РАЗРАБОТКА АРХИТЕКТУРЫ ИНТЕЛЛЕКТУАЛЬНОГО ЧАТ-БОТА

Анна Литвин, Виталий Величко, Владислав Каверинский

***Аннотация**: В статье рассмотрена общая концепция приложений типа чат-ботов, развитие идеи чат-бота и их практического значения. Проведен анализ существующих на данный момент современных чат-ботов и платформ для их построения. Проведена классификация чат-ботов по способу создания, по архитектуре и по принципу выдачи ответа. Предложена концепция интеллектуального чат-бота, совмещающего в себе возможность интеграции с популярными месседжерами и обновляемую на основании потребностей пользователей базу знаний.*

***Ключевые слова**: чат-бот, анализ естественно-языкового текста, генерация осмысленного текста, база знаний.*

***ITHEA Keywords**: D.2.11 Software Architectures*

Введение

Идея чат-бота – виртуального собеседника в целом не нова. История чат-ботов восходит к 1966 году, когда Вейзенбаум написал компьютерную программу под названием ELIZA. Она имела всего 200 строк кода [Weizenbaum, 1976]. На сегодняшний день чат-боты прошли эволюцию от достаточно примитивных программ с минимальной функциональностью, основанных исключительно на заложенных в них шаблонах и ограниченном малом наборе правил, до продвинутых программ, не только общающихся с пользователем, но и автоматизирующих выполнение рутинных задач.

Для внешней оболочки чат-бота многие системы обмена сообщениями (меседжеры) предоставляют готовые решения и API (Application Programming Interface) для работы с ними. Создание внутреннего логического содержания чат-бота является задачей более сложной. В любом случае, перед разработчиком чат-бота (особенно интеллектуального чат-бота) стоит задача

построения системы, работающей с естественным языком (в нашем случае – со славянскими флективными языками) и базой знаний, способной адекватно анализировать поступающие сообщения и генерировать ответы. Фактически, чат-бот должен в той или иной мере реализовывать идею «Текст → Смысл → Текст» [Melchuk, 1995]. То есть, программа должна из поступающего текста выделить смысл сообщения и в соответствии с этим смыслом предоставить адекватный ответ. Но несмотря на то, что компьютерные системы достигли немалых успехов в задачах распознания, проблемы понимания смысла всё ещё остаются не до конца решёнными. Одним из перспективных вариантов средства работы со смыслом текста является онтология.

Постановка задачи

Основными целями данной работы является обзор принципов работы и архитектуры известных чат-ботов и платформ для их создания, проведение классификации чат-ботов, построение архитектуры интеллектуального чат-бота, работающего с пополняемой (обучаемой) базой знаний, основанной на онтологии.

Обзор систем и подходов разработки чат-ботов

Чат-бот – это программный продукт на основе искусственного интеллекта, действующий в устройстве, приложении или веб-сайте, который пытается оценить потребности пользователей, а затем помочь им выполнить определенную задачу, например, коммерческую транзакцию, бронирование гостиницы, получение информации о продукции и многое другое. На данный момент существуют множество чат-ботов, построенных на основе парадигмы обработки текста пользователя и возвращения релевантного ответа.

Все чат-боты можно принципиально разделить на те, которые основаны на чётко установленных правилах и на обучаемые. В первом случае чат-бот отвечает на вопросы, основываясь на некоторых, заложенных в него правилах. Это вовсе не означает примитивность такого бота. Заложенные правила могут быть как очень простыми, так и очень сложными. Он просто статичен, не способен к сбору новой информации и её интерпретации [SiteActive]. Обучаемый чат-бот использует в своей работе подходы, основанные на

машинном обучении. По принципу выдачи ответа чат-боты можно разделить на два класса: основанные на поиске и на генерации [Melnichuk, 2018].

В моделях, основанных на поиске, чат-бот осуществляет выбор ответа из библиотеки предопределенных ответов основываясь на анализе контекста. Не следует полагать, что в данном подходе предполагается только набор полностью заранее заготовленных фраз. Шаблоны могут иметь некоторую гибкость в плане подстановки понятий, на основе анализа конкретного контекста. Контекст может включать в себя текущую позицию в дереве диалога, все предыдущие сообщения в диалоге, ранее сохраненные переменные (например, имя пользователя). Опираясь на анализ оперативно полученных в ходе диалога данных, в имеющиеся шаблоны встраивается необходимая информация. Боты, основанные на генерации способны генерировать оригинальные ответы без использования заготовленных шаблонов [Glek, 2018].

Следует отметить, что в последнее время приобретают популярность чат-боты, работа которых основана на нейронных сетях с применением Deep Learning [Tarasov, 2015]. Такие боты с первого взгляда могут произвести хорошее впечатление. Но детальный анализ и работа с ними позволяет выделить некоторые их существенные недостатки. В первую очередь – это медленная скорость обучения. Чтобы обучить чат-бота, основанного на нейронных сетях, требуется много времени и большие объёмы данных. Это не всегда возможно и рационально [Levin, 2016].

Интересной особенностью чат-ботов, работающих на основе нейронных сетей это то, что они испытывают трудности с тем, чтобы просто сказать «не знаю». В случае недостатка или отсутствия релевантной информации по заданному вопросу генерируется странный ответ, не содержащий полезной информации, а то и вовсе лишённый смысла.

Третий недостаток, часто отмечаемый у всех систем, работающих на нейронных сетях – это то, что не ясно каким образом, руководствуясь чем был дан тот или иной ответ, а тем более совет. А, следовательно, и проверить адекватность такого ответа сложно [Levin, 2016].

Примером известного современного чат-бота может служить разработанный Стивом Уорсвиком чат-бот Mitsuku [Mitsuku]. Для разработки данного бота

использован язык AIML (Artificial Intelligence Markup Language) [AIML]. Его называют одним из лучших чат-ботов, он является 4-х кратным лауреатом премии Лобнера. Однако тестирование Mitsuku позволило определить важный недостаток: бот плохо удерживается в контексте диалога. То есть, проще говоря, забывает, о чём только что шла речь. Это может приводить к неадекватным ответам. Например, бот спрашивает пользователя: «What is your favourite film?» («Какой твой любимый фильм?»). Пользователь отвечает: «Tell me a story», имея в виду название некоторого фильма. Бот же, в свою очередь, воспринимает эту фразу как обращение к нему с просьбой рассказать историю и рассказывает некоторую историю. Кроме того, Mitsuku рассчитан на общение на неспециализированные темы. Если беседа переходит в узкую предметную область, возможны неадекватные, лишённые смысла ответы.

Одним из примеров современных платформ (фреймворков) для создания чат-ботов, использующих в том числе и подходы, базирующиеся на рекуррентных нейронных сетях, является DeepPavlov [DeepPavlov]. Более подробно речь о ней пойдёт ниже.

Альтернативой можно считать парадигму базы знаний, строящуюся на онтологии. Онтологией называется иерархический способ представления набора понятий и их отношений. Такая база может быть обучаемой – онтология может строиться на основе естественно-языковых текстов. Кроме того, следует отметить, что онтологии могут быть перенесены для работы с другим естественным языком. Необходим лишь перевод терминов.

Также существует множество конструкторов чат-ботов, которые позволяют создавать простые чат-боты. Большинство таких ботов реализуют модель архитектуры, основанную на поиске. Это означает, что чат-бот осуществляет выбор ответа из библиотеки предопределенных ответов основываясь на анализе контекста. Контекст может включать в себя текущую позицию в дереве диалога, все предыдущие сообщения в диалоге, ранее сохраненные переменные (например, имя пользователя).

Появилось немало платформ, позволяющих создать чат-бота даже без написания программного кода [Cristina Krestu, 2019]. Можно привести следующие примеры: Aimylogic, Bot Kits, Botmother, Botsify, Chatfuel**.** Мы не

будем останавливаться на подробном описании каждой из перечисленных платформ, они в целом подобны. Но следует отметить, что такие системы обладают рядом существенных недостатков. Во-первых, они не являются свободными приложениями с открытым исходным кодом, что усложняет анализ их работы, модификацию и возможность расширения функционала на усмотрение разработчика. Второй важный недостаток – это ограниченность перечня сервисов обмена сообщениями, с которыми они могут взаимодействовать. Например, только с Face Book-ом или Telegram-ом. И основной, важный для нас недостаток – они в основном не интеллектуальные. То есть используют готовые шаблоны ответов, максимум некоторые могут реализовывать простую аналитику или подключаться к RSS-каналу (например, для регулярного сообщения свежих новостей).

Часто можно встретить упоминания использования при построении продвинутых интеллектуальных чат-ботов сервисов лингвистического анализа. Наиболее известные из них – это IBM Watson Conversation, Dialogflow, LUIS.

