Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

КУРСОВАЯ РАБОТА

РАЗРАБОТКА ИНТЕРАКТИВНОГО ПОМОЩНИКА ДЛЯ ОСУЩЕСТВЛЕНИЯ РОЗНИЧНЫХ ПОКУПОК

DEVELOPMENT OF AN INTERACTIVE SHOPPING ASSISTANT FOR RETAIL PURCHASES

по направлению подготовки 01.04.02 Прикладная математика и информатика

образовательная программа «Интеллектуальные системы и структурный анализ»

Студент группы

Пащенко Никита

(Ф.И.О.)

Руководитель КР

Кандидат технических наук, доцент

Ильвовский Дмитрий Алексеевич

(должность, звание, Ф.И.О.)

Москва, 2021

Содержание

Введение

Персональный интерактивный помощник является популярным направлением в области мобильных технологий. Данная технология позволяет автоматизировать повседневные дела, что делает её крайне эффективной, учитывая современный темп жизни. Большинство пользователей мобильных телефонов используют голосовых помощников с целью создания расписаний, написания заметок, установок различных напоминаний. Составление списка покупок является частным случаем одной из таких задач и, учитывая тот факт, что на российском рынке голосовые помощники появились относительно недавно, актуальным является реализация приложения, позволяющего вести список покупок, используя русский язык.

Таким образом, курсовая работа посвящена разработке интерактивного голосового помощника для осуществления розничных покупок. В ходе работы рассмотрены существующие платформы для реализации голосовых чат-ботов и возможные подходы к персональным рекомендациям.

Разработана система, позволяющая вести список покупок, отслеживать стоимость товаров и предлагать рекомендации на основе предыдущих покупок пользователя и его текущего списка товаров. Проведено тестирование системы на основе базовых запросов. Согласно полученным результатам, разработанная система позволяет успешно обрабатывать большинство запросов на добавление, удаление и обновление списка покупок, система умеет достаточно точно рекомендовать товары на основе рецептов и истории покупок пользователя.

Выбор голосового помощника

Голосовой помощник — программное обеспечение, позволяющее управлять мобильным устройством или компьютером посредством голосовых команд. Современный голосовой помощник может упростить поиск информации в Интернете, запустить различные системные функции и приложения, или выступать в роли виртуального собеседника.

На сегодняшний день наблюдается тенденция к закреплению за популярными операционными системами собственных голосовых помощников. Так, на iOS штатным ассистентом является программа Siri, на Android — Google Assistant, на Windows — Cortana. Практически каждая крупная компания-разработчик внедряет в свои девайсы собственные реализации виртуальных помощников. На рисунке ниже приведена сравнительная информация по самым популярным голосовым помощникам.

Сводная таблица характеристик

Особенности	Siri	Google Assistant	Alexa	Алиса	Cortana
Год выпуска	2010	2012	2014	2017	2014
Операционная система	iOS	Android | iOS	Android | iOS | Fire TV	Android | iOS | Windows	Windows | Android | iOS | Xbox One
Русский язык	+	+	-	+	-
API	SiriKit	Google Assistant SDK	Alexa Skills Kit | Alexa Voice Service	Yandex IO	Cortana Skills Kit | Cortana SDK
"Умные" колонки	-	Google Home	Amazon Echo	Яндекс.Станция	Harman Kardon Invoke
Режим диалога	+	+/- (не в Google Now)	+	+	+
Взаимодействие с приложениями	+/-	+/-	Только запуск	Запуск и приложения Яндекс	Только запуск

Рисунок 1 – Характеристики популярных голосовых помощников

Ведущими голосовыми помощниками в мире все же являются именно зарубежные, такие как Amazon Alexa и Google Assistant, но у данных голосовых помощников либо нет поддержки русского языка, либо возникают с ним серьезные

проблемы. Например, Google Assistant довольно часто ставит ударения не там, где нужно, а Alexa в целом не поддерживает русский язык. Поэтому, так как наша система подразумевает использование русского языка как средство общения между пользователем и приложением, в качестве платформы для разработки был выбран голосовой помощник Алиса от компании «Яндекс».

Алиса

Алиса — виртуальный голосовой помощник, созданный компанией «Яндекс». Распознает естественную речь, имитирует живой диалог, даёт ответы на вопросы пользователя и, благодаря запрограммированным навыкам, решает прикладные задачи. Алиса работает на смартфонах и компьютерах, в автомобилях и в Яндекс.Станции.

Распознать голосовой запрос Алисе помогает технология SpeechKit. На этом этапе происходит отделение голоса от фоновых шумов. Разобраться с акцентами, диалектами, сленгами и англицизмами алгоритмам позволяет накопленная «Яндексом» база из миллиарда произнесённых в разных условиях фраз [1].

На следующем этапе наделить запрос смыслом и подобрать правильный ответ позволяет технология Turing. Благодаря ей Алиса может не только давать ответы на конкретные вопросы, но и общаться с пользователем на отвлечённые темы. Для этого текст запроса дробится на токены, как правило это отдельные слова, которые в дальнейшем отдельно анализируются. Для максимально точного ответа Алиса учитывает историю взаимодействия с ней, интонацию запроса, предыдущие фразы и геопозицию. Это объясняет тот факт, что на один вопрос разные пользователи могут получить разные ответы [2].

Первоначально нейросеть Алисы обучали на массиве текстов из классики русской литературы, включая произведения Льва Толстого, Фёдора Достоевского, Николая Гоголя, а затем на массивах живых текстов из Интернета [3].

Последним этапом является озвучивание ответа, реализуется с помощью технологии Text-to-speech. Основой служат записанные в студии 260 тысяч слов и фраз, которые затем были порезаны на фонемы. Из этой аудиобазы нейросеть собирает ответ, после чего интонационные перепады в готовой фразе сглаживаются нейросетью, что приближает речь Алисы к человеческой [4].

