

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное
учреждение высшего образования
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт математики, механики и компьютерных наук
имени И. И. Воровича

Направление подготовки
02.03.02 – Фундаментальная информатика
и информационные технологии

РЕАЛИЗАЦИЯ РЕКОМЕНДАТЕЛЬНОГО ВЕБ-СЕРВИСА
НА ПЛАТФОРМЕ DJANGO

Выпускная квалификационная работа
на степень бакалавра

Студента 4 курса
Д. Е. Литовченко

Научный руководитель:
доц., к.ф.-м.н. М. Э. Абрамян

Допущено к защите:
руководитель направления ФИИТ _____ В. С. Пилиди

Ростов-на-Дону
2017

Содержание

Введение	3
1. Постановка задачи	4
2. Программный интерфейс пользователя	5
2.1. Стартовая страница сервера	6
2.2. Модуль Места	7
2.3. Модуль Рекомендации	8
2.4. Модуль Личный кабинет	10
3. Реализация сервиса	11
3.1. Проблема сбора данных и её решение	12
3.2. Контентная фильтрация	15
3.3. Коллаборативная фильтрация.	16
3.4. Проблема холодного старта и её решение	21
Заключение	24
Приложение А. Структура проекта с пояснениями	24

Введение

Рекомендательные системы стали неотъемлемой частью пользовательских приложений и сервисов, получили широкое распространение в последнее время. Рекомендательные системы - это программы и сервисы, которые пытаются определить, что нужно в данный момент времени пользователю, и предоставить ему это (или порекомендовать, откуда и произошло название). Объекты, которые рекомендуются, могут быть различны, это зависит от характера системы. Чаще всего такими объектами, находящимися в центре внимания подобных сервисов, становятся книги, фильмы, музыка, новости или веб-сайты.

Два главных подхода, которые используются в любых рекомендательных системах - это контентная и коллаборативная фильтрации. Трактовки данных понятий столь же различны, сколько и подходы к их реализации. Контентная фильтрация - это рекомендательная система, основанная на контенте, хранящемся в данных системы, и его характеристиках, свойствах, параметрах. Коллаборативная фильтрация - это рекомендательная система, опирающаяся на поведение пользователя в прошлом (информация о покупках или оценках) и поведение похожих на него пользователей. Здесь не имеет значения, с какими объектами происходит работа, но при этом могут учитываться и неявные характеристики.

Во время работы рекомендательные системы собирают данные о пользователях, при этом используется сочетание явных и неявных методов. Явный сбор данных осуществляется с помощью запроса у пользователя оценки объекта по данной шкале, предъявления пользователю двух объектов с вопросом о том, какой из них лучше, предложения создать список объектов, которые нравятся пользователю. Неявный сбор данных происходит с помощью наблюдения за тем, что просматривает пользователь, ведения записей о поведении пользо-

вателя в сети, отслеживание содержимого компьютера пользователя. Впрочем, используются и другие способы.

1. Постановка задачи

Основной целью данной работы стала реализация общедоступного во всемирной сети рекомендательного веб-сервиса кафе и ресторанов в городе Ростове-на-Дону. Данная задача повлекла за собой исследование новейших подходов к решению известных проблем и задач, возникающих при реализации рекомендательных веб-систем, а также создание собственных методов обработки информации и анализа данных.

Важным требованием к данному веб-сервису на этапе постановки задачи являлось присутствие контентной и колаборативной фильтраций, а также комбинирование явных и неявных методов сбора данных. При этом было немаловажно наличие личного кабинета пользователя, релевантной базы данных объектов для просмотра и рекомендаций, интуитивно понятного и современного интерфейса пользователя, который адаптивно подстаетается под стационарные и мобильные устройства. Отличительной особенностью данного сервиса от уже существующих предполагается новый способ решения классической проблемы холодного старта системы с помощью сбора, анализа и кластеризации данных, полученных неявно с помощью актуальных данных из социальных сетей.

Таким образом, на этапе постановки задачи были изучены существующие сервисы, предоставляющие возможности рекомендательных систем. Были рассмотрены как крупные сервисы (Amazon, Netflix, IMDB и другие), так и менее значимые региональные системы (IVI, Кинопоиск и другие). Также был произведен поиск старых и новых опубликованных статей по данной тематике. В процессе работы были изучены различные варианты пользовательских интерфейсов для

предоставления рекомендаций, доступные методы сбора и обработки данных разных пользователей и различные проблемы. Так были выявлены слабые места таких систем, что дало возможность исследовать их более детально, найти новые способы улучшения, получить результаты и сравнить их недостатки и преимущества с уже существующими системами.

2. Программный интерфейс пользователя

Рекомендательная система, описанная в данной работе, представляет из себя веб-сервис, который был запущен на локальном сервере разработчика в период разработки и отладки, а затем опубликован на общедоступном бесплатном сервере Pythonanywhere.