IBM Watson – это суперкомпьютер фирмы IBM, оснащённый системой искусственного интеллекта, созданный группой исследователей под руководством Дэвида Феруччи. Его создание – часть проекта DeepQA. Основная его задача – понимать вопросы, сформулированные на естественном языке, и находить на них ответы с помощью ИИ [IBM Watson].

IBM Watson Conversation представляет собой платформу с дружественным интерфейсом, которая позволяет быстро создавать и разворачивать чат-боты и виртуальные агенты для различных каналов, включая платформы обмена сообщениями, мобильные устройства и даже роботов. IBM Watson Conversation позволяет пользователю создать приложение, которое понимает естественный язык и способно давать ответы в ходе диалога. Виртуальный агент Watson может быть настроен для специфической предметной области [Watson Conversation].

Dialogflow — это сервис, позволяющий создавать чат-ботов для разных платформ и языков на разных устройствах. Это платформа для понимания естественного языка, которая позволяет легко проектировать и интегрировать диалоговый пользовательский интерфейс в мобильное

приложение, веб-приложение, устройство, бот, интерактивную систему голосового ответа и т. д. Dialogflow может анализировать различные типы ввода пользователя, включая текстовые или аудиовходы (например, с телефона или записи голоса). Он также может отвечать несколькими способами: с помощью текста или с помощью искусственной речи [Dialogflow].

LUIS – это технология, которая позволяет разработчикам создавать интеллектуальные приложения, понимающие естественный язык и взаимодействующие с пользователями. Например, помогает приложению анализировать высказывание «Забронируй мне билет в Лондон», и распознавать намерение клиента «Забронировать» и направление «Лондон». LUIS активно обучается и совершенствуется в процессе работы приложения под руководством разработчика. LUIS отмечает те случаи, когда он не может однозначно оценить высказывания, и даёт возможность указать, как анализировать подобный случай в следующий раз. LUIS поддерживает английский, французский, итальянский, немецкий и китайский языки. Он предлагает набор REST API, которые используется для автоматизации процесса создания приложений и их публикации. LUIS также работает с сервисом распознавания речи Microsoft Cognitive Service [LUIS].

Но закрытость исходного кода и ориентированность в большей степени на английский язык готовых решений в данной области лингвистического и смыслового анализа существенно ограничивает возможность их применения для наших задач.

Далее рассмотрим примеры сервисов для создания чат-ботов.

Сервис Ana [Ana] позволяет создавать ботов для различных сфер деятельности: электронная коммерция, автомобили, недвижимость. Данный сервис поддерживает интеграцию с Android и iOS SDK **(**software development kit), содержит интерфейс для администратора, где можно производить аналитику и управлять сервисом. Фактически – это фреймворк с открытым исходным кодом, снабжённый конструктором для автоматизации создания и администрирования ботов. Интерфейс Ana предоставляет для работы четыре основных компонента: платформа (platform); студия (studio); симулятор (simulator); SDK.

Основные компоненты представлены в платформе. Это API, который обслуживает запросы, приходящие из чата. Этот компонент отвечает за объединение шаблона диалогового потока с текущим содержимым чата и возврат позиции диалогового потока по запросу клиентов.

Студия служит для моделирования шаблонов разговоров (диалоговых потоков). В ней конструируется смысловые потоки диалогов. В конструкторе смысловых потоков предусмотрена возможность поддержки нескольких языков в одном потоке. Имеется поддержка расширенного текста (задание стилей тексту), смайликов, HTML и медиа-контентов (голосовой чат). Студия позволяет людям, не являющимися разработчиками редактировать смысловой поток и его содержимое в любом участке диалога. Внесённые изменения отражаются мгновенно.

Симулятор используется для тестирования и проверки разрабатываемых диалоговых потоков.

SDK для iOS и Android помогают интегрировать чат-бота с приложениями, работающими на мобильных устройствах.

Созданный при помощи Ana готовый чат-бот разделяется на две основные компоненты: Ana Chat Server и интерфейсный модуль бота. Последний может быть представлен в 3-х вариантах: Web Chat, Android Chat и iOS Chat. Ana Chat Server требуется для публикации смысловых потоков и обслуживания опубликованного контента по разным каналам, включая Интернет (в том числе Facebook и многие другие сервисы обмена сообщениями), Android, iOS. Ana Chat Server состоит из микросервисов, обслуживающие определенные предметные домены. Всего имеется 12 микросервисов, из которых 7 основных и 5 дополнительных. Имеются следующие микросервисы.

Основные: 1. регистратор (service-registry); 2. сервер конфигурации (config-server); 3. точка доступа к управлению программными функциями (api-gateway); 4. ws-customers; 5. сервис, выполняющий рутинные задачи (business-service); 6. регистратор пользователей (user-service); 7. ядро (core).

Опциональные: 1. сервис хранения истории (history service); 2. плагин для работы с облачным сервисом сообщений и уведомлений для Android, iOS и

веб-приложений (fcm-plugin); 3. сервис агентов (agents-service); 4. сервис работы с файлами (file-service); 5. функции аналитики (analytics).

Микросервисы доступны на одном хосте, но на разных портах.

В целом Ana – платформа для чат-ботов основанная на шаблонном подходе. При этом шаблоны не совсем примитивны. А именно, платформа предоставляет возможность отправки запросов к некоторым (в том числе и сторонним) API для получения или отправки данных, возможность встраивания функциональных кнопок, добавление изображений, видео или аудио компонентов и даже возможность перехода к другому боту или другому смысловому потоку. Таким образом, Ana подходит для достаточно простых функциональных чат-ботов, не претендующих на интеллектуальность, но способных ответить на типичные простые вопросы пользователя, предоставить пользователю необходимые ссылки, файлы, вызвать программные процедуры в режиме дружественного интерфейса. К некоторым примечательным особенностям Ana можно отнести архитектуру, основанную на микросервисах.

Также существует платформа Rasa [Rasa], которая работает в связке с инструментом NLU (Natural-language understanding), который подразумевает «понимание естественного языка». Под понятием NLU понимается пост-обработка текста после использования алгоритмов NLP (Natural-language processing – идентификация частей речи и т. д.). Это означает превращение пользовательских сообщений в структурированные данные [Semaan, 2012].

Стек технологий Rasa и NLP позволяют осуществлять обработку текста на естественном языке и на основе этого формировать общее понимание этого текста. Генерация ответа пользователю происходит на основании шаблонов, которые задаются создателем (администратором) чат-бота с применением NLP и NLU алгоритмов. Чем больше таких шаблонов будет задано, тем точнее и полнее будут ответы.

На рисунке 1 приведена архитектура типичного чат-бота, созданного на основе платформы Rasa.

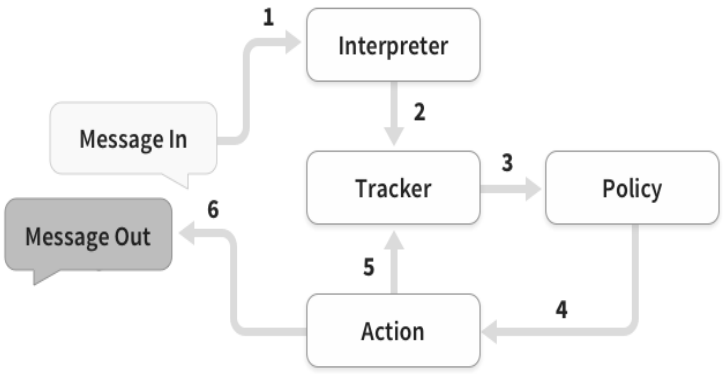

Рисунок 1. Архитектура чат-бота, созданного на основе платформы Rasa

Сообщение принимается и передается объекту Interpreter, который преобразует его в словарь, включающий исходный текст, намерение, а также любые найденные объекты. Эта структура обрабатывается NLU. Схему работы системы можно описать следующим образом:

1. Объект `Tracker` отслеживает состояние диалога. Он получает информацию о том, что пришло новое сообщение.
2. Объект Policy получает текущее состояние объекта `Tracker`.
3. Объект Policy выбирает, какое действие предпринять дальше.
4. Выбранное действие регистрируется объектом `Tracker`.
5. Ответ отправляется пользователю.