Помимо сервисов «Яндекса», в Алису могут быть интегрированы сторонние сервисы через систему навыков, использующих платформу голосового помощника для взаимодействия с пользователем. Навыки — это чат-боты и другие интернет-сервисы, активирующиеся по ключевой фразе и работающие в интерфейсе Алисы.

Процесс создания навыка для Алисы

Реализация навыков для Алисы происходит с помощью платформы Яндекс.Диалоги. Яндекс.Диалоги интегрируются по протоколу HTTPS по методу webhook. Формат взаимодействия включает в себя запросы и ответы, которыми Яндекс.Диалоги обмениваются с серверами навыков. Получив реплику пользователя от Алисы, Яндекс.Диалоги переправляют текст обработчику навыка. Текстом может быть как сказанная пользователем фраза, так и различное графическое содержимое (кнопки или изображения). Получив реплику пользователя, Яндекс.Диалоги отправляют POST-запрос на Webhook URL, указанный при публикации навыка [5].

Метод запроса POST предназначен для запроса, при котором веб-сервер принимает данные, заключённые в тело сообщения, для хранения. По сути сервер навыка получает json-файл с набором информации по пользователю и его запросу.

С точки зрения пользователя, навык — это специализированный режим Алисы, который вызывается активационным именем. В этом режиме Алиса транслирует реплики пользователя на сервер навыка, и отвечает переданным текстом, ссылками или подсказками, но во время использования навыка у

пользователя нет возможности получить доступ ко встроенному функционалу Алисы. Таким образом, Алиса в данном формате общения с пользователем лишь принимает запросы пользователя и озвучивает отправленный обработчиком навыка ответ, то есть Алиса не сможет отвечать на те запросы, которые не обрабатываются самим навыком. Более того нет возможности получить какую-либо информацию о пользователе, кроме той, что присылается json-файлом. Например, Алиса в свободном режиме может определить геолокацию пользователя или определить стоимость товара, но при работе с навыком нет какого-либо инструмента, позволяющего запросить у Алисы данную информацию, поэтому при реализации навыка не получится воспользоваться готовыми решениями, придется реализовывать их с нуля.

Таким образом, с технической точки зрения, навык — это веб-приложение, которое принимает токенизированные реплики пользователя в виде json-файлов и отправляет обратно на платформу Яндекс.Диалогов ответ, который в дальнейшем Алиса озвучит пользователю. Веб-сервис можно писать на любом удобном языке программирования или веб-фреймворке, единственные требования к работе веб-приложения — это корректный json-формат ответа и время ответа навыка не должно превышать 3 секунды.

Серверная часть веб-приложения

Flask

Для написания серверной части навыка для Алисы был выбран микрофреймворк Flask. Flask — это легкий фреймворк для создания веб-приложений, написанный на языке Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2 [6]. В нашем случае нам необходим лишь функционал по работе с POST-запросами, поэтому необходимости в более продвинутых фреймворках не имеется.

База данных

Для хранения информации о составленном списке покупок пользователей была использована база данных (БД). БД представляют собой набор структур для хранения больших объемов информации и программных модулей, которые осуществляют управление данными. В нашем случае была использована реляционная БД, так как наши данные имеют четко выраженную структуру. В качестве системы управления базами данных (СУБД) выбрана бесплатная PostgreSQL, но рассматриваемые далее приемы применимы практически ко всем СУБД, в которых используется SQL язык запросов.

В БД были реализованы следующие таблицы:

- *shopping_list* — хранение информации о списке покупок пользователей;
- *product_prices* — хранение информации о цене товаров;
- *recipes* — хранение информации о рецептах.

Давайте рассмотрим структуру каждой из реализованных таблиц. Таблица *shopping_list* имеет следующие поля:

- *id* — индекс*;*
- *user_id* — уникальный id пользователя;
- *product* — добавленный пользователем товар;
- *quantity* — количество товара;
- *created_on* — дата последнего изменения;
- *units* — единица измерения товара;
- *frequency[]* — список частот рекомендаций товара;
- *recs[]* — список рекомендованных пользователю рецептов.

Структура описанной выше таблицы представлена на рисунке 2.

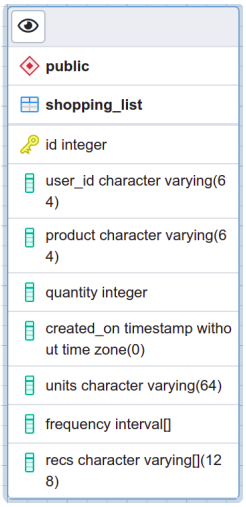

Рисунок 2 – Структура таблицы *shopping_list*

Таблица *product_prices* имеет следующие поля:

- *product* — продукт;
- *price* — цена продукта.

Структура рассмотренной выше таблицы представлена на рисунке 3.

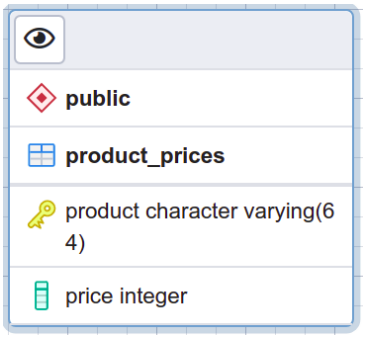

Рисунок 3 – Структура таблицы *product_prices*

Таблица *recipes* имеет следующие поля:

- *index* — ссылка на рецепт;
- *title* — название рецепта;
- *ingredients* — набор ингредиентов для рецепта;
- *category* — категория рецепта.

