

МИНОБРАЗОВАНИЯ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО» (СГУ)

Кафедра математической  
кибернетики и компьютерных наук

ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА С ПОМОЩЬЮ  
ПЛАТФОРМЫ DIALOGFLOW  
БАКАЛАВРСКАЯ РАБОТА

студента 3 курса 311 группы  
направления 02.03.02 — Фундаментальная информатика и информационные  
технологии  
факультета КНиИТ  
Вязкова Андрея Андреевича

Научный руководитель  
доцент, к. т. н.

\_\_\_\_\_

В. М. Соловьев

Заведующий кафедрой  
к. ф.-м. н.

\_\_\_\_\_

С. В. Миронов

Саратов 2019

## СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ .....	3
ВВЕДЕНИЕ .....	4
1 Основы компьютерной лингвистики .....	5
1.1 Основные понятия .....	5
1.2 Области применения .....	6
2 Чат-боты и диалоговые агенты .....	8
2.1 Чат-боты .....	8
2.1.1 Чат боты на основе правил .....	8
2.2 Чат-боты на корпусах текстов .....	10
2.2.1 Чат-боты, основанные на информационном поиске .....	10
2.2.2 Seq2Seq чат-боты .....	11
2.3 Диалоговые агенты .....	12
2.3.1 Структура управления диалогового агента .....	14
2.3.2 Понимание естественного языка для заполнения слотов ..	15
3 Платформа для создания диалоговых интерфейсов Dialogflow .....	17
ЗАКЛЮЧЕНИЕ .....	24
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	25
Приложение А Нумеруемые объекты в приложении .....	26

## ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

КЛ — компьютерная лингвистика;

ЕЯ — естественный язык;

NLP(Natural Language Processing) — Обработка естественного языка

## ВВЕДЕНИЕ

В настоящее время область распознавания человеческой речи является очень популярной. С активным внедрением машинного обучения компьютеры стали лучше понимать естественный язык. Голосовые ассистенты, чат-боты являются очень популярными механизмами взаимодействия бизнеса с клиентами. Сегодня можно забронировать билет на самолет или записаться на стрижку, всего лишь сказав «Окей, Google», или написав чат-боту в любимом мессенджере. В данной курсовой работе будет рассмотрена история развития компьютерной лингвистики, основные виды систем распознавания естественного языка, механизмы взаимодействия и основные принципы построения подобных систем.

# 1 Основы компьютерной лингвистики

## 1.1 Основные понятия

Компьютерная лингвистика (КЛ) — междисциплинарная область, которая возникла на стыке таких наук, как лингвистика, математика, информатика (Computer Science), искусственный интеллект (Artificial Intelligence). В своем развитии она до сих пор вбирает и применяет (при необходимости адаптируя) разработанные в этих науках методы и инструменты

Поскольку в КЛ объектом обработки выступают тексты естественного языка, ее развитие невозможно без базовых знаний в области общей лингвистики (языкознания). Лингвистика изучает общие законы естественного языка — его структуру и функционирование, и включает такие области:

- фонология — изучает звуки речи и правила их соединения при формировании речи;
- морфология — занимается внутренней структурой и внешней формой слов речи, включая части речи и их категории;
- синтаксис — изучает структуру предложений, правила сочетаемости и порядка следования слов в предложении, а также общие его свойства как единицы языка.
- семантика и прагматика — тесно связанные области: семантика занимается смыслом слов, предложений и других единиц речи, а прагматика — особенностями выражения этого смысла в связи с конкретными целями общения;
- лексикография — описывает лексикон конкретного ЕЯ — его отдельные слова, их грамматические и семантические свойства, а также методы создания словарей

Наиболее тесно компьютерная лингвистика связана с областью искусственного интеллекта (ИИ), в рамках которой разрабатываются программные модели отдельных интеллектуальных функций.

Общим для указанных наук является компьютерное моделирование как основной способ и итоговая цель исследований, эвристический характер многих применяемых методов. Несколько упрощенно задача компьютерной лингвистики может быть сформулирована как разработка методов и средств построения лингвистических процессоров для различных прикладных задач по автоматической обработке текстов на ЕЯ. Разработка лингвистического про-

цессора для некоторой прикладной задачи предполагает формальное описание лингвистических свойств обрабатываемого текста (хотя бы самое простое), которое может рассматриваться как модель текста (или модель языка).

## 1.2 Области применения

Сфер применения компьютерной лингвистики достаточно много, однако следует описать некоторые из них [1].

Машинный перевод (Machine Translation) — самое раннее приложение КЛ, вместе с которым возникла и развивалась сама эта область. Первые программы перевода были построены в середине прошлого века и были основаны на простейшей стратегии пословного перевода. Однако довольно быстро было осознано, что машинный перевод требует гораздо более полной лингвистической модели. В настоящее время существует целый спектр компьютерных систем машинного перевода (разного качества), от больших интернациональных исследовательских проектов до коммерческих автоматических переводчиков.

Информационный поиск (Information Retrieval) и связанные с ним задачи индексирования, реферирования, классификации и рубрицирования документов.

