Санкт-Петербургский государственный университет

БИРИНА Екатерина Дмитриевна

Выпускная квалификационная работа

Разработка рекомендательной системы для выбора кинофильмов в процессе диалога

Уровень образования: бакалавриат

Направление 45.03.02 «Лингвистика»

Основная образовательная программа СВ.5106. «Прикладная, компьютерная и математическая лингвистика (английский язык)»
Профиль «Прикладная, компьютерная и математическая лингвистика (английский язык)»

Научный руководитель:
к.ф.н., доцент, Кафедра математической лингвистики,
Митренина Ольга Владимировна
Рецензент:
к.ф.н., доцент, Кафедра общего языкознания имени Л.А. Вербицкой,
Эйсмонт Полина Михайловна

Санкт-Петербург

2022

Аннотация

Работа посвящена разработке рекомендательной системы для выбора кинофильмов в процессе диалога, что предполагает пошаговую разработку чат-бота, а также создание кинематографической базы данных. Диалог чат-бота строится по принципу заранее прописанного шаблона, где пользователь должен выбрать самый подходящий ему вариант ответа на заданный вопрос. После завершения диалога сообщения человека сохраняются и анализируются для того, чтобы из созданной заранее базы данных выбрать для пользователя киноленту с наибольшим количеством совпадений. Чат-бот для социальной сети ВКонтакте создан на языке Python с использованием API ВКонтакте. В связи с возрастающей востребованностью виртуальных помощников, результаты работы могут быть применены во всех задачах, где возможно использование однотипных диалогов с пользователями, так как для решения этих задач достаточно лишь иметь в наличии измененные вопросы и ответы, а также новую базу данных.

Ключевые слова: чат-бот, диалоговая система, база данных, Python, диалог

A chatbot is a highly demanded software application used to conduct an on-line chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent. This work is devoted to the development of a recommendation system for choosing movies in the process of dialogue, which involves the step by step development of a chatbot and the creation of a movie database. The Python programming language and VKontakte API (application programming interface) are used for this study. The chatbot dialogue is built on the principle of a pre-written pattern, where the user must choose the most suitable answer to the question. After the end of the dialogue, the messages of the person are saved and analyzed in order to select the film with the greatest number of matches for the user from the database created in advance. On account for the increasing demand for virtual assistants, the

outcome of this work can be applied in all tasks where it is possible to use similar dialogues with users, since to solve them it is enough to have new prepared questions and answers, and database that is suitable for a new purpose.

Ключевые слова: chatbot, dialogue system, database, Python, dialogue

Оглавление

Введение

Диалоговые системы являются одним из наиболее интересных, перспективных и захватывающих направлений компьютерной лингвистики. Эта научная область чрезвычайно востребована в современном мире. Чат-бот ведет коммуникацию от лица компании или бренда с целью упростить онлайн-общение (предоставить актуальную информацию в наиболее оперативные сроки), используется как альтернатива переписке с живым оператором или звонку менеджеру компании.

Возрастающая востребованность виртуальных помощников **определяет актуальность** данной выпускной квалификационной работы.

Целью выпускной квалификационной работы является разработка системы, способной дать рекомендацию в выборе кинофильма на основе собранных о пользователе данных в процессе диалога с ним. Исходя из поставленной цели, необходимо решить следующие **задачи:**

- изучить основные аспекты диалога;
- составить схему диалога;
- написать программу, собирающую необходимые данные с сайта «Кинопоиск» для того, чтобы создать базу данных фильмов;
- разработать чат-бот на языке Python, который общается с пользователем в вопросно-ответной форме и сохраняет ответы собеседника на задаваемые вопросы;
- написать программу, способную обрабатывать полученные данные, чтобы подобрать для пользователя потенциально интересную ему киноленту;
- подключить созданный чат-бот к социальной сети ВКонтакте.

Данная работа состоит из введения, четырёх глав, заключения, списка литературы из 6 названий и приложений. В первой главе рассматриваются теоретические вопросы, связанные с диалоговыми системами и чат-ботами. Вторая глава посвящена описанию работы чат-бота в социальной сети ВКонтакте. В третьей главе рассматриваются определение парсинга и его этапы,

приводится описание работы программы, которая собирает необходимую информацию с сайта «Кинопоиск» и создает базу данных фильмов. В четвертой главе описаны схема диалога и пошаговая разработка чат-бота, а также представлена оценка полученных результатов.

Глава 1. Диалоговые системы

1.1 Введение в диалоговые системы

Во многих научно-фантастических книгах и фильмах часто встречается общение человека с компьютером. Исследования в этой области начались практически одновременно с появлением первых вычислительных машин. Привлекательность общения с компьютером именно диалогом заключается в том, что это естественный для человека способ получения информации. **Диалоговые системы** (dialogue systems), а также их упрощенные версии **чат-боты** (chatterbots, chatbots) занимаются обеспечением возможности такого общения. [Постнаука]

Диалоговая система, или интерактивная система, — это автоматизированная человеко-машинная система, работающая в режиме диалога, при котором она отвечает на каждую команду пользователя и по мере надобности обращается к нему за информацией. [Мильчин 2003]

Диалог — разговор между двумя лицами, обмен репликами. [Ожегов 1949]

Диалоговая система позволяет решать разного рода задачи в процессе диалога пользователя с заранее разработанной и автоматизированной системой. Характерной чертой диалоговой системы является ориентация на создание дружественного интерфейса, основу которого составляют следующие факторы: простота пользования; гибкость диалога(способность системы учитывать различные потребности и уровень квалификации пользователей); обеспечение идентификации и защиты данных; ясность поведения системы для пользователя в любой стадии диалога; доступность системы в любой необходимый пользователю момент; простота обучения работе с системой; самостоятельность(способность системы самостоятельно разбираться в «нештатных» ситуациях). [Власов 2005]

Классификация диалоговых систем. Диалоговые системы можно охарактеризовать по следующим признакам: *General — Task-oriented* (общего назначения — задаче-ориентированный) и *Open Domain — Closed Domain* (говорящие на любую тему или только на определенную). В каждой из пар

первый компонент существенно сложнее второго. [Habr]. Например, автоинформатор (система автоматического воспроизведения сообщений по телефону) точно является *Task-oriented* и *Closed Domain*.

1.2 Аспекты диалога

Прежде чем строить диалоговую систему, необходимо разобраться в особенностях диалога на естественном языке. Построение диалога на естественном языке — это очень непростой процесс, поскольку речевое общение — это сложная деятельность, включающая в себя не только понимание человеческого языка, но и генерацию высказываний. При создании диалоговых систем нужно принять во внимание следующие аспекты: порядок обмена репликами (turn-taking), общий контекст собеседников (grounding), структуру диалога (conversational structure), а также того, кто берет на себя ведущую роль в беседе (initiative). Рассмотрим подробнее каждый из этих аспектов [Константинова, Дегтева 2016].

1. Порядок обмена репликами. Именно этот аспект задает тот момент, когда следующий собеседник должен вступить в диалог и взять инициативу на себя. Чаще всего люди интуитивно понимают, когда настает их время говорить. Но проблема заключается в том, автоматическим системам понять это бывает трудно. Логичнее всего предположить, что люди начинают говорить, когда наступает тишина, но исследования показывают, что люди могут догадаться по контексту, когда им следует вступать в разговор. Поэтому важно понимать то, что было сказано, ведь это играет в данном процессе важную роль. Существуют определенные правила, по которым можно определить момент, когда приходит время вступать другому собеседнику. Несмотря на сложность проектирования данного аспекта в диалоговых системах, он является важным фактором естественности диалога.

2. Наличие контекста диалога. Эту характеристику, также следует учитывать при разработке диалоговых систем. При общении людям необходимо иметь общий контекст, информацию, которую собеседники используют для

интерпретации высказываний друг друга. Такой тип информации по-английски называется общей почвой (common ground). В ходе разговора участники диалога выстраивают модель дискурса, куда постепенно добавляют кусочки информации для успешного ведения беседы. Участники заранее предполагают какой-то объем информации и в процессе диалога пытаются добавить его к общему контексту. Однако все участники обладают своим видением информации и потому для успешного диалога необходимо постоянное подтверждение данного контекста посредством "Да", "Ясно", "Понятно" или дополнительных вопросов. Если это условие не выполняется, собеседники могут потерять нить разговора.

3. Ещё одной не менее важной характеристикой контекста является наличие у беседы темы (topicality), на протяжении всей беседы она может меняться или же оставаться постоянной. Общая тема обеспечивает понимание собеседниками друг друга и в некоторой степени определяет то, как строится диалог.

4. Специфическая структура. Она включает в себя не только содержание реплик, а также роли собеседников и то, что каждый из них должен сообщить. Проводились исследования, которые показали, что у людей есть внутреннее понимание структуры беседы, и они могут легко различать естественные диалоги и те, которые были автоматически сгенерированы. Во многих случаях структура беседы предопределяется ситуацией и диктует содержание. Например, беседу принято заканчивать прощанием, а не приветствием. Таким образом, если собеседники ставят перед собой цель – успешно общаться, то они должны следовать всем этим правилам и принимать во внимание структуру беседы.

5. Понятие инициативы в диалоге. Оно показывает, кто ведет беседу, иными словами, имеет лидирующее положение. Обычно присутствует смешанная инициатива (mixed-initiative), где ведущая роль переходит от одного собеседника к другому. Хотя бывают ситуации, когда инициатива принадлежит только одной стороне (single-initiative), например, при допросе полицией.

Все эти аспекты диалога имеют огромное значение при разработке диалоговых систем, именно благодаря им диалог кажется максимально приближенным к естественному человеческому общению.

Современные диалоговые системы имеют сложную архитектуру и состоят из нескольких модулей. Их архитектура может сильно различаться, но чаще всего в ней присутствует пять главных элементов: распознавание речи (speech recognition), понимание языка (natural language understanding), диалоговый менеджмент (dialogue management), генерация естественного языка (natural language generation) и синтез речи (speech synthesis). [Ураев 2019] Некоторые элементы являются необязательными и отсутствуют в ряде систем. Например, распознавание и синтез речи часто исключаются, поскольку обычно они реализуются как отдельные независимые модули.

1.3 Чат-боты

Чат-бот — это виртуальный собеседник, программа, которая создана для имитации поведения человека при общении с одним или несколькими собеседниками. Они работают по принципу поиска ключевых слов или же шаблонов в высказываниях людей и реакции на них, то есть подбор ответа из уже существующего набора данных. По сути, чат-боты являются упрощенными версиями диалоговых систем. Они имеют очень простую архитектуру.

Чат-бот — это программа, работающая внутри мессенджера, которая используется в разных сферах для решения типовых задач. [Кузьменко 2019] Такая программа способна отвечать на вопросы, а также самостоятельно задавать их. Сферы путешествий и развлечений стали одними из самых первых использовать чат-боты. Чат-бот может предложить направления для путешествия/рейсы/отели/рестораны — на основании поисковых запросов, а также предпочтений пользователя. Программа обеспечивает клиентскую поддержку, предоставляя ответы на часто задаваемые вопросы.

