверки. Для того, чтобы получить оценку точности необходимо обратиться к свойству `Accuracy`, которое содержит значение типа `double`, характеризующее точность работы рекомендательной системы. Чем выше это значение, тем более точно система предоставляет рекомендации.

Реализация перекрёстной проверки представлена в листинге 3.1.

```
var crossValidation = CrossValidation.Create(  
    typeof(FetchedUser),  
    utmnRecco.Analyze(fetchedReader),  
    SupportVectorMachine<FetchedUser, Faculty>);  
crossValidation.K = 10;  
crossValidation.Learner = ((s) =>  
    new SequentialMinimalOptimization<Linear,  
        IEnumerable<FetchedUser>>());  
crossValidation.Loss = ((exp, act, p) => new  
    ZeroOneLoss(exp).Loss(act));  
  
var result = crossValidation.Learn(data);  
var accuracy = result.Accuracy;
```

Листинг 3.1 – Перекрёстная проверка

3.2 ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТА ОЦЕНКИ

Описанный в 3.1 метод был применён для оценки системы. В качестве выборки были взяты полные данные всех пользователей ТюмГУ за последние 5 лет. Размер выборки равен 8000.

Результат проверки равен 0,74.

Данный показатель является низким, что свидетельствует о том, что система недостаточно точно предоставляет рекомендации, оптимальный уровень точности для подобных рекомендательных систем является 0,95. [14]

Для улучшения точности предоставления рекомендации для описанного алгоритма необходимо улучшить предварительную обработку данных. Для этого необходимо предпринять следующие действия:

- Улучшить лемматизацию, используя более точный лемматизатор.
- Использовать различные весовые коэффициенты для анкетных данных, записей пользователя и списка подписок.
- Использовать в качестве списка стоп-слов не только слова служебных частей речи, но и некоторые общеупотребительные слова.

ГЛАВА 4. ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Интерфейс сервиса состоит из трёх страниц.

На этапе сбора и обработки данных, когда пользователь ещё не может отправлять запросы на выдачу рекомендации, на странице выводится сообщение «Сервис запускается».

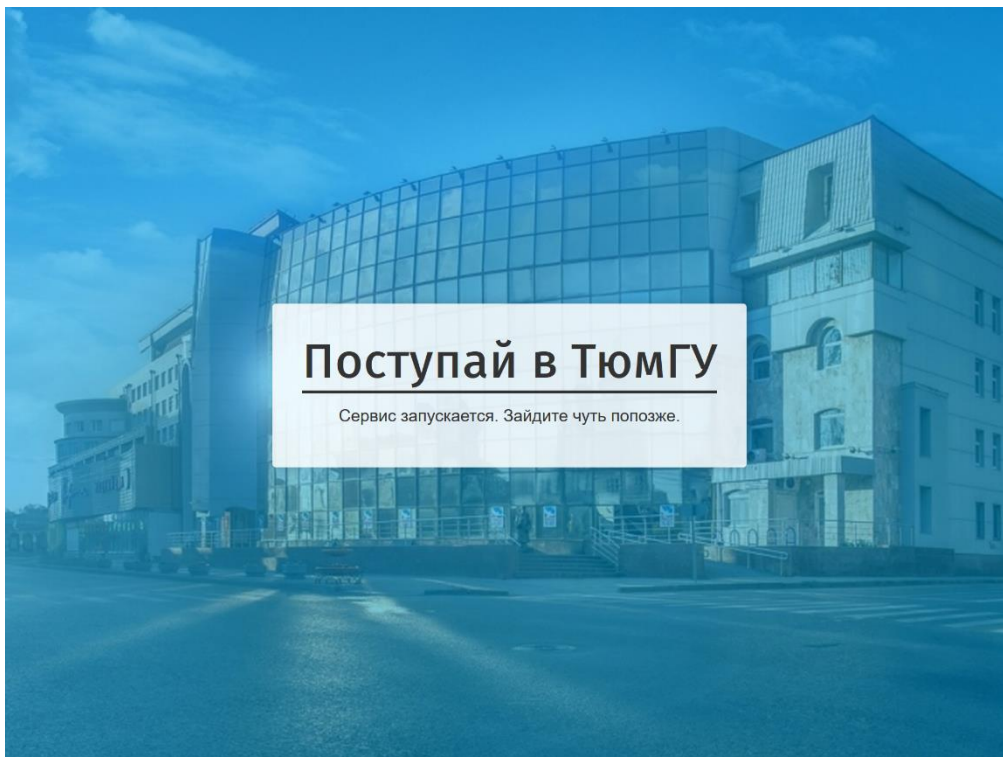


Рисунок 4.1 – Сообщение «Сервис запускается»

После того, как сервис закончил сбор и обработку данных и готов обрабатывать запросы на выдачу рекомендации, на странице отображается сообщение о просьбе вставить ссылку на свой профиль «ВКонтакте», текстовое поле для этой ссылки и кнопка «Поехали».