Полнотекстовый поиск документов в больших базах текстовых документов предполагает индексирование текстов, требующее их простейшей лингвистической предобработки, и создание специальных индексных структур. Известны несколько моделей информационного поиска, наиболее известной и применяемой является векторная модель, при которой информационный запрос представляется в виде набора слов, а подходящие (релевантные) документы определяются на основе схожести запроса и вектора слов документа. Современные интернет-поисковики реализуют эту модель, выполняя индексирование текстов по употребляемым в них словам и используя для выдачи релевантных документов весьма изощренные процедуры ранжирования. Актуальное направление исследований в области информационного поиска — многоязыковой поиск по документам.

Относительно новая задача, связанная с информационным поиском — формирование ответов на вопросы (Question Answering). Пример возможного вопроса: «Кто придумал вилку?». Задача решается путем определения типа вопроса, поиском текстов, потенциально содержащих ответ на этот вопрос

(при этом обычно применяются поисковые машины), и затем извлечением ответа из выданных текстов.

Ещё одна прикладная задача, которая возникла более 50 лет назад и развитие которой стимулировало появление сети Интернет, — это поддержка диалога на ЕЯ. Ранее эта задача чаще всего решалась в рамках какой-либо информационной системы, в частности, для обработки запросов на ЕЯ к специализированной базе данных — в этом случае язык запросов достаточно ограничен, что позволяет использовать упрощенные методы анализа вопросов, а ответы строить по шаблонам. В настоящий момент все более широкое распространение в Интернете получают чат-боты, поддерживающие беседу с человеком на некоторую тему и являющиеся наследниками известной системы ELIZA (разработанной в области ИИ в 70 гг.).

Активно развивающимся направлением является распознавание и синтез звучащей речи. Неизбежно возникающие ошибки распознавания исправляются автоматическими методами на основе словарей и морфологических моделей, также применяется машинное обучение.

## 2 Чат-боты и диалоговые агенты

В данной главе представлены основные концепции диалоговых систем или диалоговых агентов. Эти программы общаются с человеком с посредством текста или голоса и делятся на 2 типа

- Диалоговые агенты, ориентированные на специализированную задачу — предназначены для конкретной задачи и настроены на короткие диалоги с целью получения данных от пользователя для последующей задачи. К ним относятся виртуальные ассистенты, такие как Google Now (и его продолжение Google Assistant), Amazon Alexa, Siri и.т.д. Их диалоговые агенты способны указать направление движения, управлять средствами умного дома, находить рестораны поблизости, совершать телефонные звонки и отправлять текстовые сообщения. Некоторые компании размещают диалоговых агентов на своих сайтах, чтобы отвечать на вопросы пользователей.
- Чат-боты — это системы, предназначенные для имитации неструктурированных разговоров или «чатов», характерных для взаимодействия людей друг с другом. Чат-боты также пытаются пройти различные формы теста Тьюринга. Стоит заметить, что в СМИ термин «чат-бот» используется для обозначения диалогового агента, что является некорректным.

### 2.1 Чат-боты

Как было сказано ранее — чат боты — это системы, имитирующие человеческий разговор. Как и в почти любой области NLP разделяются на 2 класса:

- Чат-боты на основе правил
- Чат-боты на основе корпуса текстов

#### 2.1.1 Чат боты на основе правил

ELIZA [2] — одна из самых первых диалоговых чат-бот систем. Она была создана в 1966 году Джозефом Вейценбаумом и работает по принципу активного слушателя, основной прием которого — это перифраз сообщения, сказанного пользователем.

ELIZA работает по правилам преобразования шаблонов, например:



```

(0 YOU 0 ME) [pattern]
->
(WHAT MAKES YOU THINK I 3 YOU) [transform]

```

Рисунок 1 – Пример правила

В системе ELIZA паттерн «0» означает любую последовательность символов (аналог \* в регулярных выражениях) Символ «3» ссылается на второй «0». Это преобразует фразу «You hate me» в «What makes you think i hates you» Архитектура системы выглядит следующим образом:

```

function ELIZA GENERATOR(user sentence) returns response

Find the word w in sentence that has the highest keyword rank
if w exists
    Choose the highest ranked rule r for w that matches sentence
    response ← Apply the transform in r to sentence
    if w = 'my'
        future ← Apply a transformation from the 'memory' rule list to sentence
        Push future onto memory stack
    else (no keyword applies)
        either
            response ← Apply the transform for the NONE keyword to sentence
        or
            response ← Pop the top response from the memory stack
return(response)

```

Рисунок 2 – Упрощенное изображение архитектуры ELIZA

Однако Вейценбаум замечает, что некоторые общие слова (например слово I в английском языке) приводит к более общим ответам. Вместо него слово «Everybody» приводит к гораздо более интересным ответам, поскольку его использование указывает, что человек ссылается на конкретное событие или человека. Поэтому ELIZA предпочитает отвечать шаблонами содержащими подобные слова.

Если никаких ключевых слов в предложении не было найдено ELIZA выбирает нейтральный ответ. Также ELIZA запоминает последнее предложение в диалоге. Например, всякий раз когда слово «ту» является ключевым словом с самым высоким рангом ELIZA случайно выбирает преобразование из памяти, применяет к предложению и кладет в стек.

Люди стали серьезно воспринимать программу. Вейзенбаум рассказывал историю как его сотрудник попросил его выйти во время разговора с

```
(MEMORY MY
(0 MY 0 = LETS DISCUSS FURTHER WHY YOUR 3)
(0 MY 0 = EARLIER YOU SAID YOUR 3)
(0 MY 0 = DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR 3
```

Рисунок 3 – Пример правил, содержащихся в памяти в ELIZA

Элизой. Когда Вейзенбаум хотел сохранить диалоги для дальнейшего анализа люди стали беспокоиться о вопросах конфиденциальности, что предполагало, что они имели весьма приватные разговоры с программой.