Структура таблицы *recipes* представлена на рисунке 4.

	title	ingredients	category
/recepty/zakuski/ogurci-v-ostrom-marinade-16230	огурцы в остром маринаде	[рисовый уксус, огурец, кунжутное масло, соевы...	соленья и консервация
/recepty/zakuski/solenie-limoni-29601	соленые лимоны	[лимон, чёрный перец горошек, лавровый лист, с...	соленья и консервация
/recepty/sousy-marinady/marinovannyy-chesnok-so-svekloy-137101	маринованный чеснок со свеклой	[чеснок, вода, уксус 9%-ный, соль, чёрный пере...	соленья и консервация
/recepty/zagotovki/marinovannye-opyata-94048	маринованные опята	[опёнок, уксус, соль, лавровый лист, сахар, чё...	соленья и консервация
/recepty/zakuski/marinovannyy-arbuz-38167	маринованный арбуз	[соль, арбуз, стебель сельдерей, чеснок, укроп...	соленья и консервация
...	...	...	...
/recepty/sousy-marinady/tkemali-iz-krasnoy-smorodiny-80062	ткемали из красной смородины	[красная смородина, кинза, семя кориандра, тми...	ткемали
/recepty/sousy-marinady/sous-tkemali-17107	соус ткемали	[чеснок, слив ткемали, кинза, укроп, красный с...	ткемали
/recepty/sousy-marinady/sous-tkemali--14845	соус «ткемали».	[слив, чеснок, соль, молотый чёрный перец, кинза]	ткемали
/recepty/sousy-marinady/tkemali-iz-vengerki-45772	ткемали из венгерки	[слив, хмель-сунель, уцхо-сунеля, чеснок, семя...	ткемали
/recepty/sousy-marinady/tkemali-zelenij-28613	ткемали зеленый	[кинза, зелёная алыча, укроп, мята, зелёный ст...	ткемали

30771 rows × 3 columns

Рисунок 4 — Структура таблицы *recipes*

Данная таблица была реализована с помощью библиотеки pandas для языка Python. Выбор сделан в сторону pandas с целью упрощения процесса отладки работы алгоритма. В дальнейшем планируется реализовать используемые эвристики для нахождения наиболее подходящего рецепта с помощью языка запросов SQL и соотвественно перенести реализацию таблицы в СУБД PostgreSQL.

Для общение с БД была использована сторонняя библиотека psycopg2 — самый популярный адаптер базы данных PostgreSQL для языка программирования Python. С помощью языка запросов и библиотеки psycopg2 были реализованы функции по добавлению, удалению товаров, отбору подходящих рецептов, получению рекомендаций и т.д. Полный список реализованных функций приведен в файле *utils/db.py*.

Ngrok

Так как платформа Яндекс.Диалоги должна общаться с веб-приложением, а большинство сервисов по развертке серверов либо платные, либо имеют ограниченный функционал и, учитывая тот факт, что разрабатываемое приложение является учебным, то для избежания возможных проблем по интегрированию

серверной части на сторонний сервис было принято решение использовать сервер, реализованный на локальном ПК, и путем использования утилиты ngrok, позволяющей создавать туннели на локальные порты, реализовать общение веб-приложения с платформой Яндекс.Диалоги.

Утилита ngrok — это кроссплатформенное приложение, которое позволяет разработчикам с минимальными усилиями предоставлять доступ к локальному серверу разработки в Интернете. С помощью программы ngrok резервируется случайный публичный адрес (например, 3gf892ks.ngrok.com), все обращения по которому переходят на локальный порт. То есть утилита ngrok создает долговечный TCP-туннель из случайно сгенерированного поддомена на ngrok.com на локальную машину. После указания порта, который прослушивает веб-сервер, клиентская программа ngrok инициирует безопасное соединение с сервером ngrok, а затем любой пользователь может отправлять запросы на локальный сервер с уникальным адресом туннеля ngrok [7].

При размещении приложения с коммерческими целями необходимо будет либо арендовать сервер, например, с помощью сервиса Heroku, либо самостоятельно развернуть веб-приложение на каком-либо домене.

Клиентская часть веб-приложения

С помощью инструментов, предлагаемых платформой Яндекс.Диалоги, был реализован интерфейс для общения пользователя с навыком. Основными функциями навыка являются: добавление, удаление товаров; отображение списка покупок, рекомендаций и стоимости товаров. В следующих главах будут приведены примеры реализаций большинства функций, сейчас лишь рассмотрим те, которые не будут упомянуты далее.

При отображении списка покупок у пользователя есть возможность по нажатию на товар удалить его из списка. Пример вывода списка продуктов представлен на рисунке ниже.

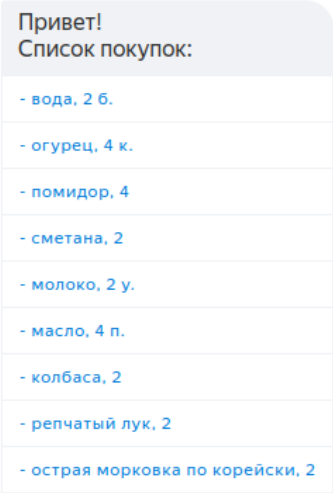

Рисунок 5 – Пример отображения списка покупок

Также у пользователя есть возможность запросить стоимость товаров в корзине. Информация о цене товара хранится в таблице *product_prices*. Если товар не будет найден в этой таблице, то ему будет присвоена некоторая фиксированная стоимость. Пример обработки данного запроса представлен на рисунке 6.

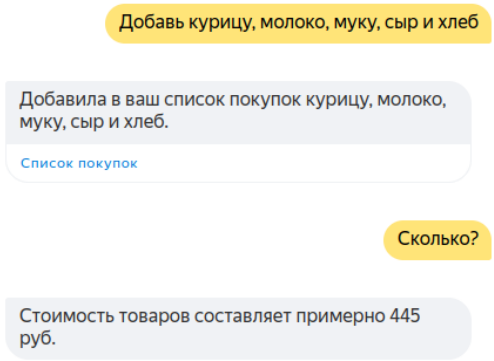

Рисунок 6 — Пример отображения стоимости товаров в корзине

Обработка языковых запросов

Классификация типа запросов пользователя

Давайте подробнее рассмотрим обработку языковых запросов. Яндекс.Диалоги предоставляют свой собственный функционал по работе с естественным языком. Принцип работы может описан следующим образом. После того, как пользователь произносит команду, Яндекс.Диалоги распознают ее текст и извлекают именованные сущности — слова и фразы, которые описывают определенные объекты [5].