Таким образом, пользовательский интерфейс представляет собой набор HTML-страниц с динамическим содержимым, которые логически разделены между собой. Отображение некоторых данных на стороне пользователя и некоторые элементы интерфейса формируются при помощи скриптов языка JavaScript (и, в частности библиотеки JQuery), а также таблицы стилей CSS для формирования современного адаптивного дизайна сайта. Любая страница состоит из трех условных частей:

1. Навигационный бар (header), который позволяет легко перемещаться между блоками (модулями) сервиса.
2. Тело страницы (body) - здесь расположен весь основной статический контент и динамический контент.
3. Нижняя часть страницы ('подвал', footer) - в ней отображается вся контактная информация (телефоны, адреса, ссылки) и авторская информация (имена, фамилии, должности)

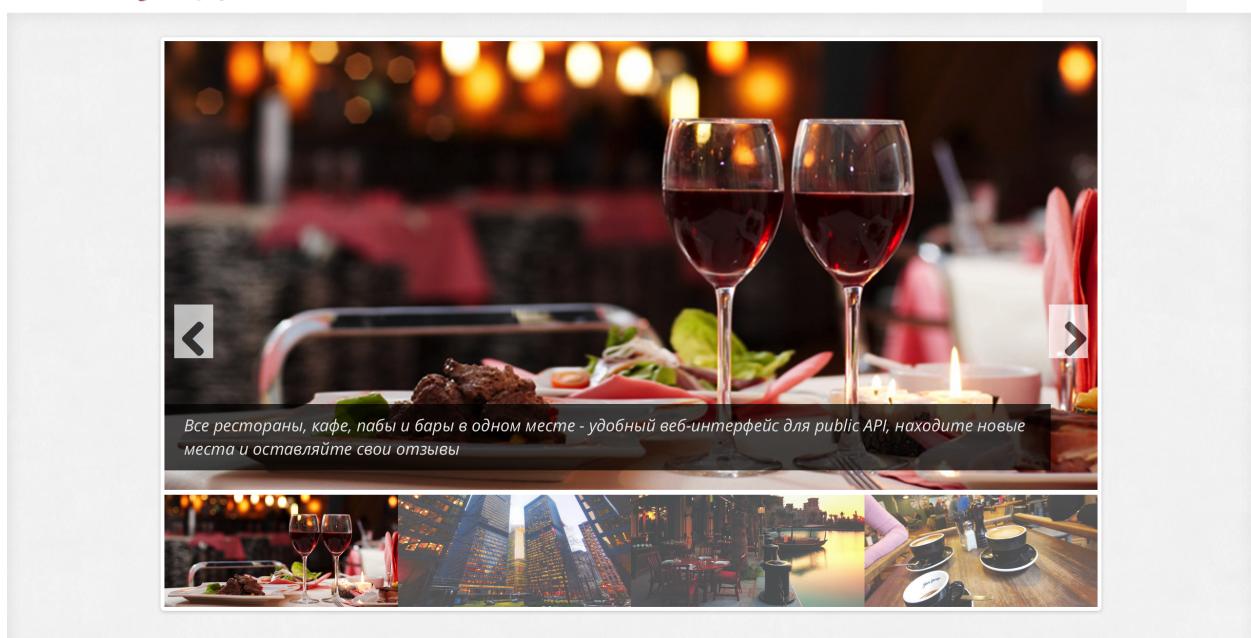


Рисунок 1 – Скриншот 1 стартовой страницы

2.1. Стартовая страница сервера

Стартовая страница - это страница, на которую попадает пользователь, когда заходит на сайт или успешно регистрируется в системе / входит в систему под своими логином и паролем. В навигационном баре отображается различное число доступных ссылок на другие модули, в зависимости от того, произведен ли в данный момент вход в систему или нет (есть приватные модули, доступ к которым есть только у авторизованных пользователей).

На главной странице возможен переход к модулю "Вход в систему состоящий из подмодулей "Регистрация"(позволяющий создать нового пользователя в системе) и "Авторизация"(позволяющий существующему пользователю войти в систему). Расположенный далее элемент JQuery Slider позволяет пользователям в форме слайд-шоу наглядно познакомиться с основным функционалом сервиса и его особенностями.

*Недавно
добавленные*



МЕРКУРИЙ

5.0



ВЕГАС

4.4

*Фото
отсутствует*

РАЙ

3.0



РЕСТОРАН-БАР КОЛЯДА

4.6

О ПРОЕКТЕ Сервис создан в качестве иллюстрации функционала к исследованию, проведенным в дипломной работе. Узнать больше	E-MAIL ПОДПИСКА Введите e-mail для подписки: <input type="text" value="Email..."/> Подписаться	ФОТОГРАФИИ	КОНТАКТЫ Научный руководитель: Абрамян М.Э. Адрес: г. Ростов-на-Дону, ул. Мильчакова 8а VK: vk.com/dmitritov4 Email: dmitriyonline222@gmail.com
---	--	-------------------	--

Рисунок 2 – Скриншот 2 стартовой страницы

Завершает стартовую страницу блок последних добавленных в базу данных объектов, за которым следует footer страницы.

2.2. Модуль Места

С помощью навигационного бара осуществляется переход к модулю 'Места'. В данном разделе осуществляется постраничное отображение всех объектов, хранящихся в базе данных, в порядке убывания их среднего рейтинга. Присутствует возможность перейти по ссылке к детальному описанию всех характеристик объекта. При наведении указателя мыши на объект с помощью средств JavaScript появляется дополнительная графическая и текстовая информация о параметрах объекта. Слева от секции с объектами находится форма контентной фильтрации - она позволяет указывать желательные характеристики объектов и формировать выборку критериев при поиске наиболее подходящих. Многие поля формы допускают множествен-

ное выделение. По умолчанию объекты отсортированы по убыванию значения их параметра "Рейтинг-", который является вещественным числом от 1.0 до 5.0 и отражает рейтинг заведения по версии сервиса Google.