Недостаток подобной архитектуры заключается в том, что такая модель медленно обучается и существует необходимость увеличения количества шаблонов для ведения адекватного диалога. Такая модель не является генеративной, а корректность ответа бота зависит от заданных текстовых шаблонов. Она несколько более продвинутая, чем описанная выше Ana, в том отношении, что может осуществлять анализ смысла сообщений пользователя, это даёт большую свободу ведения диалогов, а бот может выступать в некоторой степени как виртуальный собеседник, а не только как интерактивный интерфейс. В то же время чисто шаблонные ответы снижают гибкость подобной

системы, в том смысле, что в ней не может быть получено новой информации (пускай и являющейся логическим следствием из имеющихся данных), а только один из множества шаблонов. В то же время в имеющихся описаниях нет упоминаний об архитектуре, основанной на микросервисах, которая является одним из немногих плюсов Ana, также сильно ограничены возможности относительно управления при помощи чат-бота некоторым дополнительным программным функционалом и поддержка сторонних API.

Платформа DeepPavlov – это библиотека искусственного интеллекта с открытым исходным кодом, построенная на Tensor Flow и Keras [DeepPavlov]. Это платформа, обладающая рядом возможностей, среди которых можно выделить следующие основные: ответы на вопросы по тексту (текст QA); ответы на вопросы, в частности модель ODQA; распознавание именованных сущностей (NER); анализ тональности.

Ответы на вопросы по тексту (текст QA) – это задача поиска ответов на вопросы в заранее заданном фрагменте текста (например, в параграфе из Википедии).

Например, если имеется следующий текст:

«*Урбанизация (от лат. Urbanus — городской) – процесс повышения роли городов, городской культуры и «городских отношений» в развитии общества, увеличение численности городского населения по сравнению с сельским и «трансляция» сформировавшихся в городах высших культурных образцов за пределы городов*».

Вопрос может быть таким:

«*Что делает урбанизация?*»

В свою очередь ответ будет таким:

«*увеличивает численность городского населения по сравнению с сельским*»

Практическим применением данного функционала может быть, например, поиск ответов на вопросы по документации.

Кратко рассмотрим подходы, лежащие в основе поиска ответа на вопрос в естественно-языковом тексте. В DeepPavlov для данной задачи имеются две модели: BERT и R-Net. Обе модели предсказывают начальную и конечную позицию ответа в данном контексте.

BERT (Bidirectional Encoder Representations from Transformers) представляет собой преобразователь, предварительно обученный работе с моделью «маскированного языка» и задачам прогнозирования следующего предложения [Devlin, 2019]. В структуре работы модели BERT выделяются два этапа: предварительная подготовка и точная настройка. Во время предварительного обучения модель обучается на немаркированных данных. Для тонкой настройки модель BERT сначала инициализируется предварительно подготовленными параметрами, и все параметры настраиваются с использованием маркированных данных из последующих задач. Отличительной особенностью модели BERT является её унифицированная архитектура для решения различных задач. Архитектура модели BERT представляет собой многослойный двунаправленный преобразователь, основанный на оригинальном алгоритме, описанном в работе [Vaswani, 2017], программно реализованном в библиотеке tenor2tensor. В BERT не используются традиционные языковые модели слева направо или справа налево для предварительной подготовки. Вместо этого используются две неконтролируемые процедуры, называемые Masked Language Model (далее «Masked LM» или MLM) и «Предсказание следующего предложения» (Next Sentence Prediction или NSP). Для создания двунаправленного представления в модели BERT просто случайным образом маскируют некоторый процент входных токенов (токеном может выступать слово или последовательность слов), а затем прогнозируют эти замаскированные токены. Эта процедура лежит в основе «Masked LM». Обычно случайным образом маскируют 15 % от всех токенов в каждом обрабатываемом участке текста. Предсказывают только замаскированные слова, не стремясь полностью восстановить всю входную информацию. Хотя указанный подход и позволяет получить двунаправленную предварительно обученную модель, его недостатком является то, что создаётся несоответствие между предварительным обучением и последующей тонкой настройкой, поскольку токен [MASK] (который ставился на место

маскируемых токенов) не виден во время тонкой настройки. Чтобы смягчить этот эффект, «замаскированные» слова не всегда заменяются фактическим токеном [MASK]. Генератор обучающих данных выбирает 15% позиций токенов случайным образом для прогнозирования. Если i-й токен (слово) выбран для маскировки, мы заменяем его на токен [MASK], но только в 80 % случаев. В остальных 20 % случаев случаях поступаем одним их двух способов: в 10 % производится замена текущего токена на случайный токен, в оставшихся 10 % токен остаётся неизменным.

Следующей задачей является предсказание следующего предложения. Многие важные задачи, такие как «Ответ на вопрос» (QA) и «Вывод» сообщения на естественном языке (NLI), основаны на понимании взаимосвязи между двумя предложениями, которая непосредственно не отражается в языковой модели. Чтобы обучить модель, которая понимает отношения между предложениями, в рамках модели BERT производится предварительное обучение задаче прогнозирования следующего предложения, которая может быть сгенерирована из любого одноязычного корпуса. Например, в проанализированном корпусе в 50 % случаев B является предложением, следующим за предложением A (помечено как IsNext), а в других 50 % случаев это случайное предложение из корпуса (помечено как NotNext).

Точная настройка (следующий этап) не является сложным процессом, поскольку механизм самодостаточности в преобразователе позволяет BERT моделировать многие последующие задачи путем замены соответствующих входов и выходов. По сравнению с предварительным обучением, точная настройка является относительно малозатратной процедурой. Все результаты тестирования, приведенные в документации [BERT, 2018], могут быть воспроизведены не более чем за 1 час на одном облачном TPU (Tensor Processing Unit) или за несколько часов на графическом процессоре.

Предварительно обученный BERT можно использовать для ответов на вопросы в наборе данных SQuAD (Stanford Question Answering Dataset), просто применив два линейных преобразования к выходным данным. Первое и второе линейное преобразование используются для прогнозирования вероятности того, что текущий подтекст является начальной или конечной позицией ответа.

В DeepPavlov помимо BERT также используется другая модель ответов на вопросы, которая основана на R-Net, предложенной Microsoft Research Asia [R-Net, 2018]. Описание данной модели приведено в работе [Wei, 2017].

R-NET по своей сути – это прямая модель, основанная на нейронных сетях, предназначенная для ответов на вопросы в стиле понимания, целью которой является ответить на вопросы по данному отрывку текста. Данная сеть была протестирована на больших наборах данных, таких как набор данных для ответов на вопросы Стэнфорда (SQuAD) и Microsoft MAchine (MS-MARCO). Схема модели R-NET приведена на рисунке 2.

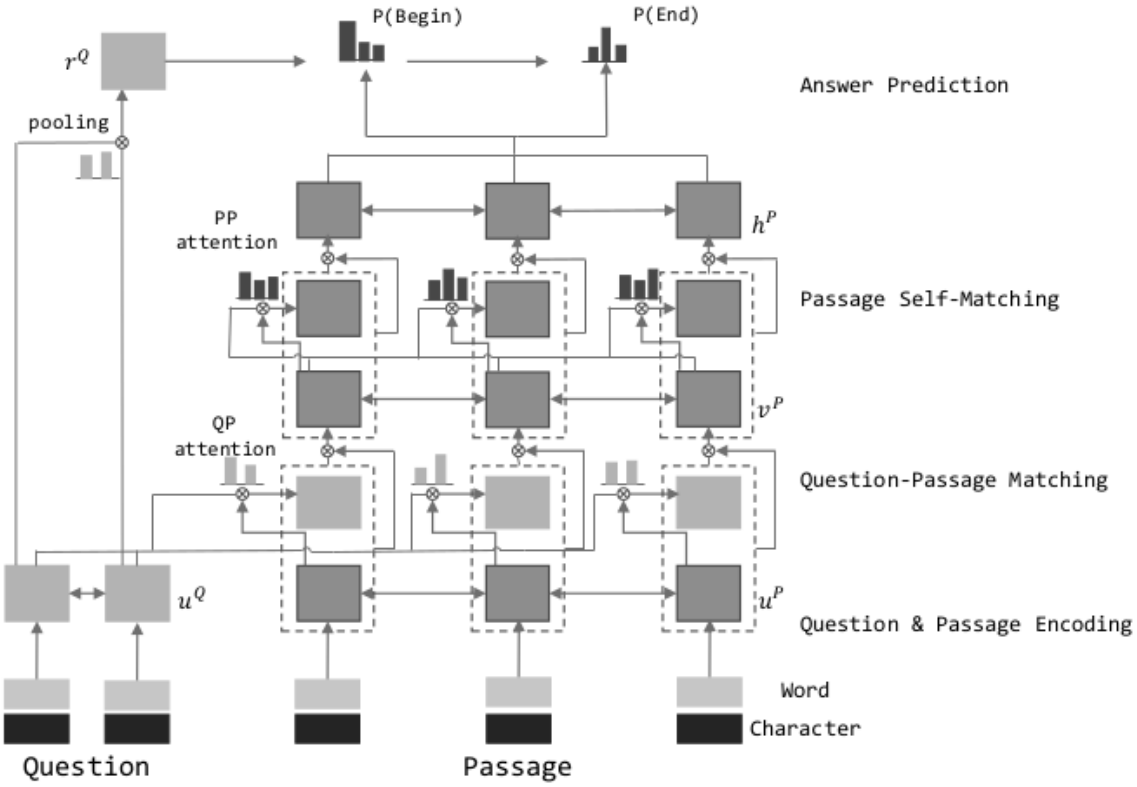

Рисунок 2. Архитектура R-NET

Для формирования ответа на вопрос в стиле понимания прочитанного задаются некий отрывок текста P и вопрос Q. Задача состоит в том, чтобы предсказать ответ A на вопрос Q на основе информации, извлеченной из P. Архитектура R-NET сопоставляет вопрос и отрывок, затем переходит к

рекуррентным сетям, чтобы получить схему текстовой последовательности (в которой ищется ответ) с учетом вопроса. Затем механизм самосогласованной обработки уточняет полученную ранее схему, сопоставляя текстовую последовательность с самой собой, что позволяет кодировать информацию из всей данной последовательности. На последнем этапе сети указателей используются для определения местоположения ответов в заданном текстовом отрывке.