На текущий момент Яндекс.Диалоги распознают:

- имена (фамилия, имя, отчество);
- указания на местоположение;
- даты и время;
- целые и дробные числа.

В нашем случае, к сожалению, данные сущности не особо применимы. Однако для упрощения задач NLP (Natural Language Processing) Яндекс.Диалоги предоставляют специальный инструмент — встроенный язык описания пользовательского запроса. С его помощью можно описать правила, по которым Яндекс.Диалоги будут классифицировать запросы и извлекать из них нужные данные, то есть возможность описывать свои собственные именованные сущности. Когда пользователь произносит команду, Яндекс.Диалоги распознают текст и извлекают фразы, которые описывают намерения пользователя согласно созданным правилам. Распознанные сущности Яндекс.Диалоги присылают обработчику навыка.

Чтобы формализовать разбор реплик пользователя, Яндекс.Диалоги используют интенты, формы и слоты. Интент — это задача, которую пользователь формулирует в конкретной реплике. Форма — контейнер с информацией, который

Яндекс.Диалоги заполняют, распознавая запрос пользователя. Слот — поле формы. Каждый слот имеет название, тип данных и признак обязательности [5].

При обработке реплики Яндекс.Диалоги сначала определяют, к какому интенту она относится. После этого извлекают из реплики необходимые параметры и заполняют ими слоты формы. Распознанные данные Яндекс.Диалоги отправят в навык.

Используя выше описанные синтаксис были созданы интенты, представленные на рисунке ниже.

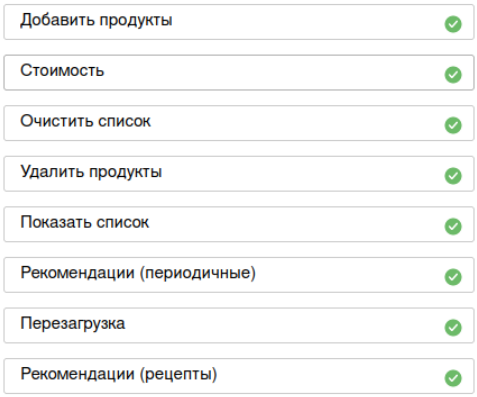

Рисунок 7 – Реализованные интенты на платформе Яндекс.Диалоги

На рисунке 8 приведен пример одного из созданных интентов.

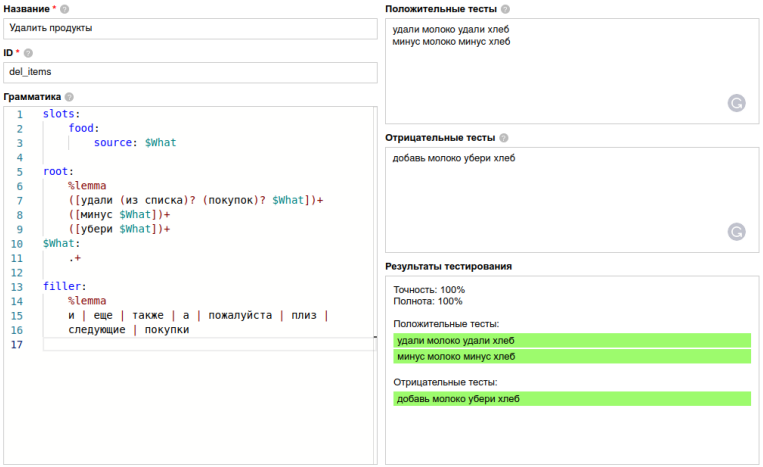

Рисунок 8 – Пример реализованного интента

Как видно из рисунка 8, создание интентов позволяет при обработке запроса пользователя фильтровать стоп-слова, лемматизировать текст запроса и получать тип запроса, т.е. название определённого системой интента. Таким образом, классификация запроса пользователя происходит с помощью встроенных инструментов сервиса Яндекс.Диалоги, а непосредственная обработка запроса реализована с помощью веб-приложения.

Обработка запросов пользователя

После получения системой запроса от пользователя в наш навык будет отправлен json-файл, содержащий в том числе токены и название интента. Пример представления необходимой нам информации представлен на рисунке ниже.

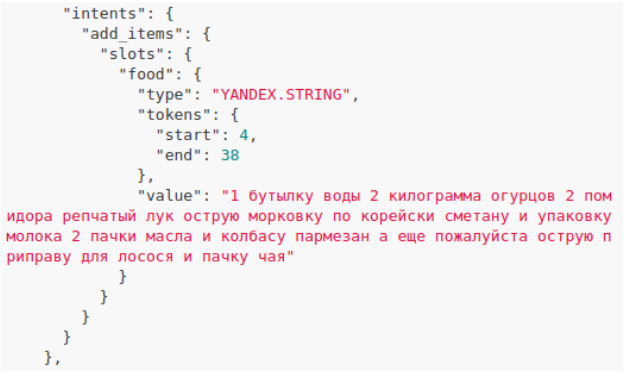

Рисунок 9 – Пример получаемого навыком json-файла

То есть мы получили следующую информацию. Во-первых, запрос относится к типу *add_items*, во-вторых, в поле *value* мы имеем информацию о товарах, которые пользователь хотел добавить в свой список покупок. Таким образом, необходимо реализовать обработку запроса пользователя в зависимости от типа полученного интента. В каждом из интентов токены будут представлять собой информативную часть запроса пользователя, то есть непосредственно продукты, их количество и единицы измерения.

Для обработки запросов пользователя на языке Python была использована библиотека pymorphy2. С помощью которой мы можем реализовать следующий необходимый нам функционал:

1. Приведение слова к нормальной форме. Данная операция необходима для стандартизированного хранения товара в БД, что исключит дублирование товаров, а также позволит получать корректные рекомендации.
2. Согласование слов по роду, падежу и числу. Данный функционал нам необходим на этапе формирования ответа пользователю.

При работе данного модуля используется словарь OpenCorpora, а для незнакомых слов строятся гипотезы. Также библиотека достаточно быстрая, что позволит укладываться в ограничение в 3 секунды по обработке запроса.