По статистике, сегодня количество пользователей мессенджеров уже превысило количество пользователей социальных сетей [Sensor Tower 2019]. В

мессенджере Telegram можно найти десятки тысяч разносторонних ботов. [Татаркин, Шмелева 2018] Некоторые из них способны рассказать о погоде (@WeathersBot) или помочь выбрать подарок. Есть боты, которые помогают пользователю найти музыку под настроение, например, @moodmusicbot. Также существуют боты, которые могут помочь изучать иностранные языки. [Будникова, Бабенкова 2020] Например, @andyrobot. Он помогает практиковать и изучать английский язык. В нем можно изучать отдельные слова, учить грамматику, строить диалоги и играть в игры. Также этот бот предлагает аудио с правильным произношением. @slangbot — Еще один помощник в изучении английского. В нем можно запросить толкование слова или получить объяснение случайного выражения из словаря этого бота. Еще есть @tap2bot. Этот бот ищет все: одежду, книги, фильмы, авиабилеты, отели, самые дешевые такси. Работает с Флибустой, РуТрекером, ВКонтакте и другими ресурсами. [Netology]

Но все эти чат-боты не выполняют главную задачу, поставленную еще в 20 веке, а именно создание такой диалоговой системы, общаясь с которой нельзя было бы точно сказать, с человеком или же с машиной ты ведешь диалог. Еще в 1950-м году Аланом Тьюрингом был предложен так называемый тест Тьюринга, задачей которого было определение качества имитации человеческого языка компьютером. В 1990-м году была учреждена премия Лёбнера, ежегодное соревнование чат-ботов на прохождение теста Тьюринга. [Коробкова, Долгова 2010]

В 1966-м году была создана программа ELISA, ее можно считать одним из первых чат-ботов. Принцип работы ELISA заключался в поиске ключевых слов во введённых пользователем сообщениях, при обнаружении ключевого слова, применялось правило, по которому сообщение пользователя преобразовывалось и возвращалось в виде предложения-результата. В случае если же ключевое слово не было найдено, ELISA либо давала пользователю общий ответ, либо повторяла один из предыдущих комментариев. ELISA также была запрограммирована на имитацию поведения психотерапевта, работающего по клиент-центрированной методике. Она пародировала речевое поведение

психотерапевта, реализуя технику активного слушания: переспрашивала пользователя и использовала фразы типа «Пожалуйста, продолжайте». Подобный подход позволял ELISA «притвориться, что она не знает почти ничего о реальном мире». Учитывая все это, программе удалось ввести в заблуждение некоторых людей, которые действительно думали, что разговаривают с реально существующим человеком. Из-за этого некоторые утверждают, что ELISA — одна из программ (возможно первая), которые смогли пройти тест Тьюринга. Однако это утверждение очень спорно, так как людей, которые общались с ELISA, изначально инструктировали так, чтобы они думали, что с ними будет разговаривать настоящий психотерапевт, и не подозревали о том, что они могут разговаривать с компьютером. [Habr2]

1.4 Использование диалоговых систем и чат-ботов

Создание диалоговых систем в современном мире является очень интересной и крайне востребованной задачей, так как, по статистике, на сегодняшний день число пользователей мессенджеров, например, Telegram и WhatsApp, беспрерывно растет, их уже стало больше, чем пользователей социальных сетей. Очевидно, что все компании также будут стремиться идти за интересами человека и предоставлять ему свои услуги не только, например, через веб-сайты, где пользователю необходимо выбрать какие-то графические элементы на веб-сайте, или же кликнуть на какие-то ссылки, но и через текстовые каналы коммуникации, когда он сможет общаться с различными сервисами в форме диалога и решать свои проблемы. [Бурцев 2017]

Еще одним важным преимуществом диалоговых систем является экономия человеческих ресурсов (уменьшение числа секретарей «на телефоне») и, соответственно, средств компаний. Гораздо проще и дешевле сделать подобную систему, а не держать целый штат сотрудников, которые, на самом деле, однотипно отвечают на одни и те же вопросы. Диалоговые системы уменьшают нагрузку на call-центры, снижают стоимость обслуживания, увеличивают выручку.

На данный момент диалоговые системы нашли широкое применение в нашей повседневной жизни, их можно встретить практически везде. [Матвеева, Золотарюк 2018] Так, например, если вы звоните в банк или в поликлинику, вы общаетесь, по крайней мере сначала, с диалоговой системой. На многих сайтах, посвященных онлайн покупкам можно также встретить различные диалоговые системы, целью которых является помощь клиенту. Ради быстрых и точных ответов люди готовы общаться с ботами-продавцами, консультантами, секретарями. Чат-боты и диалоговые системы можно использовать для поиска разнообразной информации. Например, чтобы узнать прогноз погоды, или же посмотреть афишу мероприятий. Вдобавок систему диалогов можно легко найти и в компьютерных играх. Сейчас существуют голосовые помощники или же виртуальные ассистенты, которые также являются диалоговыми системами. Например, Siri от Apple, Cortana от Microsoft и Алиса от Яндекса. [Барашко, Васильев, Зубань 2020]

Рассмотрим следующие основные направления применения чат-ботов, которые предложили специалисты Шауар Б. и Этвел Э. [Шауар, Этвел 2007]:

1. В сферах электронной коммерции и бизнеса. Чат-боты, использующиеся в данных сферах, имеют наибольший функционал и могут решать многие задачи. Например, рекламные коммуникации, продажи, онлайн-консультирование и др.
2. В сфере получения информации. Подобные чат-боты используются в электронных средствах массовой информации, для того чтобы предоставить пользователям интересные им новости.
3. В сфере обучения иностранным языкам и другим дисциплинам. Такой чат-бот можно назвать «бот-учитель». С их помощью можно самостоятельно изучать иностранные языки. Они способны предоставить учебный материал по дисциплине, а также оценить знания учеников.
4. В сфере развлечений. Чат-боты в данной сфере могут предоставлять пользователю интересную информацию, играть с ним и выполнять другие развлекательные функции.

Таким образом, диалоговые системы пользуются огромной популярностью в современном мире и применяются во многих сферах жизни человека, например, в бизнесе, образовании, СМИ и сфере развлечений, что характеризует их как довольно универсальный инструмент коммуникаций. [Смыслова 2018]

Глава 2. Работа чат-бота в социальной сети ВКонтакте

2.1 Чат-боты во ВКонтакте

Общение в социальных сетях и мессенджерах построено именно на диалоге между пользователями, а, следовательно, в них можно найти множество примеров диалоговых систем. [Параскевов, Каденцева, Мороз 2017] Это и различные развлекательные боты, целью которых является помощь пользователю в изучении языков или же, например, в поиске музыки (подробнее раздел 1.3). Но также на просторах социальных сетей можно найти ботов, которые помогают в бизнесе и общаются с потенциальными клиентами по определенным командам, отвечая на стандартные вопросы. Подобные чат-боты помогают автоматизировать рутину, они рассказывают о скидках, рассчитывают стоимость услуг и так далее. Когда пользователь начинает диалог с чат-ботом, программа отправляет в ответ заранее заданные сообщения. [Черноморова, Воробьев 2020]

Рассмотрим подробнее чат-боты, которые можно найти во ВКонтакте. Их можно разделить на примитивные и продвинутые. **Примитивные** чат-боты могут общаться с пользователем только по заданным командам, т.е. присылать ответы, которые есть в их базе данных. В том случае, если пользователь задаст вопрос, которого нет в базе, бот может либо промолчать, либо прислать сообщение «Я вас не понимаю». Кроме того, такие чат-боты могут предлагать выбрать одну из команд — например, как «Займобот». **Продвинутые** чат-боты используют элементы искусственного интеллекта. Они обучаются в процессе общения с пользователями, поэтому могут то, чего не могут примитивные программы — например, улавливают смысл и анализируют настроение сообщения, связываются с внешними базами данных и передают оттуда информацию. [Netology2]

Существуют специальные сервисы для создания примитивного чат-бота Robochat, BotVK, Chatgun и другие, которые достаточно просто подключить к своей группе, а дальше работая со встроенными шаблонами и следуя инструкции с готовыми алгоритмами, можно создать своего бота. При помощи такого

сервиса можно не только быстро создать чат-бота, который способен проконсультировать клиента, пообщаться с ним или ответить на 2–3 вопроса, но и, покопавшись в настройках и уделив разработке больше времени, можно получить полноценного помощника, который автоматизирует рутину. Но в рамках выпускной квалификационной работы я создала своего чат-бота с нуля на Python, используя API Вконтакте. [Мешков, Прокопенко 2021]

2.2 API Вконтакте

API (application programming interface) — это посредник между разработчиком приложений и какой-либо средой, с которой это приложение должно взаимодействовать. Он упрощает создание кода, поскольку предоставляет набор готовых классов, функций или структур для работы с имеющимися данными. [Ong 2015]

API ВКонтакте — 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. [Документация ВК]

Например, для получения данных о пользователе с идентификатором 210700286 необходимо составить запрос такого вида: *https://api.vk.com/method/users.get?user_id=210700286&v=5.52*

Рассмотрим отдельно все его составляющие. [Документация]

- https:// — протокол соединения.
- api.vk.com/method — адрес API-сервиса.
- users.get — название метода API ВКонтакте. Методы представляют собой условные команды, которые соответствуют той или иной операции с базой данных — получение информации, запись или удаление. Например, users.get — метод для получения информации о пользователе, messages.send – метод для отправки сообщения пользователю. Именно эти методы понадобятся нам в дальнейшем.

- ?user_id=210700286&v=5.52 — параметры запроса. После названия метода нужно передать его входные данные (если они есть) — как обычные GET-параметры в http-запросе. В нашем примере мы сообщаем серверу, что хотим получить данные о пользователе с id=210700286 и формат этих данных должен соответствовать версии API 5.52. Входные параметры всегда перечислены на странице с описанием метода.

В ответ сервер вернет JSON-объект с запрошенными данными (или сообщение об ошибке, если что-то пошло не так). JSON — это формат записи данных в виде пар «имя свойства»: «значение».

Ответ на наш запрос выглядит так:

{"response":[{"id":210700286,"first_name":"Lindsey","last_name":"Stirling"}]}

Также для создания чат-бота во ВКонтакте нам понадобится модуль Long Poll. Именно он позволяет работать с сообщениями из сообщества в режиме реального времени.

Глава 3. Парсинг

3.1 Введение в парсинг

Парсинг — это процесс сбора данных из интернета с последующей их обработкой и анализом для различных целей. К этому способу прибегают, когда нужно обработать большой массив информации, с которым сложно, а иногда и невозможно, справиться вручную. Программа, которая занимается парсингом, называется парсер. [Закалин 2018] Для того, чтобы собрать необходимую информацию для базы данных кинофильмов, было необходимо создать парсер для сайта «Кинопоиск». Рассмотрим этапы парсинга и необходимые библиотеки для каждого из них. [Григорьев 2017]

Этапы парсинга:

1. Поиск данных
2. Извлечение информации
3. Сохранение данных

Начнем с первого этапа. Для того чтобы приступить к работе, в первую очередь нам будет необходима ссылка на страницу, с которой мы и будем парсить необходимую информацию. Также нам понадобятся такие библиотеки, как BeautifulSoup [BeautifulSoup], Requests [Requests], и модуль Regular Expressions(Re) [Regular Expressions].

На втором этапе начинается использование установленных библиотек и ссылки на сайт, информация которого заинтересовала нас.

Requests — это HTTP-библиотека для языка программирования Python. Она позволяет легко отправлять запросы HTTP, которые лежат в основе всемирной сети. При открытии веб-страницы, браузер направляет множество запросов на сервер этой веб-страницы, затем сервер отвечает на них, пересылая все необходимые данные для вывода страницы, после чего браузер отображает страницу, чтобы пользователь мог увидеть ее. [Крапивин, Гареева 2021]

Requests самостоятельно заходит на сайт по ссылке, которая была передана в функцию requests.get(ссылка). Этот тип запроса идентичен тому, какой использует браузер для просмотра этой же страницы, единственное отличие заключается в том, что библиотека Requests не может выполнить рендеринг кода HTML, поэтому на выходе получается только сам код.

HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык разметки документов для просмотра веб-страниц в браузере. Он работает за счет распространения элементов документа со специальными тегами. В HTML-документе хранится слишком много информации, поэтому необходимо использование библиотеки BeautifulSoup, благодаря которой проще находить нужные данные.

BeautifulSoup - это парсер для синтаксического разбора файлов HTML и XML, написанный на языке программирования Python. Он способен преобразовать даже неправильную разметку в дерево синтаксического разбора. В функцию BeautifulSoup подается результат работы библиотеки Requests, на выходе мы получаем структуру html-кода изучаемого сайта. Далее начинается работа с полученной информацией, извлекаем конкретные интересующие данные. [Zheng, He, Peng 2015]

Для того, чтобы понять, какие конкретно теги нам нужно исследовать для получения необходимых данных, в браузере можно воспользоваться инструментом «Inspect» (CTRL+SHIFT+I), это позволит достаточно просто увидеть, какая из частей разметки отвечает за тот или иной элемент страницы. Достаточно навести мышью на определенный тег, как он подсветит соответствующую информацию на странице. Найдя некий шаблон на странице, можно создать код, который бы работал для него. Функция find() выдаст информацию только из первого встреченного заданного тега. В случае когда нужно получить данные из нескольких одинаковых тегов, стоит использовать функцию findAll().