Подробности реализации модели R-NET раскрыты в работе [R-Net].

Многие чат-боты реализуют подход называемый «открытая доменная система ответов на вопросы» (англ. Open-domain question answering или ODQA) [Harabagiu, 2006], [Sun, 2018]. ODQA – это задача поиска ответа на любой вопрос внутри коллекции документов, например, в Википедии. Решение задачи идет в два шага: сначала выбираются релевантные документы; затем в них ищутся ответы. Бизнес-решения на основе ODQA – это, например, диалоговые ассистенты, отвечающие на вопросы по корпоративным базам знаний, справочной и технической документации.

Задача ODQA сочетает в себе реализацию задачи поиска документов (поиск соответствующих статей) с задачей машинного понимания текста (определение диапазона ответов из этих статей). Система ODQA может использоваться во многих приложениях. Чат-боты применяют ODQA для ответа на запросы пользователей, в то время как бизнес-ориентированные решения Natural Language Processing (NLP) используют ODQA для ответа на вопросы, основанные на внутренней корпоративной документации. Ниже на рисунке 3 показан типичный диалог с системой ODQA.

Q: What’s the name of Anakin Skywalker?

A: Darth Vader.

Q: Who destroyed the Death Star?

A: Luke Skywalker.

Рисунок 3. Типичный диалог с системой ODQA [Konovalov, 2019]

Существенной частью любой диалоговой системы, которая необходима для извлечения информации из текста является распознавание именованных сущностей (англ. Named Entity Recognition или NER). NER в составе системы – это компонент для распознавания именованных сущностей. Задача заключается в классификации токенов текста по известным категориям – имена людей, количество, локации, организации, время и дата, цена и валюта, и т. п. Например, с помощью платформы DeepPavlov можно распознать до 19 сущностей [NER].

Ещё одной методикой, получившей распространение при создании чат-ботов является анализ тональности. Это задача, заключающаяся в автоматизированном представлении текстовых и эмоциональных оценок авторов (мнений) по отношению к объектам, речь о которых идёт в тексте. Подобный компонент, если он присутствует в системе, позволяет, например, проводить оценку комментариев [Text QA].

В сущности, DeepPavlov представляет собой достаточно большой фреймворк для работы с естественным языком для построения чат-ботов. В частности, он предоставляет следующие пакеты: набор предварительно обученных NLP моделей, компонентов диалоговой системы и шаблонов; фреймворк для реализации и тестирования своих собственных диалоговых моделей; инструменты для интеграции приложений со смежной инфраструктурой (месседжеры, службы поддержки и т. д.); инструменты для доступа и работы с соответствующим наборам данных. Согласно официальной документации [DeepPavlov Doc] общая структура системы, построенной при помощи DeepPavlov выглядит как показано на рисунке 4.

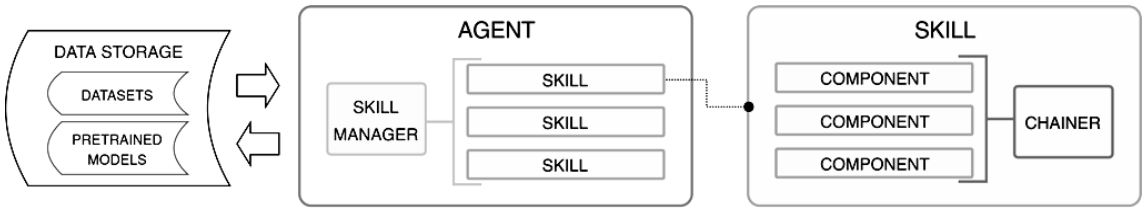

Рисунок 4. Концептуальная схема архитектуры платформы DeepPavlov

Представленная структура имеет в своём составе следующие базовые составляющие:

- агент (AGENT) – диалоговый агент, общающийся с пользователями на естественном языке (в текстовом режиме);

- «навык» (SKILL) – выполняет задачи, необходимые пользователю. Как правило, это предоставление информации или реализация транзакции (например, ответ на вопрос, бронирование билетов и т. д.).

- хранилище данных (DATA SORTAGE) – сервис, хранящий базы данных и предварительно обученные модели и предоставляющий API для работы с ними;

- менеджер «навыков» (SKILL MANAGER) – компонент, выполняющий выбор «навыка», необходимого для корректной для генерации ответа в конкретном случае;

- объединитель (CAINER) – строит связи между агентом и моделями из разнородных компонентов. Это позволяет обучать и работать с несколькими моделями как с единой сущностью;

- компонент (COMPONENT) – функциональная оболочка модели или «навыка»;

- модель (MODEL) – любая NLP модель. В платформе DeepPavlov выделяются 3 основных типа моделей: основанные на правилах (необучаемые), основанные на простом машинном обучении (обучаются вначале один раз), основанные на глубоком обучении (могут обучаться самостоятельно, без учителя в процессе работы).

Наименьшим строительным блоком такой системы является Компонент (COMPONENT). Компонент отвечает за одну определённую функцию. Он может быть реализован в виде нейронной сети, или другой модели машинного обучения или представлять собой систему, основанную на правилах. Компоненты могут быть объединены в Модель (MODEL) или Навык (SKILL). Модель решает несколько большую задачу по сравнению с отдельным Компонентом. Тем не менее, с точки зрения реализации Модели не отличаются от Компонентов. Отличие Навыка от Модели заключается в том, что вход и

выход Навыка должны быть строками. Поэтому Навыки обычно предназначены для задач ведения диалога. Предполагается, что Агент – это многоцелевая диалоговая система, которая включает в себя несколько Навыков и может переключаться между ними. Это может быть диалоговая система, которая содержит Навыки, ориентированные на достижение текущей цели чат-бота, и выбирает, какой из Навыков использовать для генерации ответа, в зависимости от данного пользовательского ввода.

DeepPavlov можно подключать к своему проекту на Python как внешнюю библиотеку, и использовать в приложении необходимый функционал. Также предусмотрена возможность интеграции собственных моделей в систему, построенную на DeepPavlov. Это является её важным преимуществом.

Как недостаток DeepPavlov можно выделить то, что эта платформа ориентирована в первую очередь на использование моделей, основанных на нейронных сетях, следовательно, она имеет и характерные для нейронных сетей проблемы.

У рассмотренных выше платформ для создания чат-ботов алгоритмы формирования ответов пользователям, в основном, основаны на простых или сложных шаблонах, которые, опираясь на содержание контекста, пытаются сформулировать осмысленный ответ. Другим вариантом является применение генеративных алгоритмов, базирующихся на машинном обучении (обычно рекуррентных нейронных сетях). Методы формирования ответа пользователю у различных чат-ботов весьма схожи. Их основной недостатком является то, что они не обладают достаточной гибкостью, ориентированы, в основном, на ограниченное количество предметных областей, это приводит к дополнительным временным затратам на обучение выбранной модели для множества предметных областей. Поэтому нами был рассмотрен подход генеративного способа предоставления ответа на основе онтологии и самонаполняемой базы знаний, которая не привязана к конкретной предметной области.