ELIZA используется и в наше время. Некоторые чат-бот системы основаны на модифицированных версиях ELIZA.

## 2.2 Чат-боты на корпусах текстов

Чат-боты, основанные на корпусах текстов, вместо использования правил, используют данные из диалогов между людьми или из диалогов человека и машины. Подобные данные можно собирать из диалогов на чат-платформах, Твиттера или диалогов в фильмах. Данные источники могут обеспечить большие объемы данных и они похожи на естественные разговоры.

Существует два типа чат-ботов:

- основанные на информационном поиске (Information Retrieval)
- основанные на машинном обучении (Sequence to Sequence)

Также как и чат-боты на основе правил, чат-боты на основе корпусов слабо моделируют контекст диалога. Вместо этого они генерируют ответ, основанный на последнем сказанном пользователем предложении. Таким образом они имеют много общего с вопросно-ответными системами, которые фокусируются на отдельных ответах, игнорируя контекст.

### 2.2.1 Чат-боты, основанные на информационном поиске

Принцип, лежащий в основе данных чат-ботов, заключается в том, чтобы реагировать на некоторый пользовательский ход  $X$ , повторением соответствующего хода  $Y$  из корпуса текста. Различия между подобными системами заключаются в том, как они выбирают корпус текста, и как они решают какое пользовательский ход выбрать.

Распространенным выбором корпуса, является сбор данных из человеческих разговоров. Это могут быть платформы для ведения микроблогов

(Twitter, Weibo). Другой подход заключается в сборе данных из диалога человека и чат-бота.

Используя корпус текста и предложение пользователя, система может выбрать алгоритм извлечения информации из корпуса. Два самых простых метода:

1. Вернуть ответ на наиболее похожий ход: учитывая запрос пользователя  $q$  и корпус  $C$ , найти ход  $t$  из  $C$ , который наиболее похож на  $q$ , и вернуть следующий ход, т. е. ответ человека на  $t$  из  $C$ .

$$r = response(\operatorname{argmax}_{t \in C} \frac{q^T t}{||q||t||})$$

Идея состоит в том, чтобы ход, который больше всего напоминает ход пользователя и возвращать ответ человека на этот ход.

2. Вернуть наиболее похожий ход: учитывая запрос  $q$  и корпус  $C$ , вернуть ход  $t$  из  $C$ , который наиболее похож на  $q$ .

$$r = \operatorname{argmax}_{t \in C} \frac{q^T t}{||q||t||}$$

Идея состоит в том, чтобы сопоставить пользовательский ход с ходами из корпуса.

Хоть первый вариант и кажется наиболее интуитивным, второй вариант на практике показывает себя лучше.

Данный подход может быть расширен за счет дополнительного функционала, например: запоминание слов из предыдущих ходов или получение информации о пользователе.

### 2.2.2 Seq2Seq чат-боты

Альтернативный вариант использования корпуса для генерации диалога — это переходить от хода пользователя, к ходу системы. Обычно это своеобразная «версия» ELIZA, основанная на машинном обучении.

Эта идея была впервые разработана с использованием машинного перевода на основе фраз, чтобы перевести обращение пользователя к ответу системы. Однако быстро стало ясно, что задача генерации ответов слишком отличается от машинного перевода. В машинном переводе слова или фразы в исходных и целевых предложениях имеют тенденцию хорошо совмещать-

ся друг с другом, но в разговоре пользовательское высказывание не может делиться словами или фразами с последовательным ответом.

Вместо этой модели были разработаны seq2seq модели

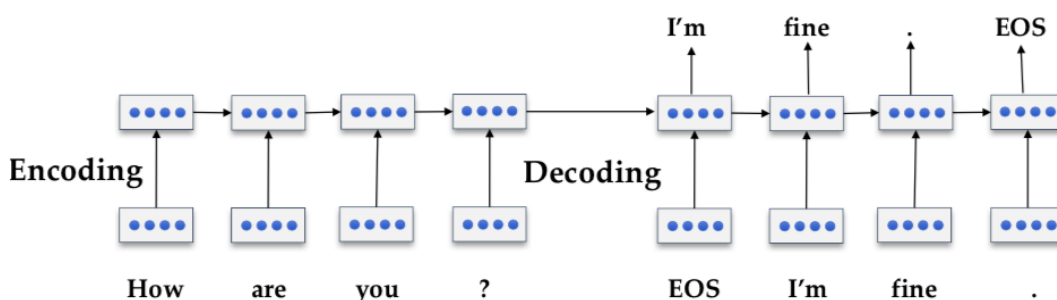


Рисунок 4 – Схема seq2seq модели

Требуется ряд модификаций базовой модели seq2seq для ее адаптации к задаче генерации ответов. Например, базовые модели seq2seq имеют тенденцию создавать предсказуемые, но повторяющиеся и, следовательно, скучные ответы, такие как «Я в порядке» или «Я не знаю», которые закрывают диалог. Эту проблему можно решить, изменив функцию для обучения модели seq2seq.