Полученная на данном этапе информация все еще далека от того, что мы хотим получить в итоге. Необходимо воспользоваться модулем Re для того,

чтобы выделить нужные нам слова и числа, а именно название фильма, жанры, год выпуска, рейтинг и так далее. Регулярные выражения(Re) - это мощный, гибкий и эффективный инструмент для сопоставления текста на основе заранее определенного шаблона. Они позволяют найти слова в тексте, используя специализированный синтаксис, с помощью которого описывается шаблон для поиска. Мы будем пользоваться функцией findall(), которая вернет нам список, содержащий все совпадения с заданным шаблоном. В данную функцию подается информация, которую мы извлекли из тегов, на выходе получим искомые слова и числа. [Прохоренок, Дронов 2019]

На последнем этапе парсинга нам необходимо сохранить всю полученную информацию. Для этого мы воспользуемся библиотекой XlsxWriter [XlsxWriter], которая позволит нам записать данные в файл Excel. Из нее нам нужны следующие функции: add_worksheet() и write(). Первая позволит нам создать лист в Excel, а вторая запишет туда все заранее собранные данные. Работа парсера останавливается на этом этапе.

3.2 Описание кода парсера

Для того чтобы написать парсер, прежде всего нужно определиться с тем, какую конкретно информацию мы собираем. Для себя я решила, что наибольший интерес представляют следующие данные: тип киноленты (фильм или же мультфильм), название, страна-производитель, год создания, жанры, продолжительность киноленты, рейтинг, а также id.

Приступаем к написанию кода парсера для сайта «Кинопоиск». Импортируем необходимые нам модули:

```
import requests
from bs4 import BeautifulSoup
import re
import xlsxwriter
```

После этого нам необходимо создать пустой список, в котором мы будем хранить ссылки, с которыми мы и будем работать впоследствии. В связи с тем, что у «Кинопоиска» стоит хорошая защита от парсинга, невозможно сразу извлекать данные с большого количества ссылок. [Поликарпов, Анисимов, Толстых 2020] Ограничимся десятью запросами за раз, это решение несомненно будет время затратным, но это позволит нам избежать блокировок. Ссылки имеют одинаковый вид, меняется лишь id фильма, поэтому создав цикл, который будет менять только последнюю цифру ссылки от 0 до 9, мы получим десять отдельных ссылок, которые мы и сохраним в наш словарь.

```
urls = []
for i in range(0,10):
        url = f'https://www.kinopoisk.ru/film/47{i}/'
urls.append(url)
```

После этого мы открываем на запись файл Excel, в которые позднее мы будем записывать извлеченные данные. Также мы создаем пустой список, в который мы будем записывать страницы Excel с порядковым номером, это необходимо для того, чтобы каждый отдельный фильм хранился на своей собственной странице. Это значительно упростит запись данных, в дальнейшем страницы будут объединены в одну, с помощью встроенных функций Excel.

```
workbook = xlsxwriter.Workbook('База_Данных_Авто.xlsx')
worksheets = []
```

Далее начинается основной цикл, в котором будет и парсинг полученной ссылки, и сохранение данных в файл Excel. Для того чтобы обращаться по очереди к каждой ссылке из нашего словаря, переменной *i* мы задаем значения от 0 и до цифры, которая равна длине нашего списка. Таким образом, к i мы станем обращаться, когда нам понадобится обратиться к ссылке под нужным

порядковым номером. С помощью requests переходим по выбранной ссылке и передаем полученный код в BeautifulSoup. Обязательно задаем кодировку 'utf-8' и библиотеку 'lxml', первая необходима для того, чтобы корректно расшифровать и считать все данные, вторая для обработки разметки сайта. Все приведенные ниже части кода находятся внутри этого цикла.

```
for i in range(len(urls)):
    r = requests.get(urls[i])
    r.encoding = 'utf-8'
    soup = BeautifulSoup(r.text, 'lxml')
```

Мы получили всю информацию с выбранной ссылки, теперь мы можем приступить к обработке полученных данных. Начнем с извлечения названия киноленты и года производства. Для этого из полученных строк нам необходимо найти интересующий нас класс. Узнать название класса можно, просто исследовав страницу. Получили тег *"styles_title__hTCAr"*, именно его мы и ищем в структуре html-кода изучаемого сайта, с помощью функции find(). Далее полученные строки очищаются от внутренних тегов, с помощью функции get_text(). Пример результата: Гладиатор (2000)Gladiator. Для выделения отдельных частей воспользуемся регулярными выражениями, так как результат стандартизован и выглядит всегда одинаково. Для нахождения и выделения русского названия: *'(.*?)\s\('*, где в скобках находится необходимая нам информация, . – это любой символ, сочетание **? нужно для того, чтобы поиск был «жадным», то есть брал максимально возможное количество символов до ограничения, ограничением служит сочетание *\s\(*, в котором *\s* – это пробел, а *\(* – скобка. Таким образом, в переменную сохранятся все символы, встретившиеся до этого сочетания. На следующем шаге мы извлекаем год: *'\((\d*?)\)'*, где стоит ограничение скобками справа и слева, *\d* – это обозначение числового символа. Для оригинального названия: *'\)(\D*)'*, слева есть ограничение в виде скобки, *\D** обозначает, что берутся все символы до первой встретившейся цифры, это

необходимо потому, что иногда у фильмов справа указывается возрастное ограничение, так как оно не является частью названия, мы просто не берем его. Если фильм является русским, то есть у него нет другого альтернативного названия, в переменной не будет ничего. На этом мы закончили с этим тегом.

```
# Русское, английское название и год
title = soup.find('div', class_ = "styles_title__hTCAr")
title1 = title.get_text()
titlerus = ''.join(re.findall(r'(.*?)\s\(',title1))
year = ''.join(re.findall(r'\((\d*?)\)',title1))
titleorig = ''.join(re.findall(r'\)(\D*)',title1))
```

Далее мы находим id фильма, его мы просто выделяем из ссылки, с помощью уже описанных ранее регулярных выражений. Ограничениями в данном случае будут слово film и две косых черты. На следующем шаге из тега *"styles_valueBlock___nWKb"* мы получаем рейтинг фильма, с помощью функции get_text().

```
# id
filmid = ''.join(re.findall(r'film\/(\d*)\/',urls[i]))

# Рейтинг
rating = soup.find('div', class_ = "styles_valueBlock___nWKb")
rating1 = rating.get_text()
```

Дальше мы разберем часть кода, результатом которой станет вся недостающая информация. Особенностью сайта «Кинопоиск» является то, что у него есть «светлые» и «темные» версии страниц с фильмами (Приложение 2 и Приложение 3). Это отображается в самой кодировке в теге, поэтому дальнейший код было необходимо прописать для обеих версий. Таким образом,

мы проверяем можно ли найти тег *"styles_valueDark__BCk93 styles_value__g6yP4"*, если программа выдаст ошибку, она автоматически сделает поиск по второму тегу *"styles_valueLight__nAaO3 styles_value__g6yP4"*. После успешного нахождения любого из тегов начинается извлечение данных. Для того чтобы обратиться к названию страны, с помощью регулярных выражений мы находим строчку, в которой есть слово *country*. В ней между символами «>» и «<» и будет указана искомая нами страна. Для поиска жанров мы поступаем точно так же, только ищем строку со словом *genre*. Единственным отличием является то, что жанров много, поэтому мы будем создавать список со множеством названий, а не просто одну переменную. Данные мы будем извлекать при заполнении таблицы Excel. Для того чтобы взять информацию про длительность киноленты, мы просто ищем строку, которая будет заканчиваться на «*мин*», из нее мы копируем впереди стоящие цифры. Таким способом данные про страну, жанры и время извлекаются для обеих версий сайта, поэтому эта часть кода используется дважды.

```
try:
        info = soup.findAll('div', class_="styles_valueDark__BCk93 styles_valu
e__g6yP4")
        country = ''.join(re.findall(r'country.*?>(.*?)<',str(info)))
        genre = re.findall(r'genre.*?>(.*?)<',str(info))
        time = ''.join((re.findall(r'>(\d*)\sмин',str(info)))[0])

except AttributeError:
        info = soup.findAll('div', class_="styles_valueLight__nAaO3 styles_valu
e__g6yP4")
        country = ''.join(re.findall(r'country.*?>(.*?)<',str(info)))
        genre = re.findall(r'genre.*?>(.*?)<',str(info))
        time = ''.join((re.findall(r'>(\d*)\sмин',str(info)))[0])
```

Теперь мы можем приступить к записи всех полученных данных в файл Excel. Первым шагом мы добавляем название страницы с порядковым номером в созданный заранее список страниц, затем мы извлечем это название и присвоим его странице уже непосредственно в Excel файле. Это необходимо сделать из-за специфики библиотеки xlsxwriter. Открыв созданную страницу, мы заполняем ячейки под номером «1» названиями столбцов, которые будут общими для всех кинолент.

```
worksheets.append(f'worksheet{i}')
worksheets[i] = workbook.add_worksheet()

# В первые ячейки пишем названия наших столбиков
worksheets[i].write('A1', 'Название')
worksheets[i].write('B1', 'Тип')
worksheets[i].write('C1', 'Страна')
worksheets[i].write('D1', 'Год производства')
worksheets[i].write('E1', 'Жанр')
worksheets[i].write('F1', 'Время')
worksheets[i].write('G1', 'Рейтинг')
worksheets[i].write('H1', 'id')
```

Теперь мы начинаем заполнение полученных столбцов. Из-за специфики pandas, который мы будем использовать на этапе выбора киноленты, наиболее подходящей под запросы пользователя, нельзя писать все жанры в одну ячейку, так как это ухудшит и замедлит работу библиотеки. Поэтому мы создаем переменную *j*, которая будет принимать значения от 2 и до цифры, которая равна длине нашего списка с жанрами + 2. Переменная *j* будет использоваться для того, чтобы указать номер заполняемой ячейки, так как ячейка под номером «1» занята названиями столбцов, мы начинаем со второй. Таким образом, сначала мы проверяем не является ли первый жанр мультфильмом, так как на «Кинопоиске»

информация о том, что кинолента является мультфильмом, указана только там. Если это так, то в тип киноленты мы записываем 'мультфильм', если нет, то 'фильм'. Остальные ячейки мы просто заполняем полученными данными, кроме ячеек с жанрами. Их мы заполняем в первый раз первым жанром, во второй вторым и так далее. Именно для этого и были созданы наш цикл и переменная *j*.

```
for j in range(2,len(genre)+2):
        if genre[0] == 'мультфильм':
                worksheets[i].write(f'B{j}', 'мультфильм')
        else:
                worksheets[i].write(f'B{j}', 'фильм')
        worksheets[i].write(f'A{j}', titlerus + '(' + titleorig + ')')
        worksheets[i].write(f'C{j}', country.lower())
        worksheets[i].write(f'D{j}', year)
        worksheets[i].write(f'E{j}', genre[j-2])
        worksheets[i].write(f'F{j}', time)
        worksheets[i].write(f'G{j}', rating1)
        worksheets[i].write(f'H{j}', filmid)
```

На этом запись данных закончена, мы можем закрыть получившийся файл Excel, делается это уже вне основного цикла.

```
workbook.close()
```

Получившиеся файлы я объединила в один. Также я добавила вручную в итоговую базу данных столбец со способом создания мультфильма, так как эта информация не представлена на сайте «Кинопоиск», но мне она показалась значимой. Для фильмов в этом столбце стоит прочерк. Итоговая база данных была отсортирована по рейтингу по убыванию для того, чтобы выдавать пользователю более качественную киноленту (Приложения 4 - 7). Получившаяся база данных содержит в себе более 1000 кинолент, 30 жанров и 25 стран.

Глава 4. Разработка чат-бота для социальной сети ВКонтакте

4.1 Схема диалога

Прежде чем приступить к написанию самого чат-бота, мне было необходимо составить схему диалога, которая представлена в Приложении 1. [Кочковая, Чечевичко 2021]

В первом сообщении чат-бот ставит пользователя в известность, что он ведет диалог с программой-собеседником, а также дает ему две опции на выбор:

1. прекратить общение, написав специальное ключевое слово, о котором пользователь узнает из этого же сообщения;
2. начать опрос, итогом которого станет выбранная программой кинолента.

Далее начинается сам опрос. В первом вопросе у пользователя уточняется, что именно он хочет посмотреть: фильм или мультфильм. Если пользователь выбирает мультфильм, то уточняется способ создания: рисованный, компьютерный, кукольный или пластилиновый. Этот выбор дается пользователю потому, что не все люди любят кукольные или пластилиновые мультипликационные фильмы.

Следующий вопрос является общим, в нем уточняется страна производитель. Здесь пользователь сам пишет страну, а не выбирает из предложенных вариантов, как это было в прошлых вопросах. Написанная человеком страна ищется в базе данных, если такой страны нет, то чат-бот переспросит пользователя. Далее у пользователя уточняется год создания фильма, а если точнее, то 20 или же 21 век. В следующем вопросе уточняется продолжительность киноленты. Предлагаемое время будет либо меньше 100 минут, либо больше.