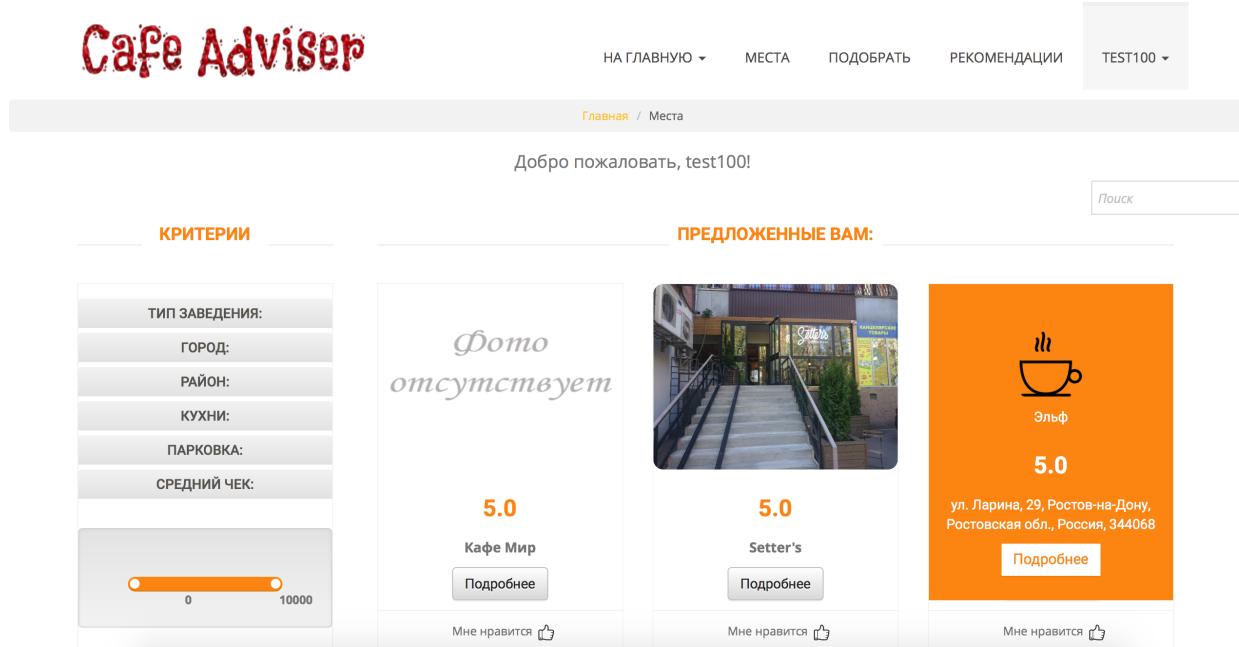


Рисунок 3 – Скриншот модуля Места

2.3. Модуль Рекомендации

"Рекомендации" - это модуль, полностью отражающий работу колаборативной фильтрации по поиску пользователей, наиболее похожих по поведению на текущего авторизованного в системе пользователя.

Главной частью этого модуля с динамически формируемой информацией является сгруппированная секция рекомендуемых объектов с основной информацией о них. Также есть возможность перейти по ссылке на более подробное описание объекта, которое хранится в базе данных.

Уважаемый test100! На основе Ваших предпочтений мы сформировали список заведений, которые могут Вам понравиться.

ПРЕДЛОЖЕННЫЕ ВАМ:



Рисунок 4 – Скриншот модуля Рекомендации

This screenshot shows the detailed view of a specific establishment. It includes a large image of the restaurant's interior, basic information like ID and quality rating, and a section for similar establishments.

Drago Steakhouse
ID: 103
Качество: 4.9
★★★★★
Тип заведения: ресторан
Город: Ростов-на-Дону
Полный адрес: ул. Суворова, 75, Ростов-на-Дону, Ростовская обл., Россия, 344022
Район: Кировский район
Кухни: европейская, восточная, средиземноморская,
Средний чек: 3000
Наличие парковки: есть

ПОХОЖИЕ ЗАВЕДЕНИЯ:

Рисунок 5 – Скриншот 1 страницы деталей объекта

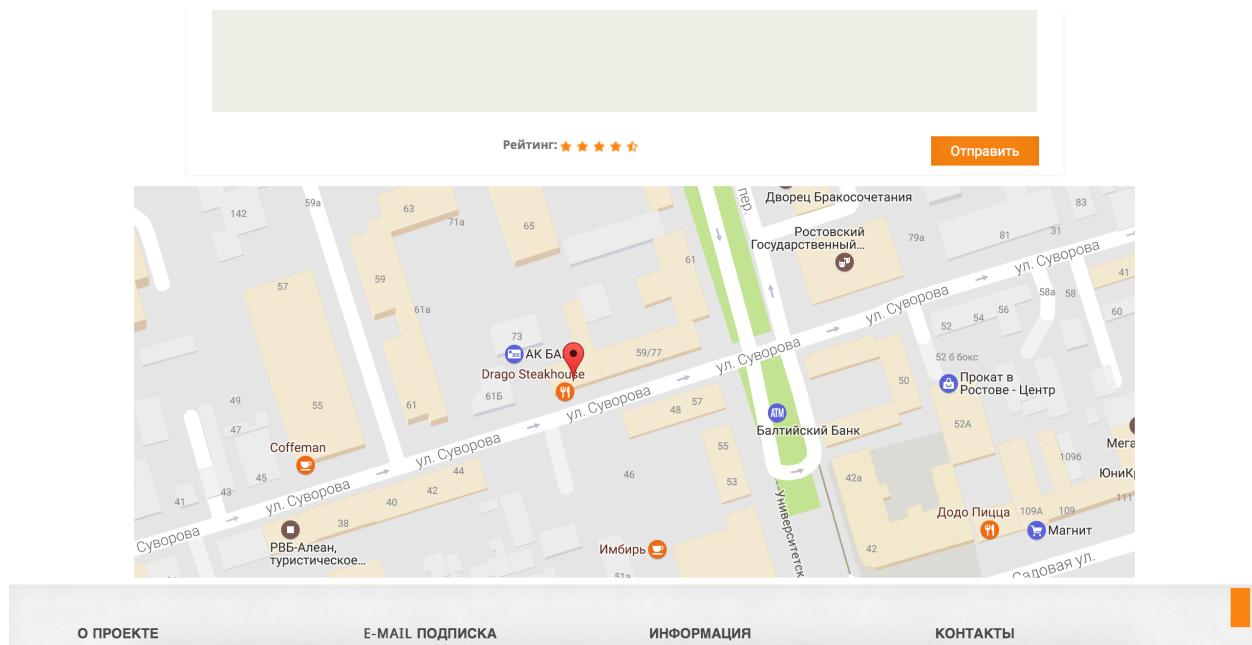


Рисунок 6 – Скриншот 2 страницы деталей объекта

2.4. Модуль Личный кабинет

Личный кабинет - это специальный раздел сайта, доступный только для пользователя, авторизованного в системе. Сервис предоставляет возможность текущему авторизованному пользователю просматривать актуальную информацию о себе (дата регистрации, имя, фамилия, возраст, город и другое), менять её, а также просматривать список объектов, которые отмечены пользователем как понравившиеся. Они являются своеобразными закладками пользователя, а также они используются рекомендательной системой для построения коллaborативной фильтрации.

Модуль Личный кабинет содержит разделы "Профиль" и "Любимые места" функционал которых описан выше. Скриншот 1 демонстрирует интерфейс раздела "Профиль". Скриншот 2 демонстрирует интерфейс раздела "Любимые места" где наглядно отображают-

2

test100

(test100@mail.ru)

Дата регистрации: March 10, 2017, 6:34 p.m.

Имя: Дмитрий

Фамилия: Дмитриев

Возраст: 21

Город: Ростов-на-Дону

Пол: М

О ПРОЕКТЕ Сервис создан в качестве иллюстрации функционала к исследованиям, проведенным в дипломной работе. Узнать больше	E-MAIL ПОДПИСКА Введите e-mail для подписки: <input type="text" value="Email..."/> <input type="button" value="Подписаться"/>	ФОТОГРАФИИ	КОНТАКТЫ Научный руководитель: Абрамян М.Э. Адрес: г. Ростов-на-Дону, ул. Мильчакова 8а VK: vk.com/dmitrov4 Email: dmitriyonline222@gmail.com
--	---	-------------------	--

Рисунок 7 – Скриншот 1 модуля Личный кабинет

ся некоторые параметры объектов, которые были идентифицированы пользователем в качестве понравившихся.

3. Реализация сервиса

Веб-сервис CafeAdviser предоставляет два типа рекомендаций: контентную (по составленному пользовательскому запросу) и коллаборативную (на основе поведенияя пользователя в прошлом и поведении похожих на него пользователей). Для реализации данного сервиса были использованы: среда разработки Sublime Text 2 (легковесный редактор кода с подсветкой синтаксиса), Python версии 3.4, веб-фреймворк Django версии 1.8 (а также набор пакетов предназначенных для работы с ним). Для работы с данными задействована свободная реляционная система управления базами данных MySQL 5.7.

Отличительной особенностью реализации данной работы стало активное взаимодействие со сторонними крупными сервисами,

Уважаемый test100! Список заведений, которые Вы отметили как понравившиеся:

ПОНРАВИВШИЕСЯ ВАМ:



5.0

Setter's



5.0

St.Tropez pool cafe



4.9

Drago Steakhouse

[Подробнее](#)

[Подробнее](#)

[Подробнее](#)



Рисунок 8 – Скриншот 2 модуля Личный кабинет

предоставляющими открытые API, в процессе автоматического формирования базы данных. Отдельно нужно упомянуть Open API крупнейшей социальной сети ВКонтакте, созданной Павлом Дуровым в 2006 году. Open API - это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу. Синтаксис запросов и тип возвращаемых ими данных строго определены на стороне самого сервиса. Данные, полученные в результате работы с API VK, используются в алгоритме решения проблемы холодного старта коллаборативной фильтрации.

3.1. Проблема сбора данных и её решение

В результате реализации сервиса "CafeAdviser" был обнаружен ряд проблем, свойственных подобным системам. Так, проблема сбора данных заключается в том, что все данные о характеристиках и параметрах объектов, которые будут храниться в базе данных системы, невозможно найти, задействовав только один информационный ис-

точник. Решением стало формирование собственной базы данных из нескольких крупных источников с помощью открытых API. Это, например, сервисы Google, Yandex, 2GIS. Использовалась и комбинация запросов к API. Под комбинацией запросов здесь понимается цепочка обработки данных, в которой на вход запросу к API одного сервиса подается запрос, а ответ от этого сервиса обрабатывается определенным образом и передается на вход запросу к API уже другого сервиса, получая следующий ответ. Последний ответ в такой цепочке после обработки является результатом всей вычисляющей функции.