Другая проблема архитектуры генерации ответов заключается в ее неспособности моделировать более длинный предшествующий контекст диалога. Это можно сделать, позволив модели увидеть предыдущие ходы, например, используя иерархическую модель, которая накапливает информацию за несколько предыдущих ходов. Наконец, генераторы ответов фокусируются на генерации единичных ответов, и поэтому не имеют тенденций к тому, чтобы непрерывно генерировать ответы, которые связаны между собой в несколько ходов. Эту проблему можно решить с помощью обучения с подкреплением, а также таких методов, как состязательные сети, чтобы научиться выбирать ответы, которые делают общий разговор более естественным

### 2.3 Диалоговые агенты

Современные диалоговые системы основаны на предметной онтологии, структурированных знаниях, представляющая типы намерений, которые система может извлечь из пользовательских предложений.

Онтология определяет один или несколько кадров (фреймов), каждый из которых представляет собой набор слотов, а слот определяет значения,

которые может принимать каждый слот. Эта архитектура была впервые введена в 1977 году в системе для планирования поездок GUS, лежащей в основе большинства современных коммерческих цифровых помощников. На рисунке 5 показан пример диалога с системой.

GUS: Hello. My name is GUS. I can help you plan a simple trip by air. Where do you want to go ?  
 Client: I want to go to San Diego on May 28  
 GUS: What time do you want to leave?  
 Client: I must be in San Diego before 10 am  
 GUS: Would you like P.S.A. flight 102 that arrives at 9.15 am ?  
 Client: What is the next flight ?  
 GUS: Air California flight 310 that leaves at 8.30am and arrives at 10.00am  
 Client: I'll take the first one  
 GUS: In what name should I make a reservation, please?  
 Client: Dan Bobrow  
 GUS: I have confirmed the following flight:  
 Client: P.S.A. flight 102 on Wednesday May 28 from San Jose to San Diego that leaves at 7.30 am and arrives at 9.15 am.  
 GUS: What date do you want to return on ?  
 Client: On Friday in the evening.  
 GUS: Would you like the flight that leaves at 7.45 pm ?  
 Client: That's fine.  
 GUS: I have confirmed the following flight: P.S.A. flight 307 on Friday May 30 from San Diego to San Jose that leaves at 7.45 pm and arrives at 9.30 pm Thank you for calling. Goodbye

Рисунок 5 – Пример диалога с GUS

Набор слотов в GUS определяет, что система должна знать, а заполнитель каждого слота ограничен значениями определенного семантического типа. Например, в туристическом домене слот может иметь такие значения как город (следовательно, принимать значения, такие как Сан-Франциско или Гонконг) или дата, название авиакомпании или время:

Slot	Type
ORIGIN CITY	city
DESTINATION CITY	city
DEPARTURE TIME	time
DEPARTURE DATE	date
ARRIVAL TIME	time
ARRIVAL DATE	date

Рисунок 6 – Пример слотов и их семантических типов

Типы в GUS (как и в любом современном диалоговом агенте) могут

иметь иерархическую структуру. Например дата может состоять из месяца и дня.

### 2.3.1 Структура управления диалогового агента

Архитектура управления диалоговыми системами на основе кадров разработана вокруг фрейма. Цель состоит в том, чтобы заполнить слоты в фрейме теми заполнителями, которые хочет пользователь, а затем выполнить соответствующее действие (ответить на вопрос или забронировать рейс). Большинство основанных на фреймах диалоговых систем основаны на конечных автоматах, которые специально разработаны для этой задачи проектировщиком диалогов.