Существует множество информационных систем, использующих онтологии для реализации отдельных компонентов программы, например, Ontology-Driven Automation of IoT-Based Human-Machine Interfaces Development [Ryabinin, 2019],

[Chuprina, 2019]. Указанная система позволяет проводить визуальную аналитику вариативности речевого поведения пользователей социальных сетей в зависимости от психологических черт личности. В статье [Ryabinin, 2018] рассматривается создание фундаментальной концепции личности, которая бы позволила описывать, объяснять и прогнозировать речевое и неречевое поведение человека и социальных групп, включая группы пользователей социальных интернет-сервисов (англ. Social Network Services, SNS). Несмотря на то, что эти приложения не являются чат-ботами, подходы разработанные при их создании могут быть эффективно использованы при разработке интеллектуальных чат-ботов.

Принцип обучения интеллектуального агента на основе онтологии описан в работе [Smirnova, 2012]. Выделяются следующие этапы:

1. Выявление основных ключевых слов, связанных с предметной областью. Составление глоссария.

2. Системно-онтологический анализ предметной области, в результате которого строится интерпретационная модель предметных знаний. В процессе анализа эти знания разделяются на инвариантные и прагматические знания.

3. Создание необходимых шаблонов на базе онтологии, обучение интеллектуального агента в соответствии с построенной онтологией. На данном этапе в основном используются инвариантные знания.

4. Проработка связей реплик с предысторией разговора с пользователем, установление «якорей», позволяющих агенту вести связный диалог. Такие цепочки в диалоге разрабатываются с учетом связей между различными понятиями в составленной онтологии.

5. Дополнение базы знаний более точными формулировками в соответствии с предметной областью.

6. Дополнительная разработка шаблонов, содержащие все основные определения предметной области.

7. Часто складываются ситуации, когда интеллектуальный агент «не понимает» реплику пользователя, тогда он должен задать пользователю уточняющий вопрос.

8. Проработка реплик агента после продолжительного ожидания (паузы в беседе).

9. Проработка ситуаций дублирования ключевых слов в качестве сигналов для вызова шаблонов. Так, если ключевое слово встречается несколько раз, это может служить веским аргументом для вызова некоторой функции или выбора специфической шаблона для ответа.

10. Анализ разговоров агента с пользователями, выявление нелогичных ответов и корректировка его реакции, т. е. последовательное обучение агента. Данный этап может продолжаться в течение всего времени существования агента.

Преимущество подхода, основанного на онтологии, в сравнении с подходом, использующим для генерации ответов нейронные сети – это в первую очередь возможность интерпретации, понимания и описания результата работы чат-бота при выдаче ответа. Это же облегчает при необходимости ручную корректировку ответов путём внесения изменений в онтологию или в программную систему, работающую с ней. Мы можем сколь угодно расширять и масштабировать онтологию. Кроме того, существует возможность автоматического построения онтологий на основе набора текстов некоторого предметного домена. Такое обучение происходит значительно быстрее, чем обучение нейронной сети, а вероятность ошибочного обучения значительно ниже. Более того, как указывалось выше, при желании, некорректности, возникшие в ходе автоматического построения онтологии можно исправить вручную, чего так просто не сделаешь с неверно обученной нейронной сетью. Кроме того, онтология лишена многих других присущих нейронным сетям проблем, таких как переобучение, «забывание» старых обучающих наборов, локальных минимумов ошибки.

Подытожив, можно провести некоторую классификацию чат-ботов. Классификация для наглядности представлена в виде схем на рисунках 5 – 7.