Например, у рекомендуемых объектов в данной системе есть такой параметр, как "Район города", который должен храниться в базе данных в качестве строки (string) и представлять из себя привычное название района города (Октябрьский, Кировский, Ленинский и т.д.). Ни один существующий открытый сервис не предоставляет информацию о районе города по названию заведения или улицы. Для решения данной проблемы был использован метод обратного геотаргетинга: сначала выполняется http-запрос к Google, где в качестве параметра передана строка с поисковым запросом (в данном случае, адрес из параметра объекта "Адрес", данный в произвольном виде), в ответ на http-запрос сервис возвращает JSON-ответ, из которого извлекаются GPS-координаты (широта, долгота) заведения, обрабатываются и передаются параметром в http-запросе к Yandex API. Yandex API отправляет JSON-ответ, из которого можно извлечь название района (в виде строки) и записать в базу данных MySQL. Часть кода функции, реализующей данное поведение, размещена в листинге ниже.

В данном отрывке кода происходит циклический обход всех объектов, полученных в результате запроса к сервису Google, в результате чего параметр "Адрес" (строка) объекта, возвращенный в ответе, обрабатывается с помощью функции transliterate, которая переводит каждый кириллический символ строки либо в одиночный латинский символ, либо в соответствующую данному кириллическому символу биграмму. Данная функция находится в модуле вспомогательных функ-

ций проекта. Таким образом, вычисленный результат работы функции transliterate передается в качестве параметра в GET-запрос к API сервиса Yandex, который, в свою очередь, отправляет JSON-ответ.

Листинг 3.1. Часть кода функции для получения параметра "Адрес" объекта с помощью обратного геотаргетинга

```
my_key = settings.GOOGLE_PLACES_API_KEY
google_places = GooglePlaces(my_key)
query_result = google_places.text_search(query=q, radius=
    search_radius, language=lang.RUSSIAN, types=[types.
    TYPE_FOOD, types.TYPE_CAFE, types.TYPE_RESTAURANT])
for place in query_result.places:
    place.get_details()
    if not Cafe.objects.filter(name=place.name).exists():
        address_list = place.formatted_address.split(',')
        city1=address_list[2].strip()
        city2=address_list[1].strip()
        city3=address_list[3].strip()
        response = urlopen(u"https://geocode-maps.yandex.
            ru/1.x/?geocode=" + transliterate(place.
            formatted_address) + u'&format=json')
        reader = codecs.getreader("utf-8")
        data = json.load(reader(response))
        geolocation = data['response'][[
            'GeoObjectCollection']['featureMember'][0][
            'GeoObject']['Point']['pos']]
        reverse_response = urlopen(u"https://geocode-maps
            .yandex.ru/1.x/?geocode=" + geolocation + u'&
            format=json&kind=district')
        district_dict = json.load(reader(reverse_response
            ))
    try:
        district = district_dict['response'][[
            'GeoObjectCollection']['featureMember'][0][
            'GeoObject']['name']]
    except IndexError:
        continue
```

Данное значение затем записывается вместе с остальными характеристиками в базу данных.

3.2. Контентная фильтрация

Контентная фильтрация, реализованная в рамках данного сервиса, является не обычным поиском по критериям. Это взвешенный поиск, результатом которого является вычисление значения функции-свертки.

Допустим, есть матрица C размерности $(m \times n)$, где m - количество объектов в системе, а n - количество параметров объекта. Элементы такой матрицы обозначим $c_i^j \in C, i = 1..m, j = 1..n, c_i^j \geq 0$. Каждая строка такой матрицы представляет набор параметров i -го объекта.

Также дан вектор параметров $P = \{p_1, \dots, p_n\}, p_i \geq 0, i = 1..n$, который динамически генерируется по запросу пользователя в ходе работы системы.

Дан вектор ценности параметров $W = \{w_1, \dots, w_n\}, w_i \geq 0, i = 1..n$. Он определяет вес i -го параметра.

Определим функции $fit_j(x, y)$ следующим образом ($x \in C, y \in P$ - параметры, $j = 1..n$):

$$fit_j(x, y) = \begin{cases} 1 & \text{если } x \text{ попадает в диапазон допустимых значений } y \\ 0 & \text{если } x \text{ не попадает в диапазон допустимых значений } y \end{cases}$$

Результатом называется вектор значений R , каждый компонент вектора является вычислением функции-свертки r , причем $r_i = \sum_{j=1}^n (w_j * fit_j(c_{ij}, p_j))$

Таким образом, для вычисления результирующего вектора необходимо лишь подобрать компоненты целочисленного вектора W так, чтобы свертка субъективно давала подходящий результат.

После подбора и тестирования были получены следующие результаты:

- kindweight = 70 - вклад параметра "Вид заведения"
- priceweight = 30 - вклад параметра "Средний чек"
- areaweight = 15 - вклад параметра "Район заведения"
- cuisineweight = 15 - вклад параметра "Кухни"
- parkingweight = 5 - вклад параметра "Наличие парковки"
- и так далее

Именно таким образом в рамках данной работы была реализована контентная фильтрация, представляющая собой взвешенный поиск по объектам, хранящимся в базе данных на сервере.

3.3. Коллаборативная фильтрация.

Коллаборативная фильтрация (англ. collaborative filtering) — это один из методов построения прогнозов (рекомендаций) в рекомендательных системах, использующий известные предпочтения (или, другими словами, оценки) группы пользователей для прогнозирования неизвестных предпочтений другого пользователя. Данный метод в рамках работы был реализован классическим способом с использованием различных метрик для определения расстояния между векторами.