Пример простой архитектуры представлен на рисунке 7

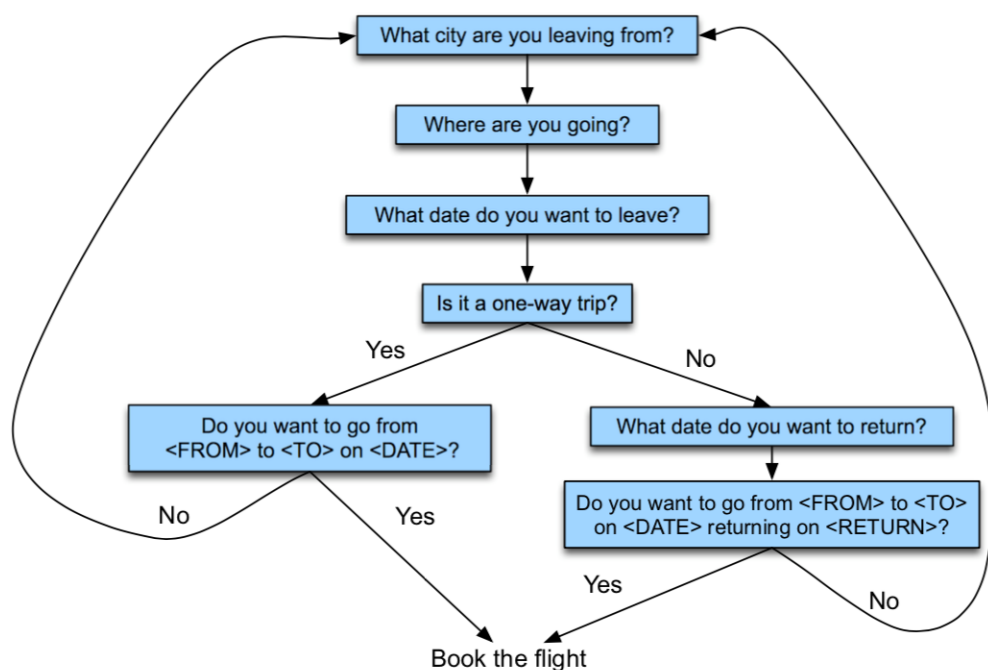


Рисунок 7 – Пример архитектуры конечного автомата для диалога

Состояния конечного автомата соответствуют вопросам к пользователю для заполнения слотов, а дуги соответствуют действиям, которые необходимо предпринять в зависимости от того, что отвечает пользователь. Эта система полностью контролирует разговор с пользователем. Система задает пользователю ряд вопросов, игнорируя (или неверно истолковывая) все, что не является прямым ответом на вопрос, а затем переходит к следующему вопросу.

Такая упрощенная архитектура с конечным состоянием обычно применяется только для простых задач, таких как бронирование авиабилетов. Для большинства приложений пользователям требуется немного больше гибкости. Например, в ситуации планирования командировки пользователь может произнести предложение, которое заполняет несколько слотов одновременно.

Поэтому архитектура GUS в той или иной форме используется в современных системах. Эти системы задают вопросы пользователю, заполняет любой слот указанный пользователем, даже если пользователь указал несколько слотов. Система пропускает вопросы связанные с этими слотами.

Архитектура имеет правила предназначенные для автоматического заполнения слотов при некоторых словах или словосочетаниях. Например для слота «Пункт назначения» для фрейма бронирования самолета система запоминает значение данного слота и заполняет этим значением слот «Город остановки» в фрейме бронирования отеля.

Когда система имеет достаточно данных она выполняет определенное действие, например: бронирование билетов или возврат сообщения пользователю.

### 2.3.2 Понимание естественного языка для заполнения слотов

Цель понимания естественного языка в — извлечь из высказывания пользователя три вещи:

1. Домен. Бронирует ли пользователь билет на самолет, устанавливает будильник?
2. Определение намерений пользователя. Какую задачу пытается выполнить пользователь?
3. Заполнение слотов. Извлечь параметры и заполнить соответствующие слоты.

Задача заполнения слотов и более простые задачи классификации домена и намерений являются частными случаями задачи семантического синтаксического анализа. Метод, используемый в системе GUS и все еще довольно распространенный в промышленных приложениях, заключается в использовании рукописных правил, часто являющихся частью правил условного действия, прикрепленных к слотам.

Системы, основанные на правилах, могут быть реализованы даже с грамматиками. Некоторые системы состоят из больших семантических грам-

матик, разработанных вручную, с тысячами правил. Семантическая грамматика — это контекстно свободная грамматика, в которой левая часть правила является выражаемым семантическим объектом (название слота).

SHOW	→ show me   i want   can i see ...
DEPART_TIME_RANGE	→ (after around before) HOUR   morning   afternoon   evening
HOUR	→ one two three four... twelve (AMPM)
FLIGHTS	→ (a) flight   flights
AMPM	→ am   pm
ORIGIN	→ from CITY
DESTINATION	→ to CITY
CITY	→ Boston   San Francisco   Denver   Washington

Рисунок 8 – Пример правил семантической грамматики

Одним из важных вопросов является отрицание. Если пользователь указывает, что он «не может летать во вторник утром» или хочет встретиться, простая система часто неправильно извлекает «утро вторника» как цель пользователя, а не как отрицание.

Ошибки распознавания речи также должны быть устранены. Одним из распространенных приемов является использование ранжированного списка предполагаемых транскрипций, а не только одну кандидатуру транскрипции. Регулярные выражения или синтаксические анализаторы могут быть просто запущены для каждого предложения в списке, и могут использоваться любые шаблоны, извлеченные из любой гипотезы.

Метод извлечения информации, основанный на правилах подход очень распространен в промышленных приложениях. Он обладает высокой точностью, и, если домен достаточно узок и специалисты доступны, он также может обеспечить достаточный охват. С другой стороны, подобные методы могут быть дорогостоящими и медленными в создании, а рукописные правила могут страдать от проблем с отзывом.

Распространенной альтернативой является использование машинного обучения. Предполагая, что имеется обучающий набор, который связывает каждое предложение с правильной семантикой, мы можем обучить классификатор для сопоставления предложений, намерений и областей, а также модель последовательности для сопоставления предложений и заполнителей слотов.



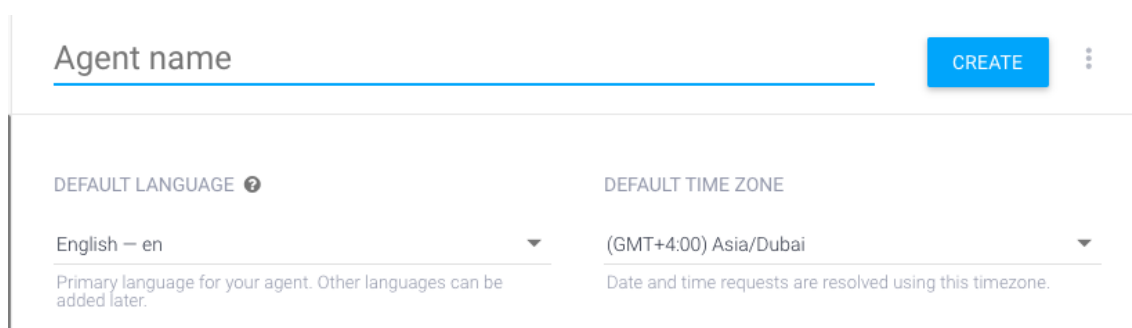
### 3 Платформа для создания диалговых интерфейсов Dialogflow

Dialogflow — платформа для создания диалговых агентов, купленная компанией Google в 2016 году. Диалговые интерфейсы, созданные с помощью Dialogflow работают на широком спектре устройств, включая: телефоны, носимые устройства, умные-колонки, машины и т. д.