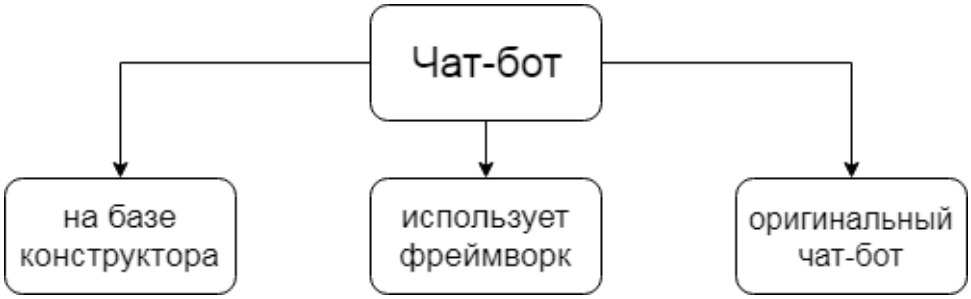

Рисунок 5. Классификация чат-ботов по способу создания

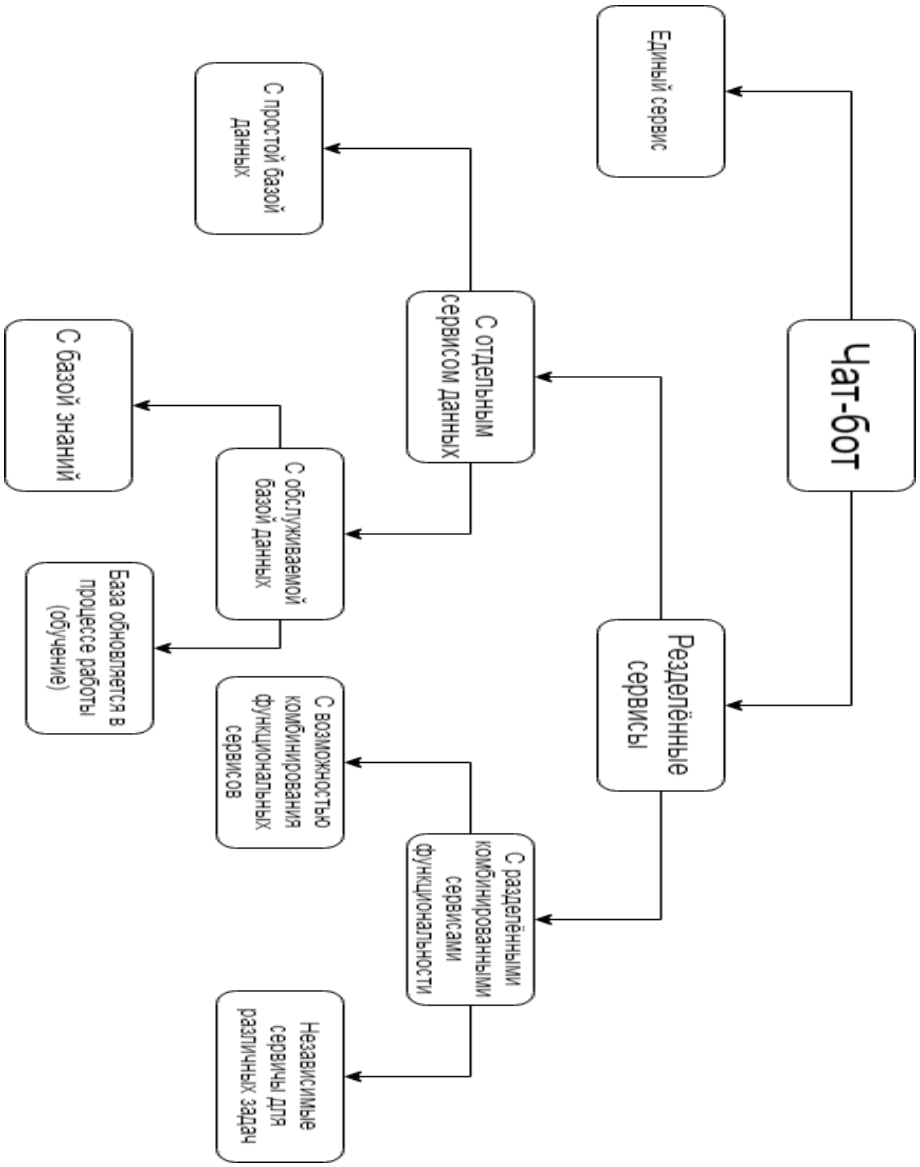

Рисунок 6. Классификация чат ботов по архитектурной схеме

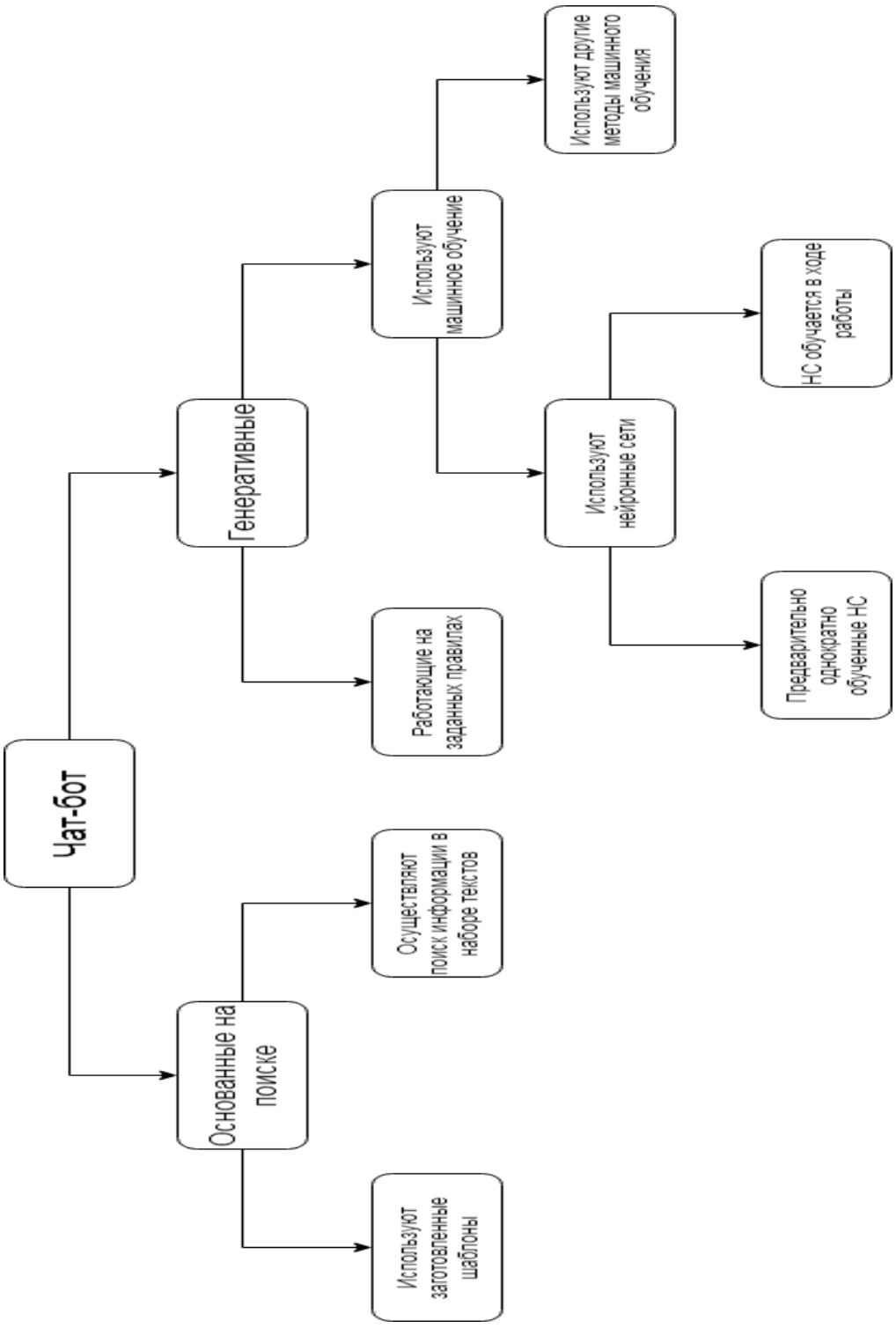

Рисунок 7. Классификация чат ботов по принципу выдачи ответа

В первую очередь чат-бот можно классифицировать по тому, каким образом он был создан. Так он может быть сделан при помощи специального конструктора даже без написания кода. В основном это достаточно примитивные чат-боты

для типичных тривиальных задач. Более продвинутые чат-боты создают с применением соответствующих фреймворков. Но даже этого может быть недостаточно. Функционал любого фреймворка ограничен. Для сложных экспериментальных и нетривиальных задач пишутся полностью оригинальные чат-боты. Это не означает, что код такого бота пишется абсолютно с нуля. Это может быть переработка или дополнение некоторого фреймворка, могут применяться программные библиотеки для решения специфических задач.

Чат-бот в виде единого сервиса в наше время встречается достаточно редко. Это скорее устаревшая концепция, применявшаяся в ранних и примитивных версиях ботов. Большинство современных чат-ботов используют разделённую и даже мультиагентную архитектуру. При этом часто отдельно выделяется сервис работы с данными. В простейших случаях – это просто база данных с СУБД. В более продвинутых вариантах база данных может содержать набор процедур, автоматизирующих её обслуживание. Это может быть сервис пополняющий её в ходе обучения бота, другой вариант – это превращение базы данных в базу знаний, в которой её структура организована так, что обслуживающие процедуры могут выделить из неё информацию, не содержащуюся в базе напрямую, а являющуюся следствием из имеющихся логических посылок. Дальнейшим развитием является более специализированное разделение функциональности в виде микросервисов. При этом возможны варианты, при которых эти сервисы гибко комбинируются между собой или же являются независимыми, но при этом, как правило, не столь узко специализированными.

Важным вариантом классификации чат-ботов является классификация по принципу выдачи ответа. Тут все чат-боты можно глобально разделить на те, которые основаны на поиске и на генеративные. Чат-боты, основанные на поиске, существуют в двух вариантах. В первом случае поиск производится среди имеющихся заранее заготовленных шаблонов, это более примитивный подход. Более продвинутая концепция состоит в поиске информации, являющейся ответом на вопрос в имеющейся базе текстов, которые подвергаются в свою очередь семантическому анализу. Но, к сожалению, такой подход подходит в основном для достаточно примитивных немногословных ответов. Генеративные чат-боты осуществляют синтез ответа на естественном

языке. Такие системы могут работать на заранее заданных правилах. При корректно разработанной большой системе правил можно получить достаточно качественный диалог. Но разработка правил долгая и скрупулёзная задача. Такая система получается очень сложной. Поэтому при разработке чат-ботов распространение получило машинное обучение. В основном применяются нейронные сети. Хотя не исключены и другие варианты машинного обучения (например, построение деревьев принятия решений). В случае использования нейронных сетей также возможны два варианта. В первом случае это может быть статичная предварительно обученная нейронная сеть. Другой вариант – это динамические (обычно рекуррентные) нейронные сети, обучающиеся без учителя в ходе работы чат-бота.

Следует отдельно упомянуть место в данной классификации интеллектуальных чат-ботов, база знаний которых построена на онтологии. Согласно первой классификации (по способу создания) это будет скорее всего оригинальный чат-бот. В принципе, не исключено использование и фреймворков для чат-ботов, но лишь как вспомогательных средств. По архитектурной схеме – это чат-бот на распределённых сервисах, с отдельным сервисом данных, с базой знаний, не обновляемой на основании непосредственно реплик, но обновляемой в ходе сбора данных, в том числе и на основании плохо распознанных реплик пользователей. По способу выдачи ответа такой чат-бот можно отнести к генеративным чат-ботам, выдача ответов которых основана на наборе правил.

Концепция интеллектуального чат-бота

Следует отметить, что большинство обычных чат-ботов не являются интеллектуальными приложениями. Они используют для выдачи ответа лишь заранее заложенные шаблоны, не адаптируются сами под текущие потребности пользователей, редко автоматически выполняют аналитику, не являются обучаемыми, не реализуют автоматическое пополнение базы знаний, или вовсе не содержат базы знаний, а ограничены просто базой данных. Всё что они могут, это определить намерение в сообщении пользователя и вернуть ему наиболее релевантный из имеющихся в базе ответов.

На рисунке 8 схематически представлена типичная архитектура простого не интеллектуального чат-бота. Данная схема является некоторым обобщением, составленным на основе анализа различных чат-ботов.

Чат-бот должен быть построен так, чтобы быть способным взаимодействовать с популярными месседжерами, такими как Telegram, Face Book, Viber. Это повышает внедряемость, а, следовательно, и потенциальную полезность, и востребованность такого приложения. Многие месседжеры предоставляют готовый веб-сервис, который обеспечивает работу с API месседжера. Из этого следует важный вывод, что перед нами не стоит задачи построения пользовательского интерфейса чат-бота и веб-сервиса обеспечивающего работу приложения э этим интерфейсом. Эти задачи высококачественно решены и реализованы популярными сервисами обмена сообщениями ввиду понимания ими важности и актуальности применения чат-ботов. Существует большое количество чат-ботов, работающих по такому принципу.