Алгоритм начинается с построения матрицы кросс-табуляции (*cross-tabulation matrix*) M размерности $n \times m$, где n - количество пользователей в системе, m - количество объектов. Каждый элемент (i, j)

такой матрицы показывает, нравится ли i -му пользователю j -ое заведение. Таким образом, каждый элемент матрицы M формируется по следующему правилу:

$$M_{ij} = \begin{cases} 1 & \text{если } i\text{-му пользователю нравится объект под номером } j \\ 0 & \text{если } i\text{- му пользователю не нравится объект под номером } j \end{cases}$$

Нужно отметить, что отсутствие оценки объекта (i, j) приравнивается ко второму случаю (i -му пользователю не нравится объект под номером j).

Для матрицы используются данные 80% пользователей, что уменьшает объем данных для вычислений в условии ограниченных вычислительных ресурсов на сервере. Этот метод сокращения объема вычислений называется *sampling*. Также производится очистка сигнала от шума (*noise reduction*), к примеру, из полученных данных перед вычислениями удаляются все нулевые векторы.

ID клиента	Кафе 1	Кафе 2	Кафе 3
154	1	1	1
155	1	0	0

Рисунок 9 – Пример небольшой части полученной матрицы крест-табуляции

С помощью матрицы M можно оценить похожесть пользователей друг на друга, сравнив схожесть строк матрицы с помощью различных метрик.

Транспонируем M и обозначим полученную матрицу как M^T . Данная матрица представляет уже набор характеристических векторов объектов, хранящихся в базе данных, а не пользователей.

ID кафе	клиент 1	клиент 2
Кафе 1	1	1
Кафе 2	1	0
Кафе 2	1	0

Рисунок 10 — Пример небольшой части транспонированной матрицы кросс-табуляции

Далее необходимо установить схожесть строк данных матриц между собой. Для этого используются различные метрики. Экспериментально был выбран метод, сочетающий две распространенные метрики: расстояние Танимото T ($\frac{c}{a+b-c}$) и Евклидово расстояние E ($\sqrt{\sum_1^n (x_i - y_i)^2}$). Результатом метода является $(T+E)/2$. Значения двух метрик складываются, а затем делятся пополам. Таким образом, мы получаем их среднее значение, которое дает более точный результат, чем каждая из метрик по отдельности.

	Кафе 1	Кафе 2	Кафе 3
Кафе 1	0	0.01345	0.08345
Кафе 2	0.01345	0	0.7504
Кафе 3	0.08345	0.7504	0

Рисунок 11 — Пример небольшой части матрицы коллaborаций

Листинг 3.2. Функция создания матрицы M

```
def create_user_vectors(userrecords):
    statistic = []
    for record in userrecords:
        statistic_user = np.zeros(Cafe.objects.all().
            order_by("-id")[0].id + 1)
        statistic_user[0] = record.user_id
        for liked_cafe in record.liked.all():
            statistic_user[liked_cafe.id] = 1
        statistic.append(statistic_user)
    numpy_statistic = np.array(statistic)
    k = 1
    answer = numpy_statistic
    final_answer = []
    for k in range(0, len(answer)-1):
        if (np.count_nonzero(answer[k]) > 1):
            final_answer.append(answer[k])
    answer = np.array(final_answer)
    np.savetxt("user_vectors.csv", answer.astype(int),
        fmt='%i', delimiter=",")
    print(answer)
    return answer
```

В результате приведенных вычислений получаем ещё одну матрицу - матрицу коллокваций C , содержащей вещественные коэффициенты, в которой элемент матрицы (i, j) показывает насколько объект под номером i похож на объект под номером j .

В приведенных фрагментах кода создаются матрицы M (матрица, характеризующая пользователей, зарегистрированных в системе) и C (матрица, показывающая схожесть зарегистрированных в системе пользователей между собой).

Для обработки и хранения матриц, а также расширенной обработки списков используется известный математический пакет NumPy для Python 3. Это расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц, вместе с большой биб-

лиотекой высокоуровневых математических функций для операций с этими массивами. Основным объектом NumPy является однородный многомерный массив. Это таблица элементов, всех одного типа, индексированных последовательностями натуральных чисел. Под многомерностью массива понимается то, что у него может быть несколько измерений или осей.

Листинг 3.3. Функция создания матрицы С

```
def create_user_matrix():
    userrecords = UserSettings.objects.all()
    cafes = create_user_vectors(userrecords)
    cafe_ids = cafes[:,0]
    num_of_cafes = len(cafes)
    cafe_matrix = np.zeros((num_of_cafes, num_of_cafes),
                           np.float32)
    np.fill_diagonal(cafe_matrix, 0)
    for i in range(0, len(cafes)):
        for j in range(i + 1, len(cafes)):
            common = 0
            count_sum = 0
            for k in range(0, len(cafes[i])):
                if cafes[i][k] == cafes[j][k]:
                    common = common + 1
            count_sum = count_sum + math.pow(cafes[i]
                                              [k] - cafes[j][k], 2)
            check = str(i) + " " + str(j) + " - " + str(
                common)
            tanimoto_coef = common / (len(cafes[i]) + len(
                cafes[j]) - common)
            euclid_coef = 1 - (math.sqrt(count_sum) / len(
                cafes[i]))
            res = str(i) + " " + str(j) + " - " + str(
                euclid_coef)
            cafe_matrix[i][j] = (tanimoto_coef + euclid_coef) /
                2
    return (cafe_ids, cafe_matrix)
```

3.4. Проблема холодного старта и её решение

Под проблемой холодного старта рекомендательных систем понимается высокая разреженность данных, которая появляется либо при старте сервиса, когда количество пользователей, зарегистрированных на нем, близко к нулю, либо сразу после регистрации нового пользователя в системе, когда от него ещё не было никакой активности, а количество обработанных им объектов близко к нулю.