Платформа поддерживает более 14 языков, в которые входит Русский.

В данной курсовой работе показан пример создания диалгового агента, предназначенного для получения информации о расписании занятий в университете.

Первым шагом является создание агента. Необходимо ввести название агента, используемый язык и часовой пояс.



The screenshot displays the 'Create Agent' form in Dialogflow. At the top, there is a text input field labeled 'Agent name' followed by a blue 'CREATE' button and a vertical ellipsis menu icon. Below this, the form is divided into two columns. The left column is titled 'DEFAULT LANGUAGE' and shows 'English — en' with a dropdown arrow. Below it, a note states: 'Primary language for your agent. Other languages can be added later.' The right column is titled 'DEFAULT TIME ZONE' and shows '(GMT+4:00) Asia/Dubai' with a dropdown arrow. Below it, a note states: 'Date and time requests are resolved using this timezone.'

Рисунок 9 – Стартовое окно создания агента

Далее откроется окно Intents. Intent — обработчик запроса от пользователя. В нем уже представлены два интента: стандартное приветствие, которое реагирует на приветствие пользователем и интент, определенный на случай невозможности распознавания пользовательского запроса.

Следующий шаг — это создание интента. Необходимо заполнить следующую информацию:

- название интента
- Контекст — сохраняет текущее состояние пользовательского запроса и передает данные от одного интента к другому. Различают два типа контекстов: входящий и исходящий. Входящий контекст позволяет интенту запускаться только тогда, когда этот контекст существует. Исходящий контекст сохраняет в себе данные, которые могут быть переданы другому интенту.
- События — позволяют запустить интент с помощью невербального сигнала, например нажатия кнопки.

- Тренировочные фразы — приблизительные фразы, с помощью которых данный интент будет активирован во время диалога. Не обязательно указывать все возможные фразы. Инструменты машинного обучения позволят расширить список фраз похожими фразами.
- Параметры — сущности, которые будут извлекаться из сообщения. Позволяют интерпретировать сообщение пользователя в набор данных, которые можно обработать. Некоторые параметры могут быть обязательными и если пользователь не объявит их во время диалога, система сама спросит его об этих параметрах. Стандартный вид вопрос можно задать в поле Default Prompt. Также в качестве данных для обработки можно использовать оригинальное значение, предоставленное пользователем. Для этого надо в столбце value выбрать интересующую сущность с пометкой original.
- Response — сообщение, которое будет возвращено пользователю. Существует два типа сообщений: статический и динамический. Первый возвращает неизменяемый текст, определенный в поле Text Response. Вторым вариантом позволяет отправить HTTP запрос на сторонний сервер, который вернет текстовый ответ.
- Fulfillment — здесь можно указать использовать ли заполнение слотов и использование стороннего сервера для формирования ответа.

В рамках моей практической работы был создан интент на действие пользователя о получении расписания занятий.

После того как интент был создан, необходимо определить список сущностей, которые могут извлекаться из сообщения пользователя. Для этого необходимо перейти во вкладку Entities и создать сущность, указав ее имя и значения для данной сущности. Также для значений можно определить синонимы, который также будут распознаны. Также можно разрешить автоматическое распознавание значений, которые не были явно указаны в сущности.

Если предполагается использовать динамические сообщения для ответа, то необходимо настроить соединение со сторонним сервером. Для этого следует перейти во вкладку Fulfillment на боковой панели и указать в поле URL ссылку на ваш сервер. Dialogflow будет отправлять POST запрос на сторонний сервер. Также можно настроить авторизацию, чтобы защитить агента от несанкционированного доступа. Также можно в этой вкладке можно напи-

сать код на языке JavaScript выполняющий обработку данных от диалогового агента. Данный код будет выполняться на облачном сервисе Google Cloud.

Чтобы диалоговый агент общался с пользователем, необходим интерфейс. В качестве этого интерфейса могут выступать различные платформы, например мессенджер или цифровой помощник. Чтобы настроить данное взаимодействие, необходимо перейти во вкладку Integrations. Dialogflow поддерживает множество платформ, таких как:

- Google Assistant (в том числе и голосовой помощник)
- Facebook Messenger
- Amazon Alexa
- Viber
- Telegram