Для обработки запросов был написан простой парсер, который позволил достаточно хорошо обрабатывать все базовые запросы пользователя. Реализованные функции по обработке запросов находятся в файле *utils/parser.py*.

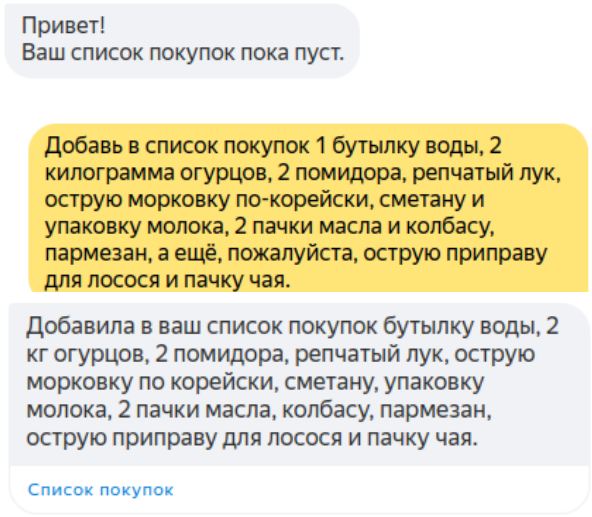

Рисунок 10 – Пример обработки запроса

На рисунке 10 приведен пример достаточно объемного запроса, который распознается системой абсолютно корректно.

В результате чего в БД появляются следующие записи.

	id [PK] integer	user_id character varying (64)	product character varying (64)	quantity integer	created_on timestamp without time zone	units character varying (64)	frequency interval[]	recs character varying[] (128)
1	672	EB345B2E194C7958F5A1D...	вода	1	2021-06-23 01:21:01	бутылка	[null]	[null]
2	673	EB345B2E194C7958F5A1D...	огурец	2	2021-06-23 01:21:01	кг	[null]	[null]
3	674	EB345B2E194C7958F5A1D...	помидор	2	2021-06-23 01:21:01	-1	[null]	[null]
4	675	EB345B2E194C7958F5A1D...	репчатый лук	1	2021-06-23 01:21:01	-1	[null]	[null]
5	676	EB345B2E194C7958F5A1D...	острая морковка по корей...	1	2021-06-23 01:21:01	-1	[null]	[null]
6	677	EB345B2E194C7958F5A1D...	сметана	1	2021-06-23 01:21:01	-1	[null]	[null]
7	678	EB345B2E194C7958F5A1D...	молоко	1	2021-06-23 01:21:01	упаковка	[null]	[null]
8	679	EB345B2E194C7958F5A1D...	масло	2	2021-06-23 01:21:01	пачка	[null]	[null]
9	680	EB345B2E194C7958F5A1D...	колбаса	1	2021-06-23 01:21:01	-1	[null]	[null]
10	681	EB345B2E194C7958F5A1D...	пармезан	1	2021-06-23 01:21:01	-1	[null]	[null]
11	682	EB345B2E194C7958F5A1D...	острая приправа для лосо...	1	2021-06-23 01:21:01	-1	[null]	[null]
12	683	EB345B2E194C7958F5A1D...	чай	1	2021-06-23 01:21:01	пачка	[null]	[null]

Рисунок 11 – Таблица *shopping_list* после запроса на добавление товаров

Для сравнения приведем результаты обработки этого же запроса, используя встроенных функционал Алисы по ведению списка покупок. Как можно заметить из рисунка ниже, не все товары определились корректно.

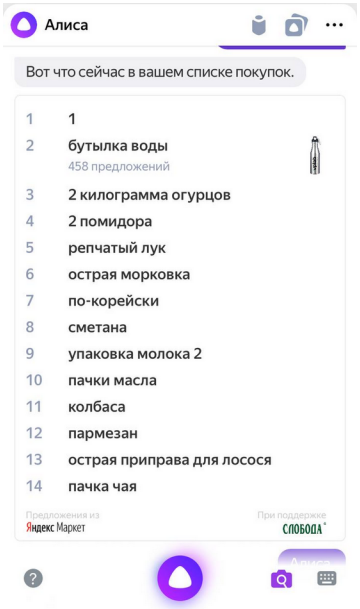

Рисунок 12 – Пример обработки запроса на добавление товаров с помощью встроенного функционала Алисы

Давайте теперь рассмотрим примеры запросов, которые нашей системе не удается распознать корректно, и сравним результаты, полученные с помощью встроенного функционала Алисы и нашей системы.

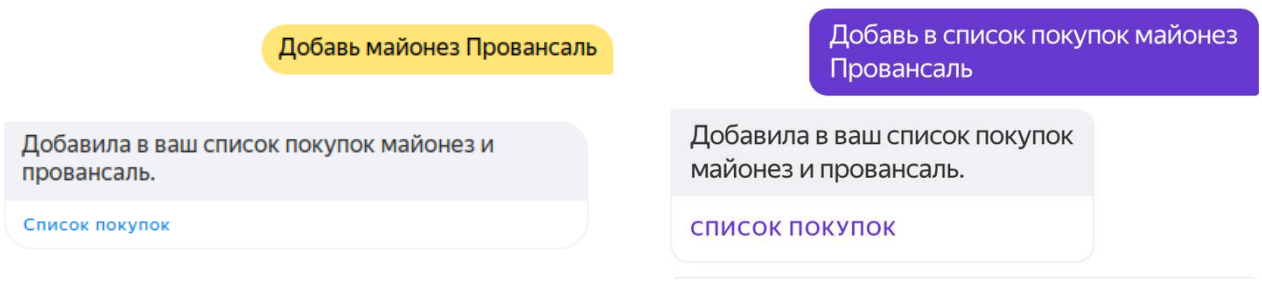

Рисунок 13 – Пример обработки запроса типа «продукт наименование» нашей системой (слева) и встроенным функционалом Алисы (справа)

Как мы видим запросы типа «продукт наименование» распознаются как два отдельных объекта. Из простых решений данной проблемы может быть предложено использование справочника наименований. Из более сложных использование продвинутых систем парсинга, обученных на корпусе соответственной предметной области. Примером подобного модуля служит библиотека Natasha, предназначенная для извлечения именованных сущностей из новостных статей [8].