После проведения исследования существующих методик решения этой классической проблемы рекомендательных систем, было решено использовать собственный способ - разбиение всех зарегистрированных пользователей на кластеры (группы) по демографическим и личностным признакам, что, несомненно, влияет на характер, темперамент человека и его предпочтения.

Решение было найдено за пределами существующей системы с помощью скрытого сбора данных из одного из крупнейших сервисов в Российской Федерации - социальной сети "ВКонтакте" (www.vk.com), используя API этого ресурса. Выполнение запросов к API выполняется не непосредственно через авторизацию на сервере, а с использованием механизма токенов, то есть необходимо получить ключ доступа `access_token`. Сервис "ВКонтакте" содержит наибольшее количество открытой информации о пользователях, характеризуя его с разных сторон. Используя API "ВКонтакте" ([`vkapi`]) и данные, указанные при регистрации в системе CafeAdviser, стало возможным найти страницу пользователя "ВКонтакте" с очень большой долей вероятности, получить необходимые данные, нормализовать их и проанализировать. После разбиение пользователей на заранее выбранное количество кластеров появляется возможность рекомендовать текущему пользователю какой-то объект из объектов, понравившихся случайно выбранному пользователю в той же демографической группе (кластере). При разбиении пользователей на группы были использованы следующие параметры: наличие высшего образования, наличие во-

енной службы, тип занятости (занятость на работе, студент, временно безработный и т.д.), возраст пользователя, количество подписчиков, музыкальные предпочтения (по жанрам).

Определение наиболее предпочтительного музыкального жанра происходит с помощью использования Наивного Байесовского Классификатора (Naive Bayes Classifier). Это простой вероятностный классификатор, основанный на применении Теоремы Байеса со строгими (наивными) предположениями о независимости.

В зависимости от точной природы вероятностной модели, наивные байесовские классификаторы могут обучаться очень эффективно. Во многих практических приложениях для оценки параметров для наивных байесовых моделей используют метод максимального правдоподобия. Другими словами, можно работать с наивной байесовской моделью, не веря в байесовскую вероятность и не используя байесовские методы.

Несмотря на наивный вид и, несомненно, очень упрощенные условия, наивные байесовские классификаторы часто работают намного лучше во многих сложных практических задачах, подобной определению музыкального жанра по входной строке с перечислением любимых групп и исполнителей. Достоинством наивного байесовского классификатора является малое количество данных для обучения, необходимых для оценки параметров, требуемых для классификации. Для обучения использован csv-файл с примерами жанров по возможным ключевым словам.

После нормализации векторов, представляющих данные пользователей из социальной сети "ВКонтакте" необходимо произвести их кластеризацию. Данная операция была проведена с использованием алгоритма K-Means ([[intuitLessons](#)]). Суть алгоритма состоит в выполнении следующей последовательности действий.

1. Выбрать количество кластеров k , которое является оптимальным для задачи (при разработке системы $k=5$, затем оно должно

но увеличиваться прямо пропорционально увеличению зарегистрированных пользователей).

2. Выбросить случайным образом в пространство наших данных k точек (т.н. центроидов).
3. Для каждой точки набора данных посчитать, к какому центроиду она ближе.
4. Переместить каждый центроид в центр выборки, которая была отнесена к этому центроиду.
5. Повторять последние два шага фиксированное число раз (до достижения заранее установленного максимального значения итераций), либо до тех пор пока центроиды не сойдутся (обычно это значит, что их смещение относительно предыдущего положения не превышает какого-то заранее заданного небольшого значения).