Полный список возможных интеграций представлен на рисунках 10 и 11

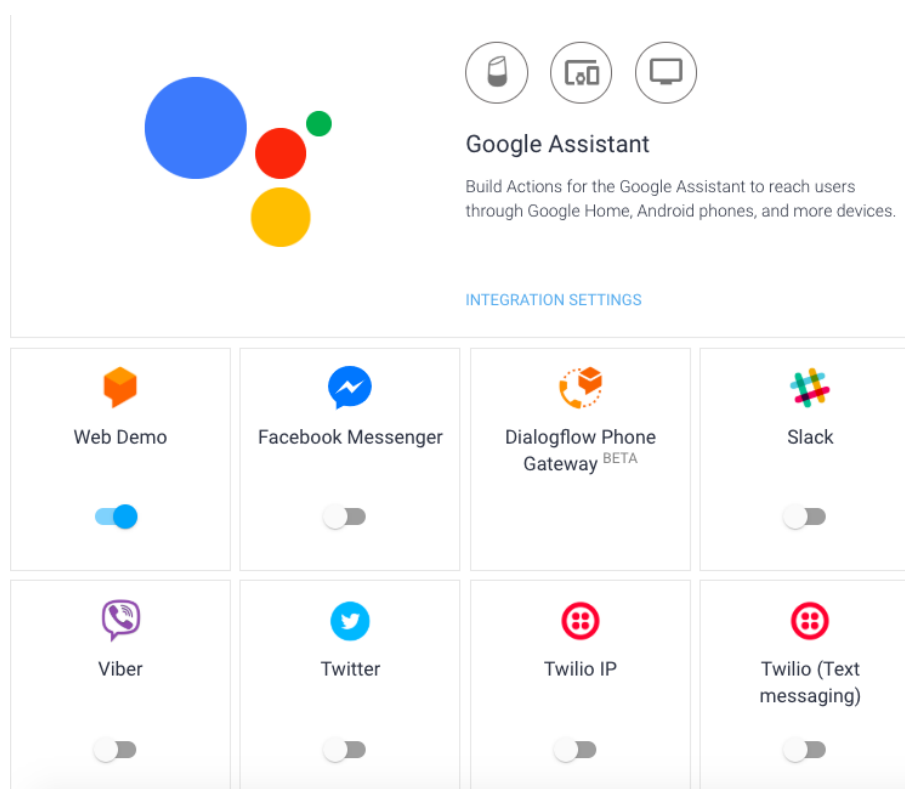


Рисунок 10 – Список платформ, поддерживающих интеграцию с Dialogflow

В своей практической работе я использовал Telegram Messenger, поэтому мне необходимо передать Dialogflow токен Telegram-бота, чтобы завершить интеграцию.

Диалоговый агент улучшает понимание человеческой речи с каждым со-

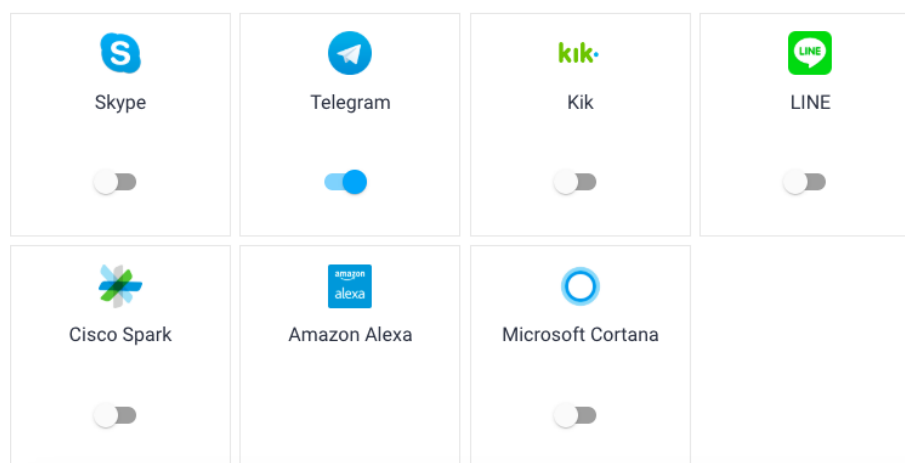


Рисунок 11 – Список платформ, поддерживающих интеграцию с Dialogflow

общением, с помощью технологий машинного обучения. Однако его можно дополнительно обучить с помощью датасетов, собранных из предыдущих диалогов. Для этого необходимо перейти во вкладку Training, нажать кнопку Upload и загрузить текстовый файл. В файле каждая фраза диалога должна быть на одной строке. Также можно обучать бота на основе предыдущих диалогов. Достаточно выбрать диалог и выбрать фразы, которые запустили верный интент.

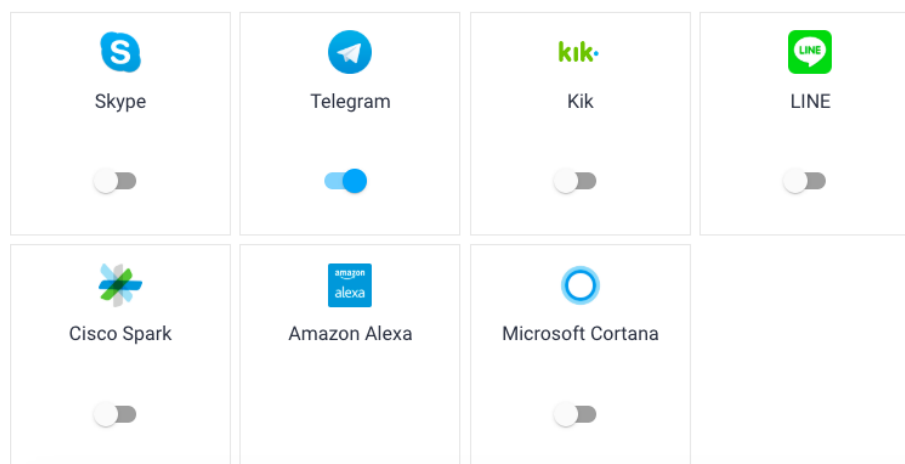


Рисунок 12 – Список платформ, поддерживающих интеграцию с Dialogflow

Так как задача диалогового агента, созданного в практической работе — возвращать расписание занятий, то необходимо хранить расписание в конкретном месте. Можно хранить расписание в сущностях диалогового агента, однако это нецелесообразно из-за большого количества данных. Решено было использовать NoSQL базу данных MongoDB. Для обработки запросов от диалогового агента был разработан HTTP сервер, написанный на языке Python