В следующем вопросе у пользователя уточняется жанр киноленты. Здесь пользователю дается выбор без каких-либо ограничений в виде предложенных вариантов ответа. Так как я работаю с сайтом «Кинопоиск», то соответственно названия жанров в моей базе данных будут только оттуда. Поэтому если жанр,

написанный пользователем, не встретился в базе данных, то чат-бот попросит пользователя написать другой жанр, или же написать, например, синоним, который будет более известным и общепринятым понятием.

Дальше подключается вторая часть программы, где происходит вычисление наиболее релевантного фильма и выдача его пользователю. Дальше бот прощается и останавливает свою работу.

4.2 Подключение API

Для своей выпускной квалификационной работы я выбрала создание примитивного чат-бота на Python, который общается с человеком на тему фильмов. Он используется для того, чтобы подбирать человеку киноленты для просмотра, основываясь на проведенном во время диалога тестировании.

Для того чтобы создать бота, вначале нужно создать сообщество во ВКонтакте, к которому мы в последствии и привяжем нашу диалоговую систему. Выбрав сообщества и введя название, а также тематику группы, на открывшейся странице настроек, выбираем «Работа с API». Далее, нам необходимо создать API-ключ, выбираем нужные вам параметры с доступом для нашего API-ключа. Затем копируем полученный API-ключ в файл, так как он понадобится нам в написании самой программы. После нам нужно разрешить сообщения, так как мы пишем чат-боты именно для сообщений нашего сообщества. Вся подготовка группы завершена. Приступаем к программной части нашего бота.

Для реализации нашего чат-бота мы просто используем библиотеку VkLongPoll, чтобы не делать это через запросы во ВКонтакте. Для этого нам понадобится библиотека vk_api (vk_api – Python модуль для создания скриптов для социальной сети Вконтакте). Установим ее через командную строку: *python -m pip install vk_api*

4.3 Описание кода

После подключения API можно приступить к написанию кода. Импортируем нужные нам модули:

```
import random
import vk_api
from vk_api.longpoll import VkLongPoll, VkEventType
import pandas as pd
```

После этого необходимо открыть файл на запись с кодировкой utf-8, именно в нем мы будем сохранять сообщения пользователя. В файле info мы будем сохранять сообщения пользователя в том виде, в котором он их написал, а также его имя и фамилию. В файле Variants мы будем хранить «очищенные» сообщения для того, чтобы с ними в дальнейшем с ними было проще и удобнее работать.

```
info = open('Dialogue.txt', 'w', encoding = 'utf-8')
Variants = open('Variants.txt', 'w', encoding = 'utf-8')
```

Также сразу откроем заранее подготовленную парсером базу данных, в которой хранится информация более чем о 1000 фильмах, используя библиотеку pandas. [McKinney 2011]

```
DataBase = pd.read_excel('База_Данных.xlsx')
```

Создаем функцию write_msg, она будет получать id пользователя ВКонтакте (user_id), которому и будет отправлено собственно само сообщение при помощи метода ‘messages.send’. Random_id – это уникальный (в привязке к API_ID и ID отправителя) идентификатор, предназначенный для предотвращения повторной отправки одинакового сообщения. Без него наш бот просто не будет работать.

```
def write_msg(user_id, message):
```

```
vk.method('messages.send', {'user_id': user_id, 'message': message, 'rand
om_id': random.randint(0, 2048)})
```

На следующем шаге в переменную token мы заносим наш API-ключ, который мы создали в нашем сообществе и скопировали в файл, он понадобится нам для того, чтобы авторизоваться как сообщество. Это мы и делаем далее. Затем мы настраиваем longpool для работы с сообщениями.

```
token = "47ec8618b2ca38f123b41cedd87c38a88978e7bf15ec31dc2f620be879
024d8a8af"
vk = vk_api.VkApi(token=token) #Авторизуемся как сообщество
longpoll = VkLongPoll(vk)
```

Следующий модуль программы — это перечисление разнообразных приветствий и прощаний. Они хранятся в виде списка, написаны строчными буквами и без знаковых препинания, так как вариантов их написания огромное количество. Понижать регистр и убирать знаки препинания будем позднее.

```
Greetings = ['привет', 'здравствуйте', 'здравствуй', 'приветик', 'приветики', '
добрый день', 'доброе утро', 'добрый вечер']
Farewells = ['пока', 'до свидания', 'до скорого', 'счастливо', 'спокойной ночи
', 'хорошего дня']
```

Также выписываем возможные варианты ответов на вопросы, которые мы будем задавать в будущем, и с которыми мы и будем искать совпадения с нашей базой данных. Ниже приведены сокращенные списки вариантов стран и жанров, в программе они значительно длиннее.

```
Countries = ['франция', 'ссср', 'великобритания', 'сша']
Year = ['до', 'после']
```

```
Genre = ['драма', 'комедия', 'биография', 'фантастика', 'приключения', 'боевик', 'триллер', 'детектив']
Length = ['короткий', 'длинный']
Type = ['фильм', 'мультфильм']
Cartoon = ['рисованный', 'компьютерный', 'кукольный', 'пластилиновый']
```

Теперь мы готовы к созданию основного цикла. В нем мы циклически будем проверять на наличие event-ов. После этого получив сообщение от пользователя, мы сможем отправить ему соответствующий ответ с помощью уже созданной нами ранее функции write_msg.

```
for event in longpoll.listen(): #Создаю основной цикл
    if event.type == VkEventType.MESSAGE_NEW: #Если пришло новое сообщение
        if event.to_me: #Если оно имеет метку для меня (то есть бота)
            # получение имени и фамилии
            user = vk.method("users.get", {"user_ids": event.user_id})
            fullname = user[0]['first_name'] +  ' ' + user[0]['last_name']
            # Записываю в файл имя пользователя и его сообщение
            info.write(fullname + ': ' + event.text + '\n')
            request = event.text.lower().strip(')?.,(!') #Сообщение от пользователя. Понижаю регистр, потому что пользователь может писать и с большой буквы,
а также знаки препинания и скобочки, которые мешают на данном этапе
            Variants.write(request + '\n') #Записываем "очищенное" сообщение
```

Разберем некоторые моменты из самого цикла.

Во-первых, user = vk.method("users.get", {"user_ids": event.user_id}). В данной строчке при помощи метода "users.get" мы достаем расширенную информацию о пользователе. Но нам понадобится только его имя и фамилия. Их мы достаем в следующей строчке: fullname = user[0]['first_name'] + ' ' + user[0]['last_name']. Заносим имя и фамилия в одну переменную, чтобы потом можно было занести эту информацию в наш файл. Данные хранятся в виде словаря, поэтому мы и обращаемся к значениям через их ключи, а именно через 'first_name' и 'last_name'.

Во-вторых, info.write(fullname + ': ' + event.text + '\n'). В этой строчке мы записываем в наш файл имя и фамилию пользователя, с которым чат-бот ведет общение, а также его сообщение. Делаем перенос на новую строчку, чтобы было более наглядно.

В-третьих, request = event.text.lower().strip(')?.,(!') . Здесь мы преобразовываем сообщение пользователя, понизив регистр и убрав знаки препинания для более удобной обработки сообщения. Дальше мы добавляем преобразованное сообщение во второй файл, который мы будем использовать для сравнения с данными из базы данных. Variants.write(request + '\n')

Давайте рассмотрим саму логику ответа на сообщение пользователя. Мы проверяем попадает ли сообщение пользователя в прописанные нами словари приветствий и прощаний, если да, то мы выдаем прописанные нами ответы, обращаясь к человеку по имени, обратившись к значению через ключ, как мы делали это ранее. Например, проверяем для приветствия:

```
if request in Greetings:
        write_msg(event.user_id, "Здравствуйте, " + user[0]['first_name'] + "! С
ейчас с Вами говорит Бот. Я могу посоветовать Вам фильм после коротко
го общения. Если вы хотите попробовать, то просто напишите 'Давай', есл
и же Вам это неинтересно, то напишите 'Стоп'")
```

Для того чтобы отправить сообщение мы должны указать id пользователя, так как это прописано в требованиях метода **'messages.send'**. Следующим вариантом является тот случай, когда сообщение пользователя попадает под определенный шаблон. Например:

```
elif request == 'как дела':
        write_msg(event.user_id, "Всё хорошо!")
```

Здесь мы просто проверяем совпадение с шаблоном, если они совпали, то мы просто выдаем заранее заготовленный ответ. Подобные шаблоны можно писать до бесконечности, но здесь я описываю только правила создания чат-бота по такому принципу. Здесь описан лишь “скелет” настоящего бота.

Следующий вариант нужен для того, чтобы остановить общение чат-бота с пользователем, после этого вся беседа запишется в наш файл. Если же человек не напишет ключевое слово, то сообщения пользователя никуда не запишутся, так как программа не закончит свою работу. Ключевым словом для нашего бота я выбрала “Стоп”, о нем я сообщила пользователю еще в первом сообщении. Это выглядит так:

```
elif request == 'стоп':
break
```

Самым последним вариантом является случай, когда сообщение пользователя не попало ни под один шаблон. В таком случае мы выдаем стандартный ответ для всех ботов “Не поняла вашего ответа”:

```
else:
        write_msg(event.user_id, "Простите, я не поняла Вашего ответа! Пожалуйста, проверьте Ваше сообщение на наличие опечаток и
```

```
попробуйте снова! Также ошибку может вызывать то, что Вы написали что-то, чего нет в моей базе данных!")
```

Приступим к описанию непосредственно общения бота и человека в вопросно-ответной форме.

```
elif request == 'давай':
        write_msg(event.user_id, "Отлично! Тогда начнем! Скажите, чтобы вы хотели посмотреть: \nФильм \nМультфильм")
elif request == Type[1]:
        write_msg(event.user_id, "Чудесно! Скажите, какой мультфильм вы бы хотели посмотреть: \nРисованный \nКомпьютерный \nКукольный \nПластилиновый")
elif request == Type[0]:
        write_msg(event.user_id, "Восхитительно! Скажите, фильм какой страны вы бы хотели посмотреть: \nФранция \nРоссия \nСША \nВеликобритания")
elif request in Cartoon:
        write_msg(event.user_id, "Восхитительно! Скажите, мультфильм какой страны вы бы хотели посмотреть? ")
elif request in Countries:
        write_msg(event.user_id, "Хорошо) Следующий вопрос. Скажите, вы хотели бы посмотреть киноленту, снятую до 2000 года или после? \nПросто напишите 'До' или 'После'.")
elif request in Year:
        write_msg(event.user_id, "Прекрасно! Следующий вопрос. Какой жанр киноленты интересен Вам сейчас больше всего?")
elif request in Genre:
```

```
write_msg(event.user_id, "Супер! Следующий вопрос. Киноленту как
ой длины Вы бы хотели посмотреть сейчас? \nКороткий (меньше 100 мин
ут) \nДлинный (больше 100 минут)")
```

В таком случае: elif request == 'давай', бот понимает, что можно начинать опрос пользователя. Далее пользователь отвечает на поставленные вопросы, если ответы попадают в список возможных, то это сигнал для бота, что диалог идет успешно и можно задавать следующий вопрос. Так, например, у нас было прописано для стран, где был снят фильм: Countries = ['франция', 'сша', 'великобритания', 'россия']. Если же ответ не попал в этот список, бот попросит пользователя исправить ответ. Также у нас был отдельный вопрос-уточнение для мультфильмов. Таким образом, для мультфильмов уточнялся способ их создания.