Следующим запросом, который может быть некорректно распознан нашей системой, являются словосочетания типа «существительное прилагательное». В нашем случае система ожидает, что прилагательное всегда следует до существительного. Пример некорректного ответа на данный запрос представлен на рисунке ниже.

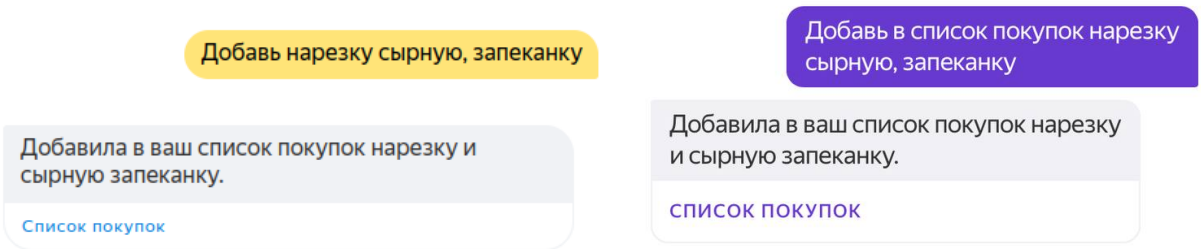

Рисунок 14 – Пример обработки запроса типа «существительное прилагательное» нашей системой (слева) и встроенным функционалом Алисы (справа)

Также у нашей системы нет обработки уменьшительно-ласкательных существительных, то есть запросы, состоящие из одного и того же продукта, но в разных формах, будут восприниматься системой как 2 разных товара. Ниже на рисунке приведен пример подобного запроса.

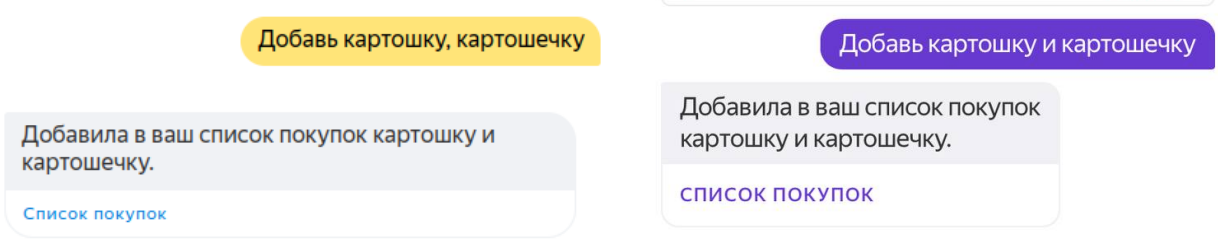

Рисунок 15 – Пример обработки уменьшительно-ласкательных выражений нашей системой (слева) и встроенным функционалом Алисы (справа)

Другим запросом, который системой будет некорректно распознан, является добавление товара, название которого состоит из 2 и более существительных или пишется через дефис.

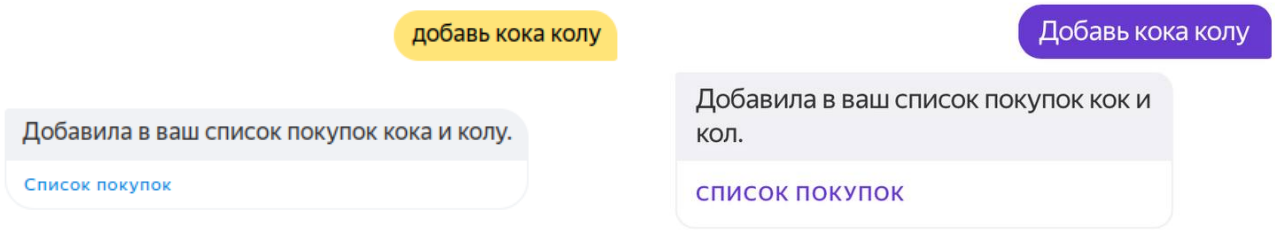

Рисунок 16 – Пример обработки запроса, состоящего из 2 существительных нашей системой (слева) и встроенным функционалом Алисы (справа)

Таким образом, при обработке запросов у системы есть следующие допущения:

- объекты типа «продукт наименование» не могут быть корректно обработаны, система определит продукт и наименование как 2 различных продукта;
- количество следует до товара;

- при определении какого-либо продукта прилагательное должно следовать до существительного;
- объекты, состоящие из двух слов, будут обработаны как 2 отдельных продукта;
- единица измерения должна быть между количеством и продуктом.

Рекомендательные системы

Существуют две основные стратегии создания рекомендательных систем — фильтрация на основе содержания и коллаборативная фильтрация. При коллаборативной фильтрации используются известные предпочтения группы пользователей для прогнозирования неизвестных предпочтений другого пользователя [9]. Так как в нашей системе не подразумевается наличия какой-либо системы оценивания, то мы будем использовать альтернативную стратегию рекомендательных систем — фильтрацию на основе содержания.

При фильтрации на основе содержания создаются профили пользователей и объектов. В нашем случае собранные товары в корзине как раз и являются профилем некоторых рецептов. Таким образом, рекомендательная система на основе составленной корзины пользователя предложит максимально подходящие рецепты блюд, что за собой подразумевает покупку товаров, которых нет в текущей корзине пользователя, но они также необходимы для приготовления рекомендованных блюд.

Рекомендательная система на основе рецептов

Для рекомендательной системы, цель которой предложить пользователю рецепты блюд, которые лучше всего подходят для его текущей корзины, необходимо иметь некоторую базу рецептов.