с использованием фреймворка Flask.

Функция `get_schedule()` получает JSON файл из запроса, извлекает параметры даты, номера курса, специализации и номера группы, затем передает их функцию `get_data_from_database()`

```
1 @app.route('/webhook', methods=['GET', 'POST'])
2 def get_schedule():
3     data = request.get_json(silent=True, force=True)
4     course = data['queryResult']['parameters']['course']
5     spec = data['queryResult']['parameters']['specialization']
6     date = data['queryResult']['parameters']['date']
7     group_number = data['queryResult']['parameters']['groupNumber']
8     response = get_data_from_database(course, spec, date, group_number)
9     return make_response(jsonify(response))
```

Функция `get_data_from_database(course, spec, date, group_number)`, проверяет какая сейчас неделя (числитель или знаменатель) с помощью функции `is_num(date)`, выполняет запрос в базу по параметрам `date` и `group_number`. Если в данный день занятий нет, то коллекция документов будет пустой и, следовательно, возвращается соответствующее сообщение. Если база данных вернула коллекцию документов, то происходит вызов функции `form_response(res)`

```
1 def get_data_from_database(course, spec, date, group_number):
2     year = date[0:4]
3     month = date[5:7]
4     day = date[8:10]
5     week = datetime.date(int(year), int(month), int(day)).isocalendar()[2]
6     if is_num(date) is False:
7         res =
8         db.schedule.find({"daysOfWeek": week_day[week], "groupNumber": int(group_number)})
9         if res.count() == 0:
10            return {'fulfillmentText': 'Сегодня занятий нет. Можете отдохнуть!'}
11        else:
12            return form_response(res)
13        else:
14            res =
15            db.schedule.find({"daysOfWeek": week_day[week], "groupNumber": int(group_number),
16                               "num": True})
17            print(res)
18            if res.count() == 0:
```

```

19 return {'fulfillmentText': 'В этот день занятий нет. Можете отдохнуть!'}
20 else:
21 return form_response(res)

```

Функция `form_response(res)` извлекает данные из коллекции JSON документов по ключам, предварительно отсортировав ее по номеру занятия:

- `subjectName` — название предмета
- `type` — тип занятия (лекция, практика)
- `teacherName` — фамилия и инициалы преподавателя
- `classroom` — номер аудитории
- `subGroup` — номер подгруппы, если это практическое занятие
- `timePeriod` — время начала и время окончания занятия

```

1 def form_response(res):
2     result = ''
3     for s in sorted(res, key=lambda i: i["lesson"]):
4         print(s['subjectName'], full_type[s['type']], sep=', ')
5         result += s['subjectName'] + ', ' + full_type[s['type']] + '\n'
6         print(s['teacherName'])
7         result += s['teacherName'] + '\n'
8         print(s['classroom'])
9         result += s['classroom'] + '\n'
10        if s['subGroup'] != 0:
11            print(str(s['subGroup']) + 'подгруппа')
12            result += str(s['subGroup']) + 'подгруппа' + '\n'
13            print(s['timePeriod'])
14            result += s['timePeriod'] + '\n'
15            print('-----')
16            result += '-----' + '\n'
17        return {'fulfillmentText': result}

```

Функция `is_num(date)`, получает на вход дату, узнает по этой дате номер недели и проверяет его на четность. Если неделя четная, то это знаменатель, иначе числитель.

```

1 def is_num(date):
2     year = date[0:4]
3     month = date[5:7]
4     day = date[8:10]
5     week = datetime.date(int(year), int(month), int(day)).isocalendar()[1]

```

```
6  if week % 2 == 1:
7  return True
8  else:
9  return False
```

После вышеперечисленных действий, управление передается обратно функции `get_schedule()`, где формируется JSON документ, который будет отправлен диалоговому агенту.

## ЗАКЛЮЧЕНИЕ

В данной курсовой работе были рассмотрены основные виды диалоговых систем, принципы взаимодействия и основы построения. Также было ознакомление с платформой создания диалоговых агентов Dialogflow, на примере создания простого диалогового агента, его настройки и взаимодействия со сторонним HTTP сервером, написанным на языке программирования Python с использованием фреймворка Flask и NoSQL базы данных MongoDB.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 ?., . . ?????????????? ?????????? ???????? ?? ?????????????? ?????? ? ?????? ??????? / . . ?. — ??? ???, 2017.
- 2 Jurafsky, D. Speech and Language Processing / D. Jurafsky, J. H. Martin. — 2017. — An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

ПРИЛОЖЕНИЕ А

Нумеруемые объекты в приложении