Дальше начинается самое сложное. Как только пользователь отвечает на вопрос про длину фильма, начинается обработка ответов для дальнейшей работы чат-бота. Это происходит таким образом:

```
elif request in Length:
        info.close()
        Variants.close()
        info2 = open('Variants.txt', 'r', encoding = 'utf-8')
        Answers = []

        for line in info2:
            if line.strip('\n') in Type:
                Answers.append(line.strip('\n'))
            elif line.strip('\n') in Cartoon:
                Answers.append(line.strip('\n'))
            elif line.strip('\n') in Countries:
                Answers.append(line.strip('\n'))
```

```
        elif line.strip('\n') in Year:
            Answers.append(line.strip('\n'))
        elif line.strip('\n') in Genre:
            Answers.append(line.strip('\n'))
        elif line.strip('\n') in Length:
            Answers.append(line.strip('\n'))
```

Для начала мы закрываем файлы, в которые записывались сообщения пользователя. Дальше мы открываем на чтение файл info2 с «очищенными» сообщениями собеседника. Также мы создаем список Answers, в который будем записывать нужные нам для анализа ответы, с помощью цикла for. После нахождения всех необходимых данных, мы наконец-то можем приступить к тому, чтобы найти наиболее подходящий для пользователя фильм и посоветовать его. Так как мультфильмы содержат дополнительную характеристику, а именно способ создания, список для них соответственно будет длиннее. Поэтому для корректной работы программы далее последует разделение. Если длина списка равна 5, то пользователь выбрал фильм. Код будет выглядеть так:

```
    true_DataBase = DataBase.loc[DataBase['Тип'] == Answers[0]]
    true_DataBase_1 = true_DataBase.loc[true_DataBase['Страна'] == Answers[1]
]
    if true_DataBase_1.empty == True:
        true_DataBase_1 = true_DataBase
    if Answers[2] == 'до':
        true_DataBase_2 = true_DataBase_1.loc[true_DataBase_1['Год произво
    дства'] < 2000]
    else:
        true_DataBase_2 = true_DataBase_1.loc[true_DataBase_1['Год произво
    дства'] >= 2000]
```

```
    if true_DataBase_2.empty == True:
        true_DataBase_2 = true_DataBase_1
    true_DataBase_3 = true_DataBase_2.loc[true_DataBase_2['Жанр'] == Answers[3]]
    if true_DataBase_3.empty == True:
        true_DataBase_3 = true_DataBase_2
    if Answers[-1] == 'короткий':
        true_DataBase_4 = true_DataBase_3.loc[true_DataBase_3['Время(в минутах)'] < 100]
    else:
        true_DataBase_4 = true_DataBase_3.loc[true_DataBase_3['Время(в минутах)'] >= 100]
    if true_DataBase_4.empty == True:
        true_DataBase_4 = true_DataBase_3
```

На первом шаге мы ограничиваем всю базу данных нужным нам типом киноленты, его выбрал пользователь еще в первом вопросе (true_DataBase = DataBase.loc[DataBase['Тип'] == Answers[0]]). Далее происходит ограничение нужной нам страной, так как я посчитала это следующим наиболее важным показателем для выбора фильма (true_DataBase_1 = true_DataBase.loc[true_DataBase['Страна'] == Answers[1]]). Крайне важными являются эти строки:

```
    if true_DataBase_1.empty == True:
        true_DataBase_1 = true_DataBase
```

В них проверяется не получился ли у нас пустой фрейм, то есть не случилось ли так, что фильмов с таким типом и такой страной просто нет в нашей базе данных. Если подобное произошло, то мы снимаем установленное ограничение и возвращаемся к последнему удачному варианту. Подобную

проверку мы будем проводить на каждом этапе, так как это позволит нам выдать пользователю киноленту со стопроцентной вероятностью.

Дальше мы проверяем нужен ли пользователю фильм до или после 2000 года и в зависимости от результата также ограничиваем варианты предлагаемых фильмов. Далее мы совершаем такие же шаги и для других данных. В конце мы даем пользователю наиболее подходящий под его запрос фильм:

```
write_msg(event.user_id, "Спасибо большое за общение! Вот, что я выбрал для Вас:" + '\n' + str(DataBase.iloc[true_DataBase_4.index[0]]['Название']) + ' (' + str(DataBase.iloc[true_DataBase_4.index[0]]['Год производства']) + ')' + '\nРейтинг киноленты: ' + str(DataBase.iloc[true_DataBase_4.index[0]]['Рейтинг']) + '\nСсылка на фильм: https://www.kinopoisk.ru/film/' + str(DataBase.iloc[true_DataBase_4.index[0]]['id']) + '/')
info2.close()
```

Мы берем номер строки, которая выдалась после работы нашей программы по подбору идеальной киноленты, и подставляем ее в нашу изначальную базу данных, чтобы выписать оттуда название фильма, год производства, его рейтинг и ссылку на фильм. После чего мы останавливаем работу нашего чат-бота, не забыв закрыть файл, с которым мы работали.

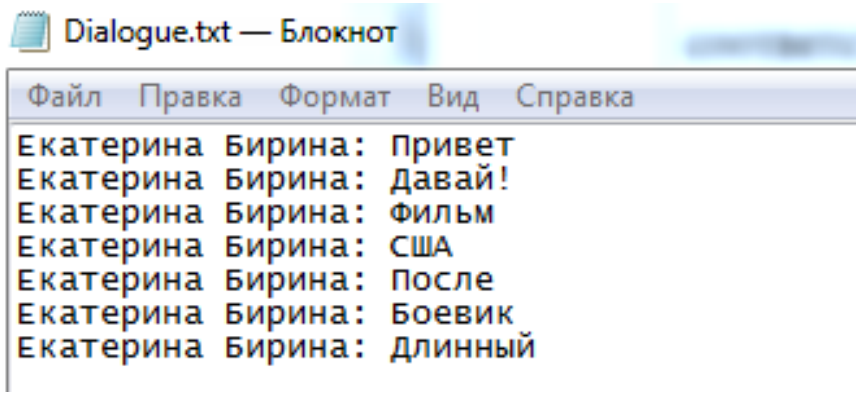
Для мультфильмов код абсолютно такой же за исключением того, что добавляется дополнительное ограничение на базу данных.

```
true_DataBase_1 = true_DataBase.loc[true_DataBase['Способ создания'] == Answers[1]]
if true_DataBase_1.empty == True:
    true_DataBase_1 = true_DataBase
```

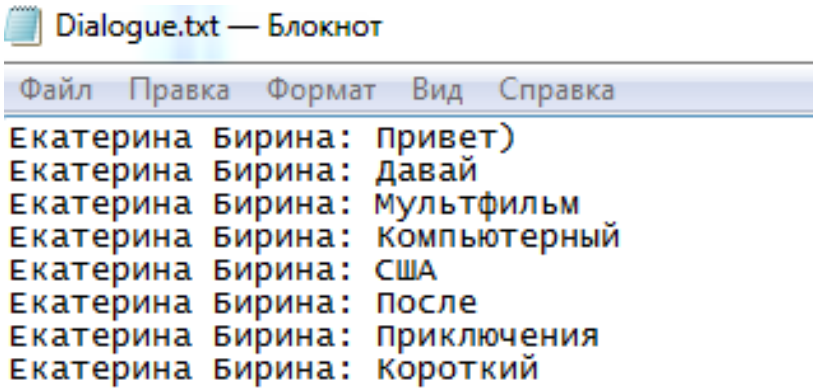
На этом цикл с ответами бота закончен. Как я и говорила ранее, шаблоны и вопросы можно дописывать бесконечно долго для того, чтобы сделать общение более похожим на человеческое.

Данные в файл с сообщениями записываются в таком виде:

1) Для фильма:

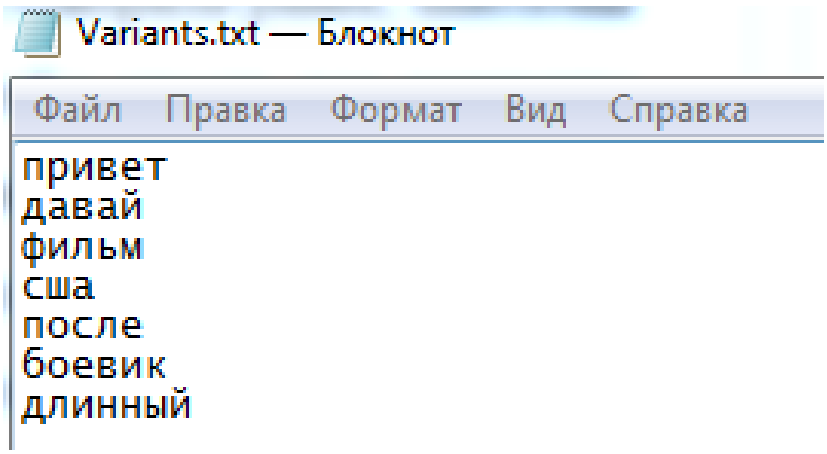


2) Для мультфильма:

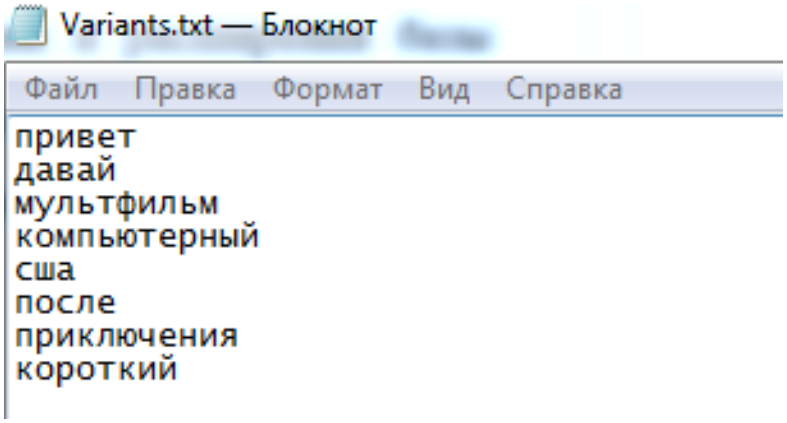


Данные в файле с «очищенными» ответами выглядят так:

1) Для фильма:



2) Для мультфильма:



Пример диалога с пользователем находится в Приложении 8 и Приложении 9.

В дальнейшем этот чат-бот может использоваться в группах во ВКонтакте по кинематографу. Кроме того, результаты работы могут быть применены во всех задачах, где возможно использование однотипных диалогов с пользователями, так как для решения этих задач достаточно лишь иметь в наличии измененные вопросы и ответы, а также новую базу данных.

4.4 Тестирование результатов

Итоговая версия рекомендательной системы была проверена на 30 пользователях. Испытания проводились в группе во Вконтакте, где была реализована диалоговая система. Каждый пользователь провел в среднем три диалога. Во всех случаях использования пользователям выдавались наиболее релевантные киноленты, не было зафиксировано ни одного случая с ошибкой или с нулевым результатом.

После остановки работы чат-бота пользователи ставили оценку рекомендательной системе. Использовалась пятибалльная шкала оценок. Кроме того, каждый пользователь отвечал на следующие вопросы:

- Удовлетворяет ли выбранный фильм всем вашим запросам?
- Понравился ли вам опрос?
- Покрыл ли опрос все ваши пожелания?
- Понравилась ли вам сама диалоговая система, не учитывая опрос?

- Возникли ли какие-то проблемы при общении?
- Возникли ли какие-то недопонимая при прохождении опроса?
- Хотите ли вы попробовать еще раз?
- Хотите ли вы предложить какие-то варианты для улучшения рекомендательной системы?

В итоге были получены следующие общие оценки: 18 «5», 10 «4» и 2 «3».

Из минусов пользователи отмечали недостаточно сильную диалоговую систему в целом. Они говорили, что за приделами опроса чат-бот часто отвечал пользователям, что не понимает их. Это вполне обоснованно. Согласно классификации диалоговых систем, мой чат-бот можно охарактеризовать как Task-oriented (задаче-ориентированный) и Closed Domain (способный говорить на строго определенную тему). Я не ставила перед собой задачу создать сложную диалоговую систему, которая будет приближена к искусственному интеллекту, поэтому часто сообщения пользователя, которые не были связаны с кинематографом, просто не распознавались.

Хотя большое количество жанров и стран-производителей, имеющихся в базе данных, способствовало тому, что пользователю не приходилось менять свои пожелания, две «3» были получены в тех случаях, когда пользователи писали жанры, которых просто не было в базе данных, несмотря на то, что в ней и находится 30 вариантов. Так, например, один из пользователей написал «гангстерский фильм», такого жанра просто нет в базе данных. Это обусловлено тем, что я брала информацию только с сайта «Кинопоиск», и там просто нет такого жанра. Зато есть «криминал» и «боевик», поэтому после создания словаря синонимов, эта проблема исчезла. Еще одним жанром, которого нет в базе данных, был «политический фильм». Аналогами в имеющейся базе можно назвать жанры «исторический», «документальный» и «биография».

Таким образом, за исключением указанных ошибок, пользователи высоко оценили рекомендательную систему, а также отметили высокую степень совпадения характеристик выданной киноленты с их пожеланиями.