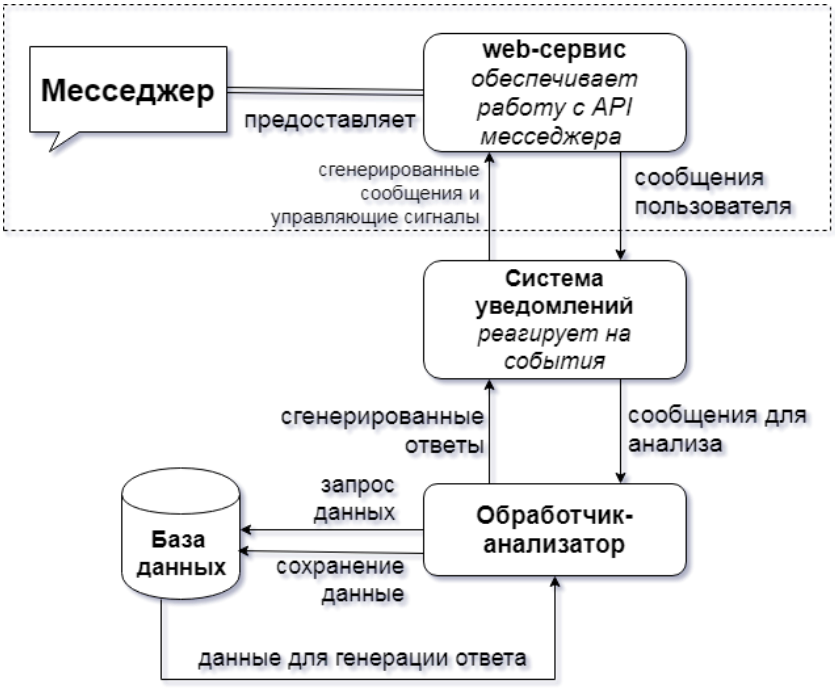

Рисунок 8. Типичная архитектура чат-бота

Предлагаемая архитектура интеллектуального чат-бота приведена на рисунке 9.

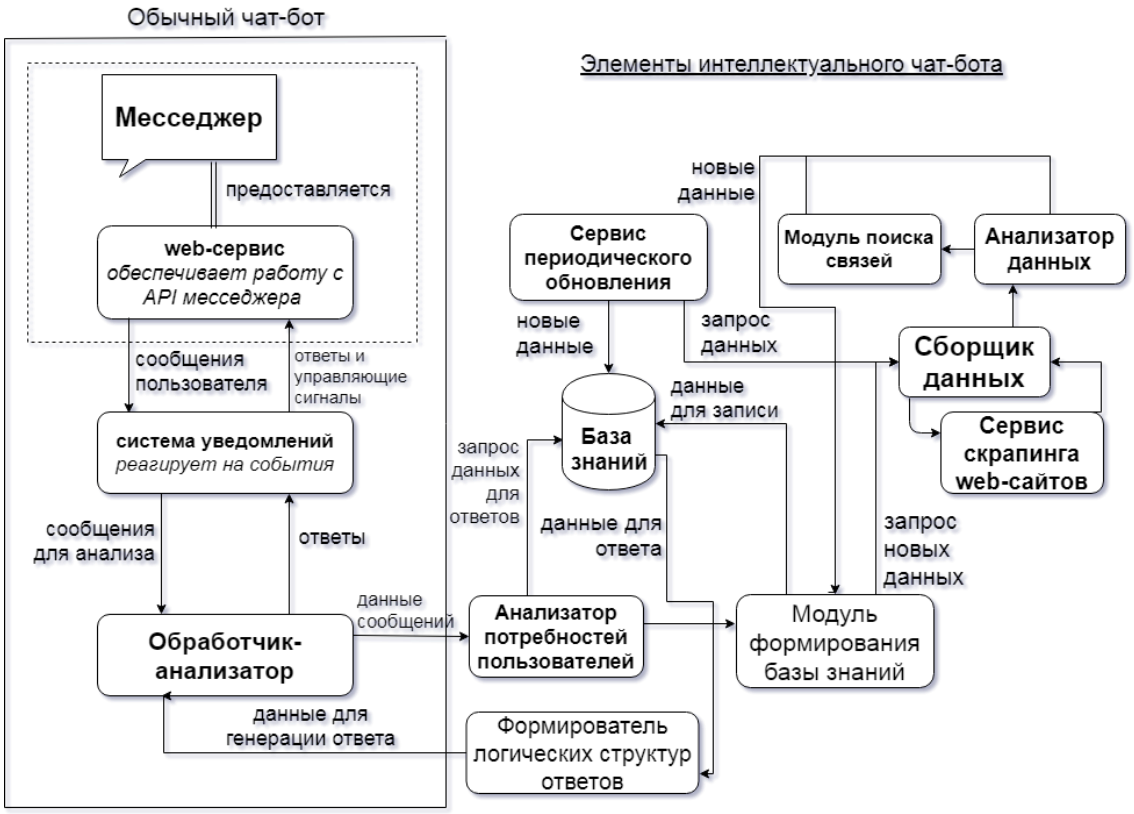

Рисунок 9. Предлагаемая архитектура интеллектуального чат-бота

В своей технической основе (левая часть схемы) предлагаемая архитектура соответствует архитектуре обычного чат-бота. Это сделано для того, чтобы не нарушать стандартность внешней парадигмы его работы и обеспечить возможность встраивания бота в популярные месседжеры и использования готовых работающих web-hook-ов (систем уведомлений).

Главной отличительной чертой является использование не просто базы данных, а базы знаний и построения системы, обеспечивающей её работу. Сама база знаний может содержать в своей основе онтологию.

Так, в данном случае, данные сообщений, подготовленные обработчиком-анализатором, поступают на анализатор потребностей и намерений пользователя. Он, в свою очередь, не только направляет запрос к базе знаний, которая может, как содержать, так и не содержать готовый ответ на поставленный вопрос. Но база знаний тем и отличается от просто базы данных, что она производит интеллектуальный анализ при помощи встроенных в неё хранимых процедур, и пытается получить новые данные, которые не

содержатся в ней непосредственно, но являются следствием из имеющихся в ней данных. Но даже этого может быть недостаточно для получения релевантного ответа. Допустим данные по данной теме или по состоянию на запрашиваемую дату напрочь отсутствуют. Поэтому другой сигнал анализатор потребностей пользователя отправляет модулю формирования базы знаний. Модуль формирования базы знаний управляет сборщиком данных, который использует сервис скраппинга веб-сайтов для поиска новой недостающей информации в сети интернет и во внешних хранилищах данных. Собранные таким образом сырые данные поступают на анализатор, который взаимодействует со специальными дополнительным модулем, предназначенным для поиска и установления связей в полученных сырых данных. Сформированные таким образом новые данные возвращаются модулю формирования базы знаний. После этого он производит обновление базы знаний за счёт пополнения её новой внешней информацией. Так как мир не стоит на месте и новые знания и данные появляются постоянно, в архитектуре предусмотрен сервис периодического обновления базы знаний. Он с установленной периодичностью выполняет по средствам сборщика данных и их анализатора обновление имеющейся базы знаний. Этот процесс можно сделать самонастраивающимся. То есть если какие-то данные обновляются редко, можно увеличить период их обновления, и наоборот, если какая-то информация очень изменчива во времени, система может уменьшить период её обновления. В ответ на поступающий запрос база знаний предаёт данные для ответа модулю формирования логических структур ответа, который формирует данные для генерации ответа в форме, удобной для обработки обработчиком анализатором, в котором происходит синтез ответа на естественном языке. Этот ответ через систему уведомлений постукает на веб-сервис месседжера, а он уже обеспечивает, то, чтобы интерфейс месседжера вывел этот ответ пользователю.

Выводы

При создании интеллектуальных чат-ботов основное внимание исследовательской деятельности следует сосредоточить на модуле генерации ответов.

При реализации веб-сервиса чат-бота следует воспользоваться одной из платформ поддерживающей работу с конкретным месседжером или социальной сетью. Это повысит эффективность использования и практическую ценность бота.

Для работы с флективными языками необходимо разработать лингвистический анализатор для представления входящей информации в виде синтаксических и семантических графов и модуль для синтеза ответов на естественном языке на основании базы знаний.

Для обеспечения интеллектуальности и обновляемости следует снабдить чат-бот сервисами сбора и анализа данных в сети интернет их анализа, и обновления информационной базы. Кроме того, такой чат-бот требует наличия модуля интеллектуального синтеза логических структур ответа.

Использование онтологии в качестве базы знаний для чат-бота является надёжным и гибким способом получения в ходе диалога информации, в том числе и не содержащейся в явном виде, но являющейся следствием имеющихся данных и связей между ними. Чат-бот, база знаний которого основана на онтологии является быстро обучаемым и не даёт нерелевантных ответов в случае недостатка или отсутствия информации по заданному вопросу.

Перечень ссылок

[Weizenbaum, 1976] J. Weizenbaum. Computer Power and Human Reason: From Judgment to Calculation. New York: W.H. Freeman and Company., 1976.

[Melchuk, 1995] И. А. Мельчук. Русский язык в модели смысл-текст. Москва, Вена, 1995.