Первым решением было попробовать найти готовый датасет рецептов. Были рассмотрены несколько вариантов. Датасет Recipe1M+ [10] является крупнейшим

среди своих аналогов. Данный датасет разрабатывался для задачи определения рецепта блюда по картинке, поэтому он содержит излишнюю для нас разметку, но в свободном доступе имеется датасет simplified-recipes-1M [11] — измененная версия Recipe1M+, где была оставлена информация только непосредственно по рецептам и их ингредиентам (убраны любые разметки, связанные с картинкам и т.д.).

Датасет simplified-recipes-1M содержит около миллиона тщательно очищенных и предварительно обработанных рецептов. Основная часть собранных данных была взята из пяти различных датасетов, самым крупным из которых является Recipe1M+. Они были объединены для создания более полной коллекции рецептов.

К сожалению, данный датасет является англоязычным, что не совсем подходит для реализации нашей системы, в которой подразумевается общение с пользователем на русском языке. Рецепты и ингредиенты можно вручную перевести, что позволит интегрировать датасет в работу нашей системы. Но было решено для проверки самого концепта работы нашей рекомендательной системы найти русскоязычный датасет.

На платформе Kaggle был найден датасет «eating dataset from eda.ru». Как видно из названия, в данном датасете используется информация, собранная на сайте «eda.ru», где представлено около 30000 рецептов.

К сожалению, в ходе работы было обнаружено, что датасет содержит большое число ошибок (систематические дублирования ингредиентов в рецептах, неправильное указание их граммовок и т.д.), поэтому предсказания по некоторым рецептах могут некорректно отрабатывать. Было принято решение получить данные по рецептам с сайта «eda.ru» самостоятельно. Таким образом, был вручную собран датасет, состоящий из 30771 рецептов. Код, используемый для парсинга сайта и дальнейшей постобработки представлен в файлах

/notebooks/eda_parsing.ipynb и */notebooks/eda_preprocessing.ipynb*. Полученная таблица рецептов представлена на рисунке ниже.

	title	ingredients	category
/recepty/zakuski/ogurci-v-ostrom-marinade-16230	огурцы в остром маринаде	[рисовый уксус, огурец, кунжутное масло, соевы...	соленья и консервация
/recepty/zakuski/solenie-limoni-29601	соленые лимоны	[лимон, чёрный перец горошек, лавровый лист, с...	соленья и консервация
/recepty/sousy-marinady/marinovannyy-chesnok-so-svekloy-137101	маринованный чеснок со свеклой	[чеснок, вода, уксус 9%-ный, соль, чёрный пере...	соленья и консервация
/recepty/zagotovki/marinovannye-opyata-94048	маринованные опята	[опёнок, уксус, соль, лавровый лист, сахар, чё...	соленья и консервация
/recepty/zakuski/marinovannyy-arbuz-38167	маринованный арбуз	[соль, арбуз, стебель сельдерей, чеснок, укроп...	соленья и консервация
...	...	...	...
/recepty/sousy-marinady/tkemali-iz-krasnoy-smorodiny-80062	ткемали из красной смородины	[красная смородина, кинза, семя кориандра, тми...	ткемали
/recepty/sousy-marinady/sous-tkemali-17107	соус ткемали	[чеснок, слив ткемали, кинза, укроп, красный с...	ткемали
/recepty/sousy-marinady/sous-tkemali--14845	соус «ткемали».	[слив, чеснок, соль, молотый чёрный перец, кинза]	ткемали
/recepty/sousy-marinady/tkemali-iz-vengerki-45772	ткемали из венгерки	[слив, хмель-сунель, уцхо-сунеля, чеснок, семя...	ткемали
/recepty/sousy-marinady/tkemali-zelenij-28613	ткемали зеленый	[кинза, зелёная алыча, укроп, мята, зелёный ст...	ткемали

30771 rows × 3 columns

Рисунок 17 – Датасет рецептов, собранный с сайта «eda.ru»

Далее был реализован жадный алгоритм поиска максимально подходящих рецептов на основе составленного пользователем списка покупок. Алгоритм работы следующий:

1. Система прогоняет список продуктов из корзины пользователя по каждому из рецептов в БД и ищет максимальные относительные пересечения среди них. Поясняющая формула приведена ниже:

$$\arg\max_{Recipe \in Recipes} \frac{|Recipe \cap Basket|}{|Recipe|}$$

2. После первой итерации выбирается наилучший результат среди рецептов без учета полных совпадений. И из рассматриваемой корзины убираются все продукты, которые являются ингредиентами для выбранного рецепта.
3. Далее алгоритм повторяет свою работу для оставшихся в корзине товаров.
4. Алгоритм заканчивает свою работу, если наберется 5 рецептов или если в корзине не останется неиспользованных продуктов.

В результате система предлагает пользователю отобранные рецепты в виде ссылок на сайт «eda.ru», откуда они и были взяты. Пример работы алгоритма представлен на рисунке 18.

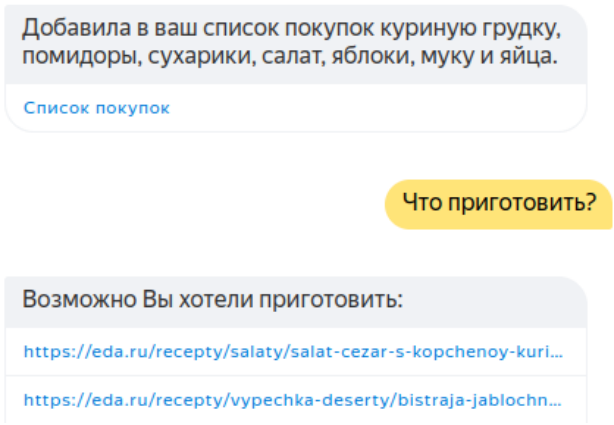

Рисунок 18 – Пример работы рекомендательной системы на основе рецептов

Как видно из рисунка, система порекомендовала приготовить салат цезарь, потому что в списке покупок были куриная грудка, помидоры, сухарики и салат, а также порекомендовала яблочный пирог, так как были яблоки, яйца и мука. Таким образом, данных подход позволяет получить рекомендации по рецептам, учитывая все товары в корзине пользователя.