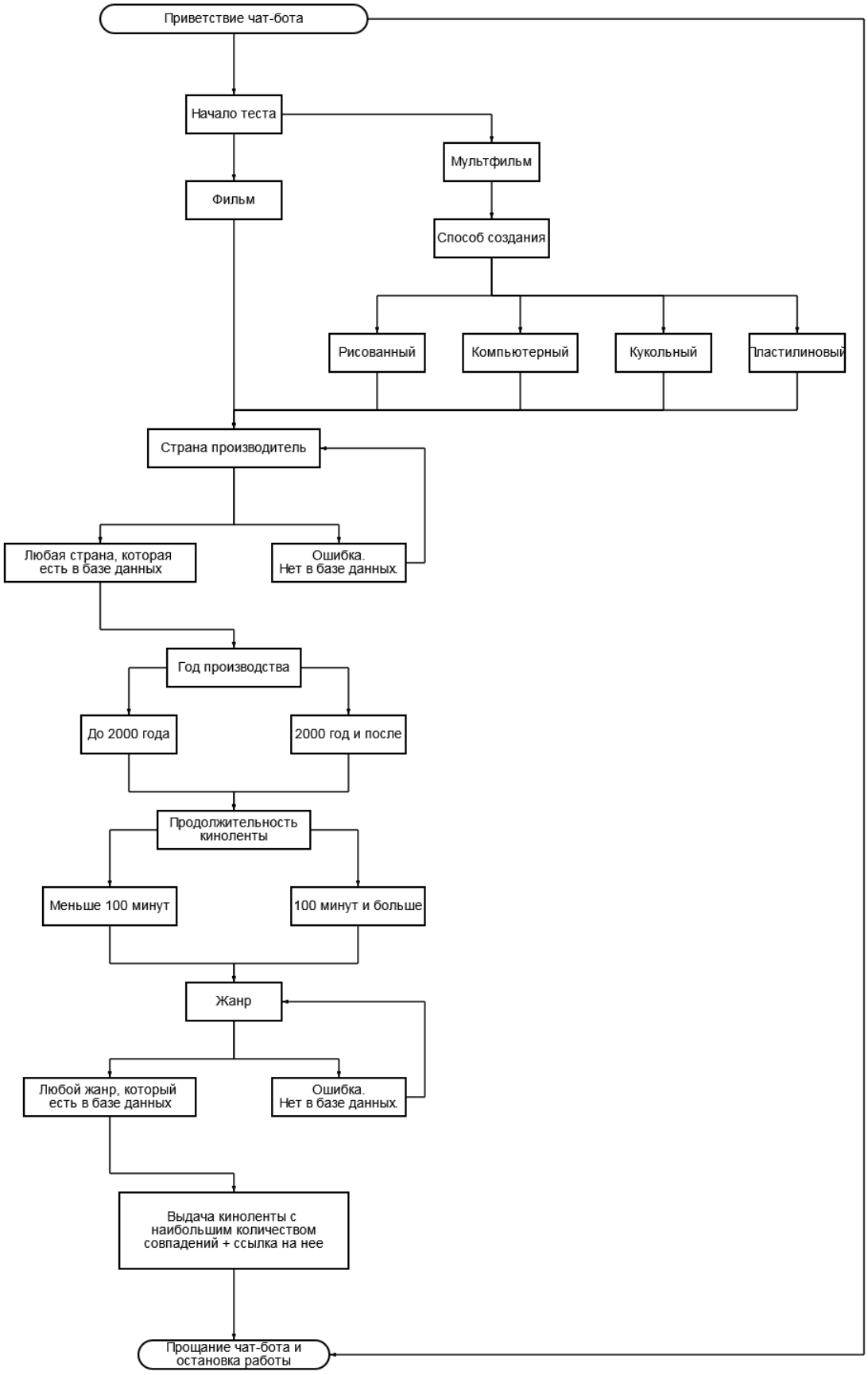
Заключение

В данной работе было описано создание рекомендательной системы для выбора кинофильмов в процессе диалога. Для этого был разработан парсер, собирающий необходимые данные с сайта «Кинопоиск» и создающий базу данных фильмов и мультфильмов. Также была составлена схема диалога, на основе которой был разработан чат-бот, который может собирать данные про пользователя, общаясь с человеком в вопросно-ответной форме на тему кинематографа. При создании чат-бота учитывались основные аспекты диалога. Разработанный чат-бот задает вопросы пользователю, после чего записывает ответы собеседника в отдельный файл. Полученные данные обрабатываются специальной программой, в результате работы которой пользователю выдается кинолента с наибольшим количеством совпадений. Данный чат-бот был подключен к социальной сети Вконтакте и успешно протестирован с помощью 30 пользователей.

После пополнения базы данных кинофильмов разработанная рекомендательная система может быть использована на одной из платформ, посвященных кинематографии. Система может быть легко расширена для учета других требований пользователя.

Приложения

Приложение 1. Схема диалога.



Приложение 2. «Светлая» версия сайта.

Трейлер
9 июля 2007

+ Буду смотреть

Все папки пользователей

Волшебник страны Оз (1939)

The Wizard of Oz 0+

Буду смотреть

Где смотреть 2

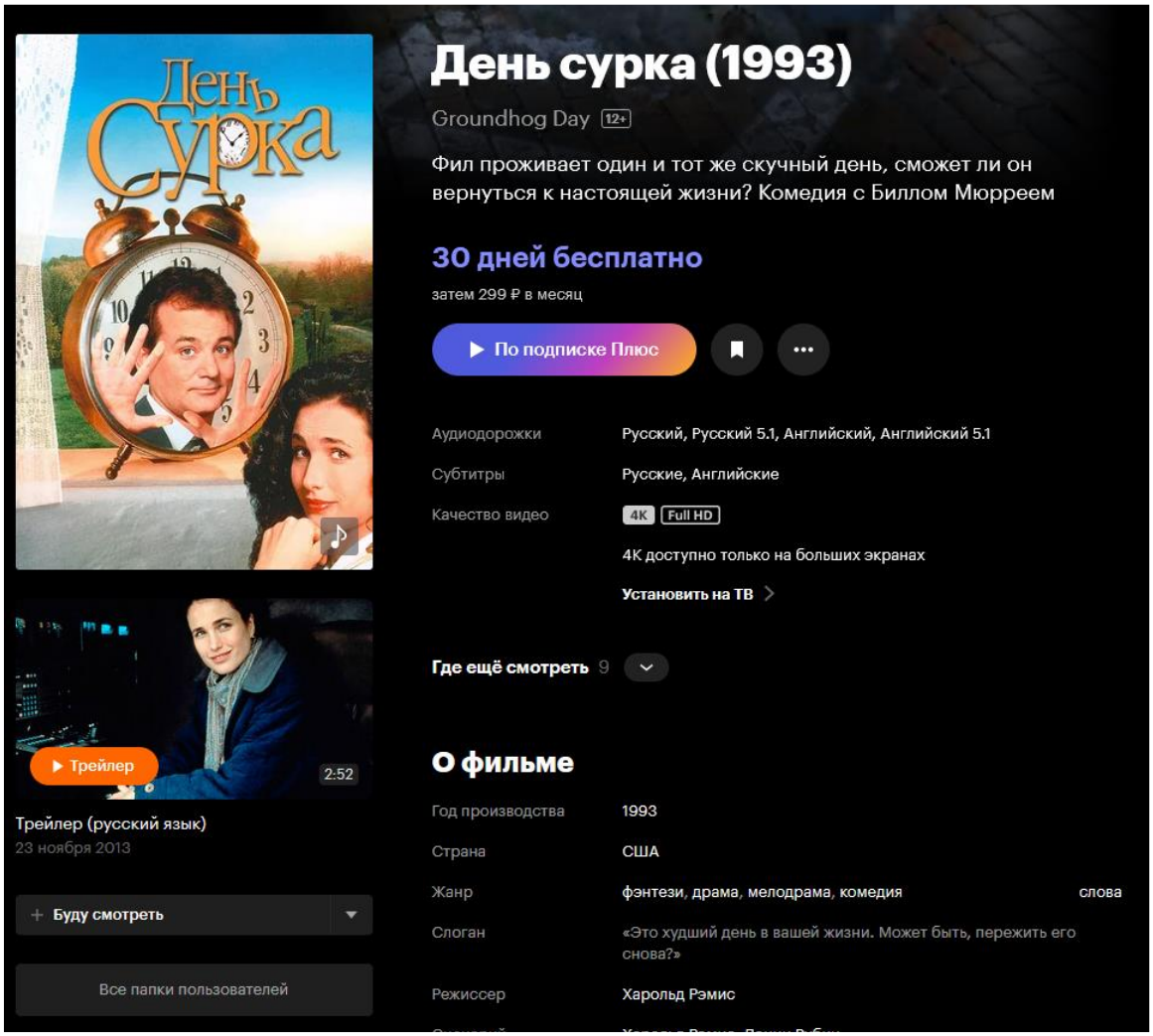
IVI

Apple TV

О фильме

Год производства	1939
Страна	США
Жанр	мюзикл, фэнтези, приключения, семейный слова
Слоган	«What is Oz?»
Режиссер	Виктор Флеминг, Джордж Кьюкор, Мервин ЛеРой, ...
Сценарий	Флоренс Райэрсон, Эдгар Аллан Вулф, Артур Фрид, ...
Продюсер	Мервин ЛеРой, Артур Фрид
Оператор	Гарольд Россон
Композитор	Харольд Арлен, Херберт Стотхарт
Художник	Малкольм Браун, Уильям А. Хорнинг, Джек Мартин Смит, ...
Монтаж	Бланш Сьюэлл
Бюджет	$2 777 000
Сборы в США	$24 770 492
Сборы в мире	+ $969 000 = $25 739 492 сборы
Зрители	3.3 млн, 22.6 тыс, 2.2 тыс, ...
Премьера в мире	10 августа 1939, ...

Приложение 3. «Темная» версия сайта.



Приложение 4. База данных 1.

	Название	Тип	Страна	Год пр	Жанр	Время	Рейтинг	id	Способ создания
2	Побег из Шоушенка(The Shawshank Redemptior	фильм	сша	1994	драма	142	9,1	326	-
3	Зеленая миля(The Green Mile)	фильм	сша	1999	драма	189	9,1	435	-
4	Зеленая миля(The Green Mile)	фильм	сша	1999	криминал	189	9,1	435	-
5	Форрест Гамп(Forrest Gump)	фильм	сша	1994	драма	142	8,9	448	-
6	Форрест Гамп(Forrest Gump)	фильм	сша	1994	комедия	142	8,9	448	-
7	Форрест Гамп(Forrest Gump)	фильм	сша	1994	мелодрама	142	8,9	448	-
8	Форрест Гамп(Forrest Gump)	фильм	сша	1994	история	142	8,9	448	-
9	Форрест Гамп(Forrest Gump)	фильм	сша	1994	военный	142	8,9	448	-
10	1+1(Intouchables)	фильм	франция	2011	драма	112	8,8	535341	-
11	1+1(Intouchables)	фильм	франция	2011	комедия	112	8,8	535341	-
12	1+1(Intouchables)	фильм	франция	2011	биография	112	8,8	535341	-
13	Иван Васильевич меняет профессию	фильм	ссср	1973	фантастика	88	8,8	42664	-
14	Иван Васильевич меняет профессию	фильм	ссср	1973	комедия	88	8,8	42664	-
15	Иван Васильевич меняет профессию	фильм	ссср	1973	приключения	88	8,8	42664	-
16	Список Шиндлера(Schindler's List)	фильм	сша	1993	драма	195	8,8	329	-
17	Список Шиндлера(Schindler's List)	фильм	сша	1993	биография	195	8,8	329	-
18	Список Шиндлера(Schindler's List)	фильм	сша	1993	история	195	8,8	329	-
19	Список Шиндлера(Schindler's List)	фильм	сша	1993	военный	195	8,8	329	-
20	Начало(Inception)	фильм	великобритания	2010	фантастика	148	8,7	447301	-
21	Начало(Inception)	фильм	великобритания	2010	боевик	148	8,7	447301	-
22	Начало(Inception)	фильм	великобритания	2010	триллер	148	8,7	447301	-
23	Начало(Inception)	фильм	великобритания	2010	драма	148	8,7	447301	-
24	Начало(Inception)	фильм	великобритания	2010	детектив	148	8,7	447301	-
25	Леон(Léon)	фильм	франция	1994	боевик	133	8,7	389	-
26	Леон(Léon)	фильм	франция	1994	триллер	133	8,7	389	-
27	Леон(Léon)	фильм	франция	1994	драма	133	8,7	389	-
28	Леон(Léon)	фильм	франция	1994	криминал	133	8,7	389	-
29	Крестный отец(The Godfather)	фильм	сша	1972	драма	175	8,7	325	-
30	Крестный отец(The Godfather)	фильм	сша	1972	криминал	175	8,7	325	-
31	Криминальное чтиво(Pulp Fiction)	фильм	сша	1994	триллер	154	8,6	342	-
32	Криминальное чтиво(Pulp Fiction)	фильм	сша	1994	комедия	154	8,6	342	-
33	Криминальное чтиво(Pulp Fiction)	фильм	сша	1994	криминал	154	8,6	342	-

Приложение 5. База данных 2.