[SiteActive] Официальный сайт компании «Сайтактив». Электронный ресурс. Режим доступа: http://promo-sa.ru/seo-terms/chat-bot

[Melnichuk, 2018] А. Мельничук. Что такое чат-бот. Электронный ресурс. Режим доступа: https://sendpulse.ua/support/glossary/chatbot

[Glek, 2018] П. Глек. Как создать чат-бота с нуля на Python: подробная инструкция. Электронный ресурс. Режим доступа: https://neurohive.io/ru/tutorial/kak-sozdat-chat-bota-s-nulja-na-python-instrukcija/

[Tarasov, 2015] Д. Тарасов. Chatbot на нейронных сетях. Электронный ресурс. Режим доступа: https://habr.com/ru/company/meanotek/blog/256987/

[Mitsuku] Страница интерфейса чат-бота Mitsuku. Электронный ресурс. Режим доступа: https://www.pandorabots.com/mitsuku/

[AIML] AIML (материал из Википедии). Электронный ресурс. Режим доступа: https://en.wikipedia.org/wiki/AIML

[Levin, 2016] И. Левин. Что может и чего не может нейросеть. Электронный ресурс. Режим доступа: https://habr.com/ru/company/neurodatalab/blog/335238/

[Ana] Официальный сайт платформы Ana для создания чат-ботов. Электронный ресурс. Режим доступа: www.ana.chat

[Rasa] Официальный сайт платформы Rasa для создания чат-ботов. Электронный ресурс. Режим доступа: https://rasa.com

[Semaan, 2012] P. Semaan. Natural Language Generation: An Overview. ournal of Computer Science & Research. Vol. 1, Issue 3. 2012. pp. 50 – 57.

[DeepPavlov] Официальный сайт платформы DeepPavlov для создания чат-ботов. Электронный ресурс. Режим доступа: https://deeppavlov.ai

[R-Net, 2017] R-NET: Machine Reading Comprehension with Self-matching Networks. Электронный ресурс. Режим доступа: https://www.microsoft.com/en-us/research/publication/mcr/

[Devlin, 2019] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019. Электронный ресурс. Режим доступа: https://arxiv.org/pdf/1810.04805.pdf

[Vaswani, 2017] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systemsю 2017. pp. 6000–6010.

[BERT, 2018] BERT in DeepPavlov. Documentation. Электронный ресурс. Режим доступа: http://docs.deeppavlov.ai/en/master/features/models/bert.html

[Wei, 2017] F. Wei, M. Zhou. R-NET: Machine reading comprehension with self-matching networks. Annual Meeting of the Association for Computational Linguistics. 2017. pp. 189 – 198.

[R-Net] R-NET: a neural networks model for reading comprehension. Электронный ресурс. Режим доступа: https://m0nads.wordpress.com/2018/04/17/r-net-a-neural-networks-model-for-reading-comprehension/

[Harabagiu, 2006] S. Harabagiu; A. Hickl Methods for Using Textual Entailment in Open-Domain Question Answering. Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL. 2006. pp. 905 – 912.

[Sun, 2018] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, W. W. Cohen. Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018. pp. 4231 – 4242.

[Konovalov, 2019] V. Konovalov. Open-domain question answering with DeepPavlov. Электронный ресурс. Режим доступа: https://medium.com/deeppavlov/open-domain-question-answering-with-deeppavlov-c665d2ee4d65

[NER] Распознавание именованных сущностей. Электронный ресурс. Режим доступа: https://demo.ipavlov.ai/?#/ru/ner

[Text QA] Анализ тональности. Электронный ресурс. Режим доступа: https://demo.ipavlov.ai/?#/ru/sentiment

[DeepPavlov Doc] Официальная документация к платформе DeepPavlov. Электронный ресурс. Режим доступа: http://docs.deeppavlov.ai/en/master/intro/overview.html

[Ryabinin, 2019] K. Ryabinin, S. Chuprina, K. Belousov Ontology-Driven Automation of IoT-Based Human-Machine Interfaces Development. Computational Science – ICCS 2019. 2019. pp. 110 – 124.

[Ryabinin, 2018] К. В. Рябинин, С. И. Чуприна, К. И. Белоусов, С. С. Пермяков. Методы визуальной аналитики вариативности речевого поведения пользователей социальных сетей в зависимости от психологических черт личности. GraphiCon 2018. 2018. pp. 167 – 171.

[Chuprina, 2019] S. Chuprina, I. Postanogov. New Intelligent Tools to Adapt NL Interface to Corporate Environments. Computational Science – ICCS 2019. 2019. pp. 27 – 40.

[Smirnova, 2012] Л. Э. Смирнова. Интеллектуальные агенты в электронном обучении. Газовая промышленность 1 (1). 2012.

[Cristina Krestu, 2019] 14 сервисов для создания чат-бота без навыков программирования. Электронный ресурс. Режим доступа: https://vc.ru/services/57488-14-servisov-dlya-sozdaniya-chat-bota-bez-navykov-programmirovaniya

[IBM Watson] IBM Watson (Материал из Википедии) Электронный ресурс. Режим доступа: https://ru.wikipedia.org/wiki/IBM_Watson

[Watson Conversation] IBM Watson Conversation (Overview). Электронный ресурс. Режим доступа: https://www.predictiveanalyticstoday.com/ibm-watson-conversation/

[Dialogflow] Dialogflow documentation. Электронный ресурс. Режим доступа: https://cloud.google.com/dialogflow/docs/

[LUIS] Умные чат-боты с LUIS AI. Электронный ресурс. Режим доступа: https://singularika.com/ru/technologies/luis_ai/

Authors' Information

Анна Литвин *– Институт кибернетики им. В. М. Глушкова НАН Украины; аспирантка. Проспект Глушкова 40, Киев, Украина, 03187; e-mail: litvin_any@ukr.net*

Основные направления научных исследований: обработка естественного языка для анализа и синтеза текста на флективных языках, интеллектуальные системы, чат-боты

Виталий Величко *– Институт кибернетики им. В. М. Глушкова НАН Украины; Кандидат технических наук. Старший научный сотрудник. Проспект Глушкова 40, Киев, Украина, 03187; e-mail: aduisukr@gmail.com*

Основные направления научных исследований: компьютерные онтологии и их использование в учебном процессе; информационные системы с обработкой объектов естественного языка

Владислав Каверинский *– Институт проблем материаловедения НАН Украины; Кандидат технических наук. Старший научный сотрудник, ул. Кржижановского 3, Киев, Украина; e-mail: insamhlaithe@gmail.com*

Основные направления научных исследований: теория и математическое моделирование процессов фазовых и структурных превращений, эволюция функции распределения дисперсных систем, модифицирование литого метала, обработка естественного языка для анализа и синтеза текста на флективных языках.

DEVELOPMENT OF AN INTELLECTUAL CHAT-BOT ARCHITECTURE

Anna Litvin, Vitalii Velychko, Vladislav Kaverinsky

Abstract*: The article discusses the general concept of chat-bots applications, the development of a chat-bots idea and their practical significance. The analysis is carried out of the currently existing modern chat-bots and platforms for their construction. A special attention was given to the chat-bot platforms Ana, Rasa and PeepPavlov. Conclusions were made on their functionality and usability. Ana is suitable for rather primitive chat-bots able to answer typical questions, provide the user with the necessary links, and invoke program procedures. RASA can analyze the meaning of user messages at the same time, purely template answers reduce the flexibility of such a system. PeepPavlov is a platform with a number of features: answers to questions on the text; answers to questions, in particular the ODQA model; Named Entity Recognition; tonality analysis. A brief review of BERT and R-NET used in PeepPavlov technologies is presented. A review on the most popular and powerful linguistic analysis services is given, namely on IBM Watson Conversation, Dialogflow, and LUIS. The article also describes the principle of training an intellectual agent based on ontology. Basic concepts of this process are listed. The chat-bots were classified by the method of creation, by their architecture and by the principle of a response making. The classification defines a place for intelligent chat-bots based on ontology. It could be a chat-bot on distributed services, with a separate data service, with a knowledge base that is not updated on the basis of replicas directly, by the method of issuing an answer, such chat-bot can be attributed to generative ones which is based on a set of rules. The concept of an intelligent chat-bot is proposed, which combines the ability to integrate with popular messengers and update the knowledge base in accordance with the needs of the user. It is declared that to ensure intelligence and updatability, the chat-bot should be provided with services for collecting and analyzing data on the Internet for their analysis, and updating the information base. Additionally, such chat-bot requires intelligent module for the synthesis of the response logical structure. Using the ontology as a knowledge base for a chat-bot is a reliable and flexible way to obtain information during the dialogue, including information that is not explicitly contained there. A chat-bot which knowledge base is based on an ontology could be quickly trained and does not give irrelevant answers in case of lack or absence of information on a given question.*

Keywords*: chat-bot, analysis of natural language text, generation of meaningful text, knowledge base*.