Пример части полученных данных, их нормализации и результат итоговой кластеризации приведен на скриншоте ниже.

```
Дмитрий Дмитриев
[4, {'first_name': 'Дмитрий', 'last_name': 'Дмитриев', 'uid': 159356741}]
[0, 0, 0, 1]
Дмитрий Литовченко
[1, {'military': [], 'occupation': {'name': 'Точка Кипения', 'type': 'work', 'id': 29913759}, 'last_name': 'Литовченко', 'music': 'Scorpions, Black Sabbath, Ozzy Osbourne, Ronnie James Dio, Aerosmith, Led Zeppelin, Guns N Roses, Slash, Red Hot Chili Peppers, Alter Bridge, Nirvana, Black Label Society, Eric Clapton, The Beatles, Deep Purple, Rainbow, Pink Floyd, Bon Jovi, Whitesnake, Joe Cocker, Bullet for my Valentine, Sunrise Avenue, System of a down, The Beatles, Green Day, The Offspring, Metallica, Skillet, Trivium, Disturbed, Maroon 5, Simple plan, Michael Jackson, J-rock: SID, Nightmare, The Gazette, Maximum the hormone. Гитарные Боги: Gary Moore, Randy Rhoads, Ritchie Blackmore, Slash, Zakk Wylde, Frank Zappa, Eric Johnson, John Petrucci, Steve Vai, Carlos Santana, Joe Satriani... Классика. Кое-что из русского рока: Чайф, Машина времени и другие... а так, слушаю всё, что нравится ;)', 'universities': [{'faculty_name': 'Институт математики, механики и компьютерных наук (бывш. Механико-математический)', 'city': 119, 'country': 1, 'education_form': 'Очное отделение', 'chair_name': 'Англоби и дискретной математики', 'name': 'ЮФУ (бывш. РГУ)', 'faculty': 2132, 'id': 3415, 'chair': 1885521, 'education_status': 'Студент (бакалавр)'}], 'first_name': 'Дмитрий', 'uid': 9209663}]
Дмитрий Литовченко
rock
[1, 0, 2, 1, 1]
Григорий Сетежев
[2, {'military': [], 'occupation': {'name': 'ЮФУ (бывш. РГУ)', 'type': 'university', 'id': 3415}, 'last_name': 'Сетежев', 'music': 'Машина Времени, Кипелов, Агата Кристи, Massive Attack, Вонобо, Ленинград, Ария, Сплин', 'universities': [{'faculty_name': 'Институт математики, механики и компьютерных наук (бывш. Механико-математический)', 'city': 119, 'country': 1, 'graduation': 2017, 'chair_name': 'Прикладной математики и программирования', 'name': 'ЮФУ (бывш. РГУ)', 'faculty': 2132, 'id': 3415, 'chair': 1885522, 'education_status': 'Студент (бакалавр)'}, {'faculty_name': 'Прикладной математики и программирования', 'name': 'ЮФУ (бывш. РГУ)', 'faculty': 2132, 'id': 3415, 'chair': 1885522, 'education_status': 'Студент (бакалавр)'}], 'first_name': 'Григорий', 'uid': 53098509}]
Григорий Сетежев
rock
[1, 0, 2, 2, 0]
Анатолий Абрамов
[1, {'military': [{"unit": "Связист запаса", "country_id": 1, "unit_id": 114631, "from": "2016"}], 'occupation': {'name': 'ЮФУ (бывш. РГУ)', 'type': 'university', 'id': 3415}, 'last_name': 'Абрамов', 'music': 'Всего по немножку. Ну "Wonderland Avenue - White Horse" Bob Sinclair Feat Stevie Edwards - World Hold On" "Status Quo - In The Army Now", "Sting - Shape of My Heart", "Universities": [{"faculty_name": "Институт математики, механики и компьютерных наук (бывш. Механико-математический)", "city": 119, "country": 1, "graduation": 2017, "chair_name": "Кафедра информатики и вычислительного эксперимента", "name": "ЮФУ (бывш. РГУ)", "faculty": 2132, "id": 3415, "chair": 6820, "education_status": "Студент (бакалавр)"}, {"faculty_name": "Очное отделение"}], 'first_name': 'Анатолий', 'uid': 56425143}]
Анатолий Абрамов
pop
[1, 1, 4, 2, 1]
[[0, 0, 0, 0, 1], [0, 0, 0, 0, 1], [0, 0, 0, 0, 0], [0, 0, 0, 0, 1], [0, 0, 0, 0, 1], [0, 0, 0, 0, 1], [0, 0, 0, 0, 1], [0, 0, 0, 0, 1], [1, 0, 2, 1, 1], [1, 0, 2, 2, 0], [1, 1, 4, 2, 1]]
[0 0 4 0 0 0 0 1 3 2]
```

Рисунок 12 – Пример части полученных данных (используются только некоторые параметры), их нормализации и итоговой кластеризации. Разбиение на 5 кластеров.

Заключение

Итогом данной работы стало создание рекомендательного веб-сервиса с использованием новых подходов при поиске ответов на ключевые вопросы подобных систем. Были проведены исследования, позволившие по-новому взглянуть на проблему холодного старта коллаборативных рекомендательных систем и классические проблемы поиска, сбора и обработки данных.

Особенностями системы являются

- собственная контентная фильтрация, представленная взвешенным поиском, где результатом является вектор, в котором каждый компонент получается путем вычисления значения функции-свертки
- новое решение проблемы «холодного старта» – скрытый поиск использования открытых данных из крупнейшей социальной сети в Российской Федерации для разбиения на демографические группы (кластеры пользователей) и исследование наилучших методов подобной кластеризации
- формирование собственной базы данных на сервере путем получения информации из нескольких источников при помощи открытых API крупных сервисов и комбинации этих данных

Приложение А. Структура проекта с пояснениями

- CafeAdviser (директория проекта)
 - __init__.py
 - urls.py (роутер проекта)
 - settings.py (опции, конфигурации и настройки проекта)

- wsgi.py
- cafe (директория приложения Cafe - непосредственная функциональная часть системы)
 - __init__.py__
 - urls.py (роутер приложения)
 - admin.py (контроллер административной панели)
 - views.py (контроллеры приложения)
 - models.py (модели приложения)
 - forms.py (формы приложения)
 - apps.py
 - myutils.py (вспомогательные функции для контроллеров)
 - tasks.py (функции, предназначенные для циклического выполнения с указанной периодичностью)
 - migrations (миграции приложения)
 - templates (шаблоны приложения)
 - static (статические файлы приложения: css, js, images и другое)
- login (директория приложения Login - обработка данных пользователей, регистрация, вход в систему)
 - __init__.py__
 - urls.py (роутер приложения)
 - admin.py (контроллер административной панели)
 - views.py (контроллеры приложения)

- models.py (модели приложения)
 - forms.py (формы приложения)
 - kmeans.py (вспомогательный модуль для реализации алгоритма K-Means (К-средних))
 - mysvc.py (вспомогательный модуль для реализации алгоритма)
 - migrations (миграции приложения)
 - templates (шаблоны приложения)
 - static (статические файлы приложения: css, js, images и другое)
- requirements.txt (список зависимостей проекта)
- trainmusic.csv (csv-файл с данными для обучения музыкальным предпочтениям классификатора Naive Bayes Classifier в кластеризации пользователей с использованием данных из социальной сети ВКонтакте)
- cafe_vectors.csv (csv-файл с данными об оценках, сгруппированных по объектам)
- user_vectors.csv (csv-файл с данными об оценках пользователей)