Альтернативные варианты рекомендательных систем

Альтернативным вариантом реализации рекомендательной системы являются рекомендации, которые позволяют отслеживать периодичность покупаемых товаров и вовремя напоминать о необходимости их купить. Первоначальной идеей решить описанную выше задачу было использование срока годности товаров.

Информацию о сроке годности планировалось получать, используя QR-код, но, как оказалось, данная информация не хранится в таком виде. Другим вариантом была возможность извлечения срока годности товара через онлайн-магазины, но опять же извлечь данную информацию оттуда нельзя. Вариант получения срока годности товара по фотографии товара так же был отвергнут, так как данная информация специально делается слабо читаемой, из-за чего это

сильно затруднит работу алгоритма, что повлечет за собой его тяжеловесную реализацию и возможный большой процент ошибок. Единственной адекватной в рамках мобильного приложения реализацией может быть ручной ввод пользователем срока годности и последующим сохранением этой информации в БД. Описанный выше подход не был реализован в системе, он функционально не удобен в случае использования мобильного телефона, вместо него было предложено альтернативное решение.

Альтернативным вариантом решения задачи о напоминании покупки часто используемых товаров являются рекомендации, основанные на истории покупок пользователя. Иными словами система может отслеживать периодичность покупки того или иного товара и напоминать о его покупке по истечению рассчитанного периода времени. Такой тип рекомендаций отлично выполняет функцию напоминания о покупке товаров первой необходимости, таких как молоко, хлеб и т.д.

Данная функция была реализована следующим образом. В БД мы храним список интервалов между покупками каждого из товаров (*frequency[]* в таблице *shopping_list*), данные интервалы рассчитываются в момент удаления товара из списка покупок: мы вычитаем из текущего времени время последнего изменения количества товара (*created_on* в таблице *shopping_list*). Когда пользователей запросит систему напомнить ему, что сегодня необходимо купить, система сравнит пройденное время с последней покупки товара с рассчитанным интервалом и, если пройденный промежуток времени больше, товар рекомендуется пользователю к покупке.

Усреднение интервалов происходило с использованием медианы. Медиана выбрана с той целью, чтобы минимизировать влияние возможных длительных перерывов в покупке какого-либо товара.

Реализация обоих видов рекомендаций представлена в файле *utils/suggest.py*.

Заключение

Курсовая работа посвящена разработке интерактивного помощника для осуществления розничных покупок. Отсутствие приложения с расширенным функционалом для ведения списка покупок с помощью Алисы показало необходимость разработки системы, позволяющей не только вести список покупок, но и осуществлять персональные рекомендации.

Разработанная система полностью выполняет основной функционал встроенного в Алису приложения по ведению списка покупок, а также предлагает свои уникальные возможности: рекомендации на основе истории покупок пользователя и рекомендации на основе рецептов. Потенциальными пользователями данной системы являются обладатели умных колонок от компании «Яндекс», а также пользователи мобильных телефонов, где Алиса может быть установлена как стороннее приложение.

Код проекта представлен в Git репозитории. Краткое описание реализованных модулей представлено в файле *README.md*.

Дальнейшая разработка будет посвящена реализации продвинутых методов обработки запросов пользователя, реализации рекомендаций на основе оцененных рецептов пользователями с сайта «eda.ru» и исследованию с целью улучшения результатов рекомендаций за счет введения взвешенных коэффициентов для каждого из ингредиентов на основе количества использования этого ингредиента в рецепте.

Список использованных источников

1. Optimization 2018: что находится «под капотом» у Алисы [Электронный ресурс] / Полякова В. Режим доступа: https://www.seonews.ru/analytics/optimization-2018-chto-nakhoditsya-pod-kapotom-u-alisy/, свободный. (дата обращения: 20.06.21)

2. Она. «Яндекс» планировал очередной эксперимент, а получил вполне разумную «Алису» [Электронный ресурс] / Колебакина-Усманова Е. Режим доступа: https://www.business-gazeta.ru/article/383937, свободный. (дата обращения: 20.06.21)

3. Как Алиса и Siri искали миелофон [Электронный ресурс] / Бояркова Г. Режим доступа: https://www.fontanka.ru/2017/10/10/121/, свободный. (дата обращения: 20.06.21)

4. Алиса, скажи что-нибудь. N+1 [Электронный ресурс] / Янгель Б. Режим доступа: https://nplus1.ru/material/2018/02/27/yandex-alice, свободный. (дата обращения: 20.06.21)

5. Алиса (голосовой помощник от компании «Яндекс») [Электронный ресурс] / Режим доступа: https://yandex.ru/dev/dialogs/alice/doc/about.html, свободный. (дата обращения: 20.06.21)

6. Flask (A Python Microframework) [Электронный ресурс] / Armin Ronacher. Режим доступа: http://flask.pocoo.org/, свободный. (дата обращения: 20.06.21)

7. Ngrok (Public URLs for testing) [Электронный ресурс] / Режим доступа: https://www.pubnub.com/learn/glossary/what-is-ngrok/, свободный. (дата обращения: 20.06.21)

8. Проект Natasha. Набор качественных открытых инструментов для обработки естественного русского языка (NLP) [Электронный ресурс] / Кукушкин А. Режим доступа: https://habr.com/ru/post/516098/ , свободный. (дата обращения: 20.06.21)

9. Koren, Y. Matrix Factorization Techniques for Recommender Systems // Computer. — IEEE. — Т. 42, № 8. — С. 30—37.

10. Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images [Электронный ресурс] / Salvador, Amaia and Hynes. Режим доступа: http://pic2recipe.csail.mit.edu/, свободный. (дата обращения: 20.06.21)

11. Dataset simplified-recipes-1M (recipe-ingredient dataset) [Электронный ресурс] / Dominik Schmidt Режим доступа: https://dominikschmidt.xyz/simplified-recipes-1M/ , свободный. (дата обращения: 20.06.21)