34	Назад в будущее(Back to the Future)	фильм	сша	1985	фантастика	116	8,6	476	-
35	Назад в будущее(Back to the Future)	фильм	сша	1985	комедия	116	8,6	476	-
36	Назад в будущее(Back to the Future)	фильм	сша	1985	приключения	116	8,6	476	-
37	Гладиатор(Gladiator)	фильм	сша	2000	история	155	8,6	474	-
38	Гладиатор(Gladiator)	фильм	сша	2000	боевик	155	8,6	474	-
39	Гладиатор(Gladiator)	фильм	сша	2000	драма	155	8,6	474	-
40	Гладиатор(Gladiator)	фильм	сша	2000	приключения	155	8,6	474	-
41	Жизнь прекрасна(La vita è bella)	фильм	италия	1997	военный	116	8,6	381	-
42	Жизнь прекрасна(La vita è bella)	фильм	италия	1997	комедия	116	8,6	381	-
43	Жизнь прекрасна(La vita è bella)	фильм	италия	1997	драма	116	8,6	381	-
44	Жизнь прекрасна(La vita è bella)	фильм	италия	1997	мелодрама	116	8,6	381	-
45	Бойцовский клуб(Fight Club)	фильм	сша	1999	триллер	139	8,6	361	-
46	Бойцовский клуб(Fight Club)	фильм	сша	1999	драма	139	8,6	361	-
47	Бойцовский клуб(Fight Club)	фильм	сша	1999	криминал	139	8,6	361	-
48	Властелин колец: Братство Кольца(The Lord of	фильм	новая зеландия	2001	фэнтези	178	8,6	328	-
49	Властелин колец: Братство Кольца(The Lord of	фильм	новая зеландия	2001	приключения	178	8,6	328	-
50	Властелин колец: Братство Кольца(The Lord of	фильм	новая зеландия	2001	драма	178	8,6	328	-
51	Карты, деньги, два ствола(Lock, Stock and Two	фильм	великобритания	1998	боевик	107	8,6	522	-
52	Карты, деньги, два ствола(Lock, Stock and Two	фильм	великобритания	1998	комедия	107	8,6	522	-
53	Карты, деньги, два ствола(Lock, Stock and Two	фильм	великобритания	1998	криминал	107	8,6	522	-
54	Властелин колец: Две крепости(The Lord of the	фильм	новая зеландия	2002	фэнтези	179	8,6	312	-
55	Властелин колец: Две крепости(The Lord of the	фильм	новая зеландия	2002	приключения	179	8,6	312	-
56	Властелин колец: Две крепости(The Lord of the	фильм	новая зеландия	2002	драма	179	8,6	312	-
57	Гладиатор(Gladiator)	фильм	сша	2000	история	155	8,6	474	-
58	Гладиатор(Gladiator)	фильм	сша	2000	боевик	155	8,6	474	-
59	Гладиатор(Gladiator)	фильм	сша	2000	драма	155	8,6	474	-
60	Гладиатор(Gladiator)	фильм	сша	2000	приключения	155	8,6	474	-
61	Темный рыцарь(The Dark Knight)	фильм	сша	2008	фантастика	152	8,5	111543	-
62	Темный рыцарь(The Dark Knight)	фильм	сша	2008	боевик	152	8,5	111543	-
63	Темный рыцарь(The Dark Knight)	фильм	сша	2008	триллер	152	8,5	111543	-
64	Темный рыцарь(The Dark Knight)	фильм	сша	2008	криминал	152	8,5	111543	-
65	Темный рыцарь(The Dark Knight)	фильм	сша	2008	драма	152	8,5	111543	-

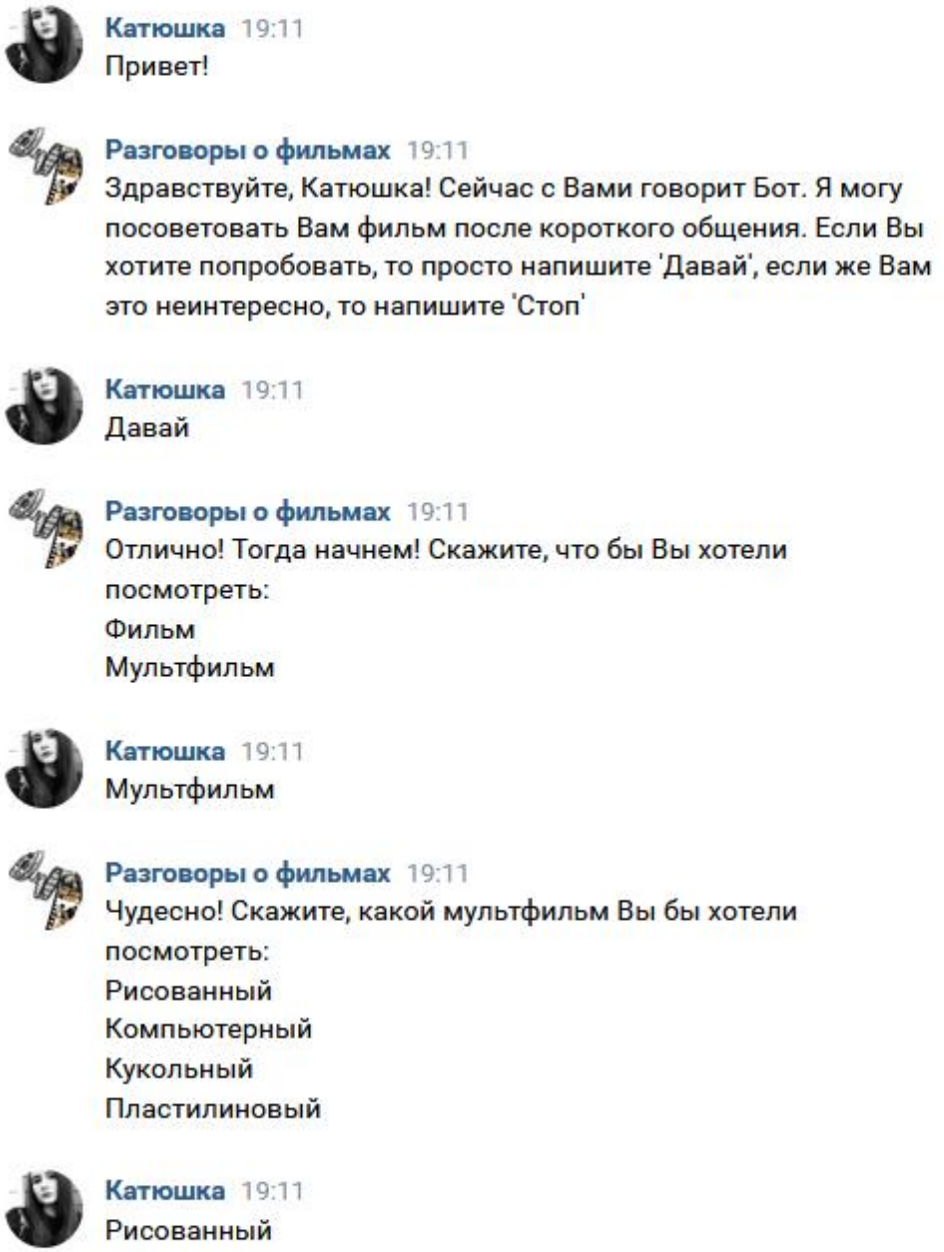
Приложение 6. База данных 3.

66	12 разгневанных мужчин(12 Angry Men)	фильм	сша	1956	драма	96	8,5	346	-
67	12 разгневанных мужчин(12 Angry Men)	фильм	сша	1956	детектив	96	8,5	346	-
68	12 разгневанных мужчин(12 Angry Men)	фильм	сша	1956	криминал	96	8,5	346	-
69	Унесённые призраками(Sen to Chihiro no kamik	мультфильм	япония	2001	аниме	125	8,5	370	рисованный
70	Унесённые призраками(Sen to Chihiro no kamik	мультфильм	япония	2001	фэнтези	125	8,5	370	рисованный
71	Унесённые призраками(Sen to Chihiro no kamik	мультфильм	япония	2001	приключения	125	8,5	370	рисованный
72	Унесённые призраками(Sen to Chihiro no kamik	мультфильм	япония	2001	семейный	125	8,5	370	рисованный
73	Пианист(The Pianist)	фильм	франция	2002	драма	149	8,5	355	-
74	Пианист(The Pianist)	фильм	франция	2002	военный	149	8,5	355	-
75	Пианист(The Pianist)	фильм	франция	2002	биография	149	8,5	355	-
76	Пианист(The Pianist)	фильм	франция	2002	музыка	149	8,5	355	-
77	В джазе только девушки(Some Like It Hot)	фильм	сша	1959	мелодрама	119	8,5	356	-
78	В джазе только девушки(Some Like It Hot)	фильм	сша	1959	комедия	119	8,5	356	-
79	В джазе только девушки(Some Like It Hot)	фильм	сша	1959	криминал	119	8,5	356	-
80	В джазе только девушки(Some Like It Hot)	фильм	сша	1959	музыка	119	8,5	356	-
81	Поймай меня, если сможешь(Catch Me If You C	фильм	сша	2002	криминал	141	8,5	324	-
82	Поймай меня, если сможешь(Catch Me If You C	фильм	сша	2002	биография	141	8,5	324	-
83	Поймай меня, если сможешь(Catch Me If You C	фильм	сша	2002	комедия	141	8,5	324	-
84	Крестный отец 2(The Godfather: Part II)	фильм	сша	1974	драма	202	8,5	327	-
85	Крестный отец 2(The Godfather: Part II)	фильм	сша	1974	криминал	202	8,5	327	-
86	Пролетая над гнездом кукушки(One Flew Over	фильм	сша	1975	драма	133	8,5	336	-
87	Матрица(The Matrix)	фильм	сша	1999	фантастика	136	8,5	301	-
88	Матрица(The Matrix)	фильм	сша	1999	боевик	136	8,5	301	-
89	Огни большого города(City Lights)	фильм	сша	1931	комедия	87	8,5	414	-
90	Огни большого города(City Lights)	фильм	сша	1931	мелодрама	87	8,5	414	-
91	Огни большого города(City Lights)	фильм	сша	1931	драма	87	8,5	414	-
92	Игры разума(A Beautiful Mind)	фильм	сша	2001	драма	135	8,5	530	-
93	Игры разума(A Beautiful Mind)	фильм	сша	2001	биография	135	8,5	530	-
94	Игры разума(A Beautiful Mind)	фильм	сша	2001	мелодрама	135	8,5	530	-
95	Эта замечательная жизнь(It's a Wonderful Life)	фильм	сша	1947	фэнтези	130	8,4	348	-
96	Эта замечательная жизнь(It's a Wonderful Life)	фильм	сша	1947	драма	130	8,4	348	-
97	Эта замечательная жизнь(It's a Wonderful Life)	фильм	сша	1947	семейный	130	8,4	348	-