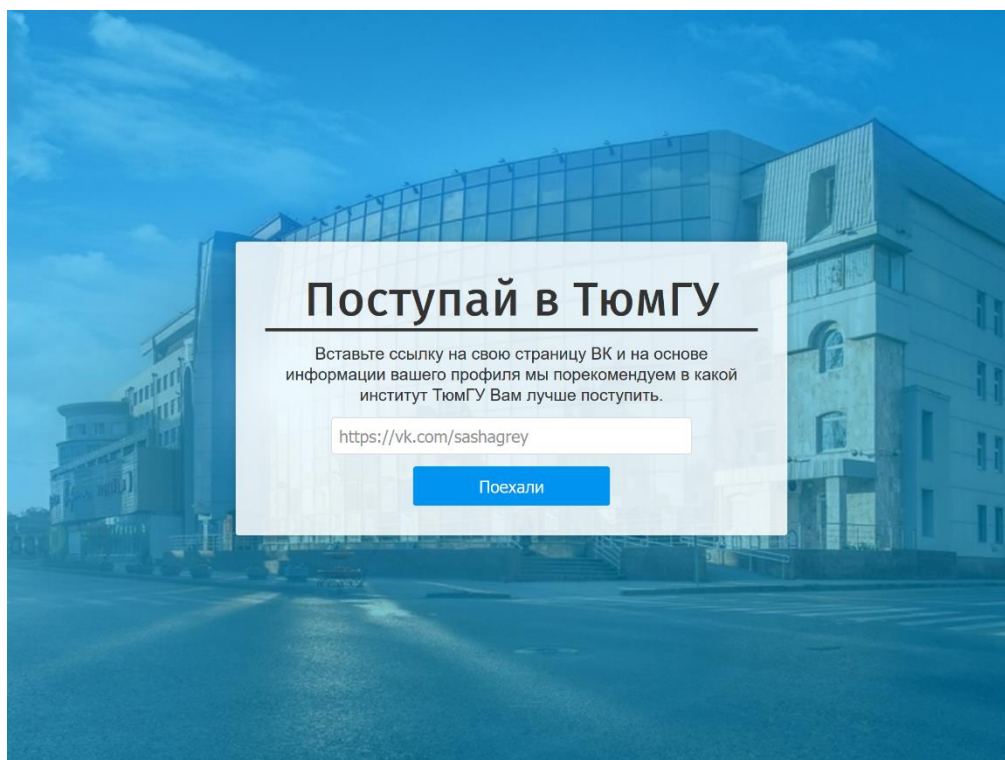


Рисунок 1.2 – Страница с полем ввода ссылки

После нажатия кнопки «Поехали» отправляется запрос на выдачу рекомендации для профиля по предоставленной ссылке. Затем, после окончания запроса, произойдёт перенаправление на страницу результатов. На странице результатов отображается однозначная рекомендация поступления в определённый институт ТюмГУ. Также в качестве отладочных данных приводится таблица результатов вычислений расстояний векторов интересов R .

Поступай в ТюмГУ

Василий Токтарев,
мы рекомендуем Вам поступить в
ИнХим

Наши вычисления

Институт	R
ИнХим	214.07
ФТИ	359.01
ИНБИО	419.97
ИФК	551.56
ИИПН	606.12
ИнЗем	677.80
ИМикН	1351.48
ИПиП	1752.09
ИФИЖ	2174.16
ФЭИ	2779.87
ИГиП	2963.60

Рисунок 4.3 – Страница результатов

ЗАКЛЮЧЕНИЕ

В данной работе был предложен метод разработки рекомендательной системы поступления в ТюмГУ на основе профиля абитуриента в социальной сети Вконтакте.

Перед реализацией поставленной цели нам необходимо было изучить уже существующие технологии. Так, в теоретической части настоящей работы были рассмотрены различные типы рекомендательных систем, технологии сбора данных о пользователях социальных сетей, а также алгоритмы выдачи рекомендаций о поступлении на основе методов обработки естественного языка, их преимущества и недостатки. Примерами подобных алгоритмов могут послужить латентно-семантический анализ и латентное размещение Дирихле. Рабочим алгоритмом для данного исследования стал латентно-семантический анализ, обладающий рядом преимуществ. Так, мы выяснили, что именно ЛСА является оптимальным методом выявления латентных зависимостей внутри множества документов, а также способен частично снимать полисемию и омонимию. Подробное описание метода приведено в соответствующей подглаве и содержит описание алгоритма, состоящего из трех шагов, а именно составления терм-документной матрицы, сокращения числа переменных операцией сингулярного разложения терм-документной матрицы, а также вычисления метрик.

Следующим шагом стало рассмотрение способов предварительной обработки данных. Так, нами были описаны такие способы, как лемматизация, стэмминг, отбрасывание стоп-слов, отбрасывание редких слов, а также выделение ключевых фраз. Поскольку стэмминг плохо подходит для русского языка, в данном исследовании используются лемматизация, отбрасывание стоп-слов и выделение редких слов.