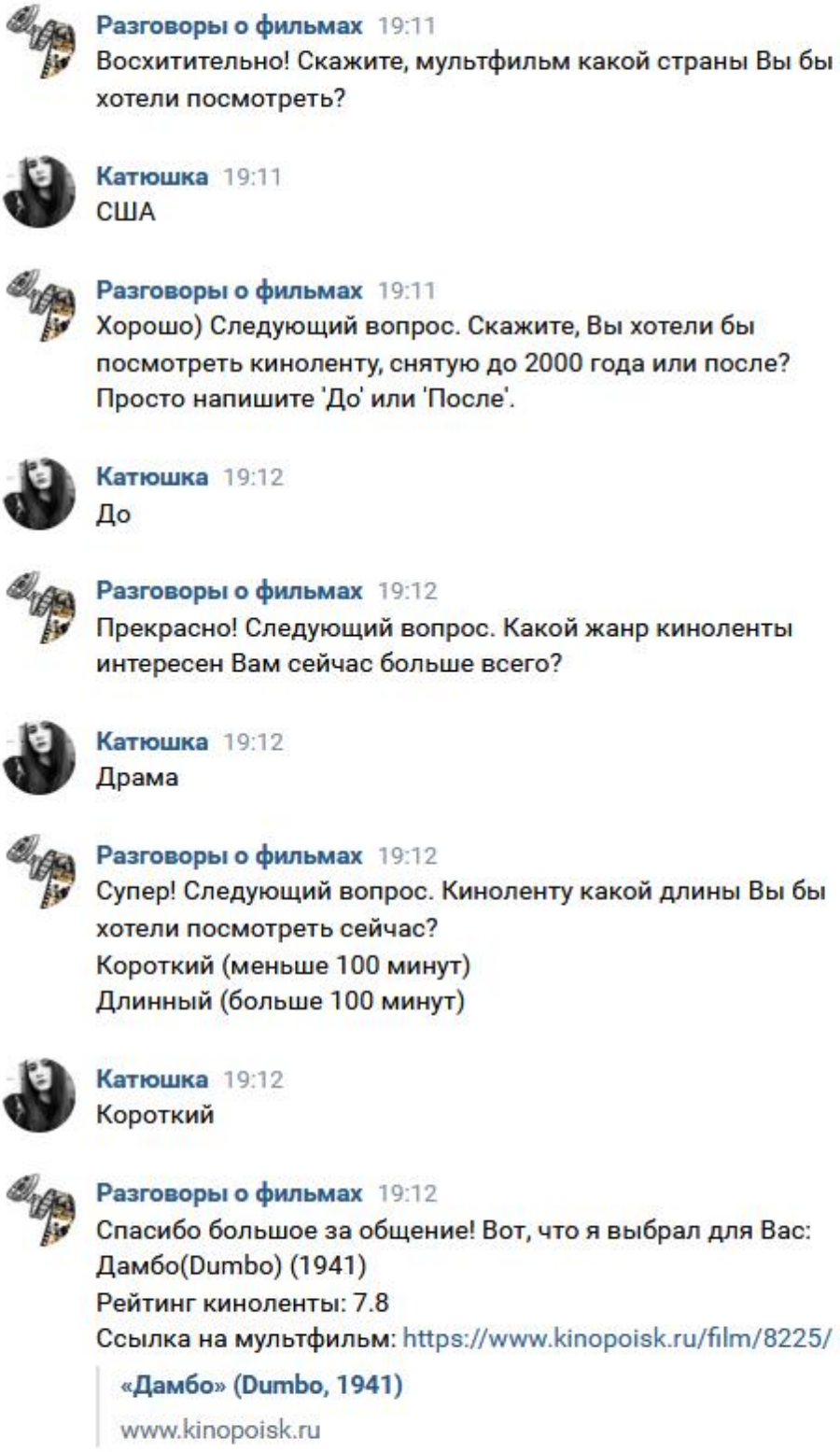
Приложение 7. База данных 4.

98	Унесённые ветром(Gone with the Wind)	фильм	сша	1939	мелодрама	222	8,4	456	-
99	Унесённые ветром(Gone with the Wind)	фильм	сша	1939	история	222	8,4	456	-
100	Унесённые ветром(Gone with the Wind)	фильм	сша	1939	драма	222	8,4	456	-
101	Унесённые ветром(Gone with the Wind)	фильм	сша	1939	военный	222	8,4	456	-
102	Молчание ягнят(The Silence of the Lambs)	фильм	сша	1990	триллер	118	8,3	345	-
103	Молчание ягнят(The Silence of the Lambs)	фильм	сша	1990	детектив	118	8,3	345	-
104	Молчание ягнят(The Silence of the Lambs)	фильм	сша	1990	криминал	118	8,3	345	-
105	Молчание ягнят(The Silence of the Lambs)	фильм	сша	1990	драма	118	8,3	345	-
106	Молчание ягнят(The Silence of the Lambs)	фильм	сша	1990	ужасы	118	8,3	345	-
107	Зверополис(Zootopia)	мультфильм	сша	2016	комедия	108	8,3	775276	компьютерный
108	Зверополис(Zootopia)	мультфильм	сша	2016	криминал	108	8,3	775276	компьютерный
109	Зверополис(Zootopia)	мультфильм	сша	2016	детектив	108	8,3	775276	компьютерный
110	Зверополис(Zootopia)	мультфильм	сша	2016	приключения	108	8,3	775276	компьютерный
111	Зверополис(Zootopia)	мультфильм	сша	2016	семейный	108	8,3	775276	компьютерный
112	Американская история X(American History X)	фильм	сша	1998	драма	119	8,3	382	-
113	Семь(Se7en)	фильм	сша	1995	триллер	127	8,3	377	-
114	Семь(Se7en)	фильм	сша	1995	драма	127	8,3	377	-
115	Семь(Se7en)	фильм	сша	1995	криминал	127	8,3	377	-
116	Семь(Se7en)	фильм	сша	1995	детектив	127	8,3	377	-
117	Храброе сердце(Braveheart)	фильм	сша	1995	история	178	8,3	399	-
118	Храброе сердце(Braveheart)	фильм	сша	1995	биография	178	8,3	399	-
119	Храброе сердце(Braveheart)	фильм	сша	1995	драма	178	8,3	399	-
120	Храброе сердце(Braveheart)	фильм	сша	1995	военный	178	8,3	399	-
121	Терминатор 2: Судный день(Terminator 2)	фильм	сша	1991	фантастика	137	8,3	444	-
122	Терминатор 2: Судный день(Terminator 2)	фильм	сша	1991	боевик	137	8,3	444	-
123	Терминатор 2: Судный день(Terminator 2)	фильм	сша	1991	триллер	137	8,3	444	-
124	Гарри Поттер и философский камень(Harry Pot	фильм	великобритания	2001	фэнтези	152	8,2	689	-
125	Гарри Поттер и философский камень(Harry Pot	фильм	великобритания	2001	приключения	152	8,2	689	-
126	Гарри Поттер и философский камень(Harry Pot	фильм	великобритания	2001	семейный	152	8,2	689	-
127	Как приручить дракона(How to Train Your Drago	мультфильм	сша	2010	фэнтези	98	8,2	280172	компьютерный
128	Как приручить дракона(How to Train Your Drago	мультфильм	сша	2010	комедия	98	8,2	280172	компьютерный
129	Как приручить дракона(How to Train Your Drago	мультфильм	сша	2010	приключения	98	8,2	280172	компьютерный
130	Как приручить дракона(How to Train Your Drago	мультфильм	сша	2010	семейный	98	8,2	280172	компьютерный
131	Спасти рядового Райана(Saving Private Ryan)	фильм	сша	1998	военный	169	8,2	371	-
132	Спасти рядового Райана(Saving Private Ryan)	фильм	сша	1998	драма	169	8,2	371	-

Приложение 8. Диалог(часть 1).



Приложение 9. Диалог(часть 2).



Список литературы

1. Константинова Н. С., Дегтева А. В. Глава 4. Диалоги и чат-боты //Прикладная и компьютерная лингвистика. – 2016. – С. 233-243.

2. Власов М. П., Шимко П. Д. Глава 7.1. Диалоговые системы //Моделирование экономических процессов. – 2005. – С. 120-122.

3. Смыслова Л. В. Чат-бот как современное средство интернет-коммуникаций //Молодой ученый. – 2018. – №. 9. – С. 36-39.

4. Ожегов С. И., Шведова Н. Ю. Толковый словарь Ожегова. – 1949.

5. Мильчин А. Э. Издательский словарь-справочник. – ОЛМА Медиа Групп, 2003.

6. Шауар Б., Этвел Э. Chatbots: они действительно полезны? //Форум LDV. – 2007. – Т. 22. – №. 1. – С. 29-49.

7. Ураев Д. А. Классификация и методы создания чат-бот приложений //International scientific review. – 2019. – №. LXIV. – С. 30-33.

8. Матвеева Н. Ю., Золотарюк А. В. Технологии создания и применения чат-ботов //Научные записки молодых исследователей. – 2018. – №. 1 – С. 28-30.

9. Закалин И. Ю. Автоматизация сбора информации в сети Интернет //Вестник магистратуры. – 2018. – №. 5-4 (80). – С. 32-33.

10. Поликарпов Е. С., Анисимов С. Л., Толстых А. А. О защищенности сайта сети интернет от автоматизированного сбора данных //Вестник Воронежского института МВД России. – 2020. – №. 1 – С. 77-84.

11. Параскевов А. В., Каденцева А. А., Мороз С. И. Перспективы и особенности разработки чат-ботов //Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. – 2017. – №. 130. – С. 395-404.

12. Барашко Е. Н., Васильев А. С., Зубань С. В. Голосовые помощники //Новые импульсы развития: вопросы научных исследований. – 2020. – №. 1-1. – С. 47-53.

13. Будникова А. С., Бабенкова О. С. Использование чат-ботов при изучении иностранного языка //Ученые записки. Электронный научный журнал Курского государственного университета. – 2020. – №. 3 (55). – С. 146-150.

14. Коробкова Е. П., Долгова Т. Г. Актуальность и практическое применение теста Тьюринга //Актуальные проблемы авиации и космонавтики. – 2010. – Т. 1. – №. 6. – С. 420-421.

15. Кузьменко Ф. В. Чат-бот как инновационная система интернет-коммуникаций //ББК 60 Д70 Ответственный редактор: Гуляев Герман Юрьевич, кандидат экономических наук Д70. – 2019. – С. 41-43.

16. Татаркин М.С., Шмелева А.А. Чат-бот в социальной сети Telegram // Мир будущего. — 2018. — С. 1-12.

17. Черноморова Т. С., Воробьев С. П. Классификация и принципы построения систем вопросно-ответного поиска //Бюллетень науки и практики. – 2020. – Т. 6. – №. 8. – С. 145-156.

18. Григорьев К. А. Принципы парсинга html-страниц //Сборник статей по итогам Международной научно-практической конференции 29 декабря 2017 г. – 2017. – С. 30-32.

19. Прохоренок Н., Дронов В. Глава 7. Регулярные выражения //Python 3. Самое необходимое, 2-е изд. – БХВ-Петербург – 2019. – С. 120-139.

20. Zheng C., He G., Peng Z. A Study of Web Information Extraction Technology Based on Beautiful Soup //J. Comput. – 2015. – Т. 10. – №. 6. – С. 381-387.

21. Крапивин Р. Р., Гареева Г. А. Получение доступа к данным путем авторизации в аккаунт с помощью библиотеки Requests в языке Python //Инновационные технологии, экономика и менеджмент в промышленности. – 2021. – С. 206-208.

22. Ong S. P. et al. The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles //Computational Materials Science. – 2015. – Т. 97. – С. 209-215.

23. Кочковая Н. В., Чечевичко Р. С. Разработка сценария для чат-бота //Научный потенциал высшей школы–будущему России. – 2021. – С. 23-26.

24. Мешков В. Е., Прокопенко М. С. Разработка чат-бота помощника на языке python //Научный потенциал высшей школы–будущему России. – 2021. – С. 26-32.

25. McKinney W. et al. pandas: a foundational Python library for data analysis and statistics //Python for high performance and scientific computing. – 2011. – Т. 14. – №. 9. – С. 1-9.

Электронные ресурсы

26. Netology: https://netology.ru/blog/bots-45 (дата обращения: 21.04.2022).

27. Постнаука: https://postnauka.ru/video/82039 (дата обращения: 21.04.2022).

28. Habr: https://habr.com/ru/company/mipt/blog/330228/ (дата обращения: 21.04.2022).

29. Netology2: https://netology.ru/blog/chat-bot-vkontakte (дата обращения: 21.04.2022).

30. Документация: https://vk-api.readthedocs.io/en/latest/vk_api.html (дата обращения: 21.04.2022).

31. Документация ВК: https://vk.com/dev/bots_docs (дата обращения: 21.04.2022).

32. API: https://dementiy.github.io/assignments/vk_api/ (дата обращения: 21.04.2022).

33. Habr2: https://habr.com/ru/company/asus/blog/404505/ (дата обращения: 21.04.2022).

34. BeautifulSoup: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата обращения: 10.05.2022).

35. XlsxWriter: https://xlsxwriter.readthedocs.io/worksheet.html (дата обращения: 10.05.2022).

36. Requests: https://pythonru.com/biblioteki/kratkoe-rukovodstvo-po-biblioteke-python-requests (дата обращения: 10.05.2022)
37. Regular Expressions: https://docs.python.org/3/library/re.html (дата обращения: 10.05.2022)