Также нами были выдвинуты требования к реализации, определены признаки, по которым будет осуществляться выдача рекомендации и форма самой рекомендации.

Во второй главе настоящей работы подробно описана программная реализация, а именно реализация сбора данных и пользователе, предварительной обработки данных, сервиса и метода выдачи рекомендации.

Кроме того, нами была представлена оценка качества работы рекомендательной системы, для которой мы использовали метод перекрестной проверки, подробно описанный в третьей главе. Оценка производилась на основе выборки из полных данных всех пользователей ТюмГУ за последние пять лет. Результат проверки равен 0,74. Подобный результат является недостаточным, и мы предлагаем несколько способов улучшения результата, среди которых улучшение лемматизации, использование различных весовых коэффициентов для различных типов данных о пользователе, а также использование в качестве списка стоп-слов не только слов служебных частей речи, но и определенных общеупотребительных слов.

Таким образом, в соответствии с поставленной целью, результатом данной работы стала разработанная рекомендательная система. На основании перекрестной проверки нами был сделан вывод о том, что система недостаточно точна, и были даны рекомендации для повышения точности рекомендаций. Задачи работы выполнены в полном объёме. Продолжением настоящего исследования может стать выполнение рекомендаций и усовершенствование системы, а сама система может быть применена на практике.

СПИСОК ЛИТЕРАТУРЫ

1. Recommender system [Электронный ресурс] // Wikipedia, the free encyclopedia: [сайт]. [2018]. URL: https://en.wikipedia.org/wiki/Recommender_system (дата обращения: 23.мая.2018).
2. А. А. Правиков В.А.Ф. Разработка рекомендательной системы с естественно-языковым интерфейсом на основе математических моделей семантических объектов // Информационные технологии в бизнесе. 2010. No. 4. P. 11.
3. Принципы работы рекомендательных механизмов Интернета. Часть 1. Введение в подходы и алгоритмы. [Электронный ресурс] // IBM developerWorks Россия : Ресурс IBM для разработчиков и IT профессионалов: [сайт]. [2014]. URL: <https://www.ibm.com/developerworks/ru/library/os-recommender1/> (дата обращения: 23.мая.2018).
4. Запросы к API | Разработчикам [Электронный ресурс] // VK Developers. Разработчикам.: [сайт]. (дата обращения: 25.мая.2018).
5. В.А. Минаев И.Д.К.И.А.К. Методы выявления латентной и негативной информации в текстовых документах // Интернет-журнал "Технологии техносферной безопасности". Sep 2016. No. 5.
6. Воронцов К.В. 2013. URL: <http://www.machinelearning.ru/wiki/images/2/22/Voron-2013-ptm.pdf> (дата обращения: 27.мая.2018).
7. Латентно-семантический анализ [Электронный ресурс] // Хабр: [сайт]. [2010]. URL: <https://habr.com/post/110078/> (дата обращения: 23.мая.2018).
8. Бондарчук Д.В. Использование латентно-семантического анализа в задачах классификации текстов по эмоциональной

окраске // Научные исследования -> информационные технологии, 2012. Р. 6.

9. Морфологический анализатор грамматического словаря и part-of-speech tagger [Электронный ресурс] // Компьютерная грамматика русского языка: лексика, морфология, синтаксис: [сайт]. [2018]. URL: <http://www.solarix.ru/> (дата обращения: 24.06.2018).
10. Преимущества и недостатки микросервисной архитектуры [Электронный ресурс] // Записки программиста: [сайт]. [2014]. URL: <https://eax.me/micro-service-architecture/> (дата обращения: 25.мая.2018).
11. Введение в ASP.NET Core [Электронный ресурс] // Microsoft Docs: [сайт]. [2018]. URL: <https://docs.microsoft.com/ru-ru/ASPNET/Core/?view=aspnetcore-2.1> (дата обращения: 17.06.2018).
12. Accord.NET Machine Learning Framework [Электронный ресурс] // Accord.NET Machine Learning Framework: [сайт]. URL: <http://accord-framework.net/> (дата обращения: 17.06.2018).
13. mehanizator. Кросс-валидация (Cross-validation) [Электронный ресурс] // Long/Short: [сайт]. [2016]. URL: <http://datascientist.one/cross-validation/> (дата обращения: 17.06.2018).
14. С.А.Амелькин. Оценка эффективности рекомендательных систем 2012. URL: <http://ceur-ws.org/Vol-934/paper44.pdf> (дата обращения: 12.06.2018).
15. Обработка естественного языка: краткий ликбез для разработчика [Электронный ресурс] // ИТ в Беларуси | dev.by: [сайт]. [2016]. URL: <https://dev.by/lenta/main/obrabotka-estestvennogo-yazyka-kratkiy-likbez-dlya-razrabotchikov> (дата обращения: 23.мая.2018).