Лекция 2.3.5. Создание чат-бота. Проектирование основного тела. Интеграция в социальные сети

Что такое чат-боты и их виды

Скорее всего, хоть раз в жизни, вы уже имели дело с чат-ботом, порой даже не понимая, что это был он. Например, когда мы ищем на компьютере информацию о каком-нибудь продукте, на экране часто появлялось окно с вопросом, не нужна ли вам помощь, или когда собираемся на концерт и заказываем такси через чат со своего смартфона. Может быть, заказывали кофе из соседнего кафе с помощью голосовой команды, и нам сообщали, когда заказ будет готов и сколько он будет стоить. Все это примеры случаев, в которых мы могли общаться с чат-ботом.

Но что же такое чат-бот? Существует множество определений данного слова:

- Чат-бот (англ. chatbot) – это программа, которая имитирует реальный разговор с пользователем.
- Чат-боты – это специальные аккаунты, за которыми не закреплен какой-либо человек, а сообщения, отправленные с них или на них, обрабатываются внешней системой.
- Чат-бот – это форма разговорного искусственного интеллекта, предназначенная для упрощения взаимодействия человека с компьютерами.
- Чат-бот – это умная программа, которая живет в мессенджерах и выполняет разные функции.

Но в нашем курсе будем использовать следующее определение:

Чат-бот – это программа, участвующая в разговоре с передачей права голоса, целью которой является интерпретация входного текста или речи и вывод соответствующих полезных ответов.

Чат-боты бывают двух основных видов:

- **Декларативные чат-боты, ориентированные на задания**, – это программы единственного назначения, основной целью которых является выполнение одной функции. Используя правила, NLP и – в меньшей степени – технологию машинного обучения, они отвечают на запросы пользователей автоматически, но делают это в режиме диалога. Общаться с такими чат-ботами нужно очень структурировано, с учетом специфики, поэтому в основном они используются для выполнения функций поддержки и обслуживания, например, в интерактивных, полнофункциональных сервисах

вопросов и ответов. Чат-боты, ориентированные на задания, могут отвечать на стандартные вопросы, например о часах работы, или выполнять простые операции без большого числа разных переменных. Несмотря на то, что в них используются принципы NLP для того, чтобы пользователи могли общаться с ними в режиме диалога, их возможности достаточно ограничены. В настоящее время такие чат-боты являются наиболее распространенными.

- **Предиктивные чат-боты на основе данных, работающие в режиме диалога,** часто называются виртуальными или цифровыми помощниками. Они обладают более развитыми, интерактивными и персонализированным возможностями, чем чат-боты, ориентированные на задания. Эти чат-боты учитывают контекст и используют принципы понимания естественного языка, NLP и машинное обучение, чтобы обучаться в процессе работы. Они применяют предсказательные и аналитические способности для персонализации на основе профилей пользователей и их поведения в прошлом. Цифровые помощники могут изучать предпочтения пользователя в течение времени, предоставлять рекомендации и даже предугадывать потребности. Они могут не только отслеживать данные и намерения, но и инициировать диалог. Примерами предиктивных чат-ботов, основанных на данных и ориентированных на потребителей, являются Siri от Apple и Alexa от Amazon.

Сегодня почти каждая компания имеет чат-бота для взаимодействия с пользователями, ведь те, в свою очередь, обладают многими уникальными и действительно полезными функциями:

1) Поддержка клиентов. Чат-бот поможет заменить неудобный FAQ на сайте, который иногда не сразу можно увидеть, сможет ответить на типовые вопросы клиента. Бот может работать 24 часа в сутки и разгрузит сотрудников.
2) Клиентский сервис. С помощью чат-бота можно делать покупки и запрашивать услуги. В розничной торговле, с постоянным расширением ассортимента, труднее искать конкретные товары. После небольшого анализа бот поймет, что интересует клиента, и отправит прямую ссылку.
3) Маркетинг. Чат-бот – это еще один маркетинговый инструмент, который поможет распространять контент, поддерживать лояльность клиентов и собирать аналитику. С помощью него можно делать рассылки, информировать клиентов об акциях, собирать комментарии о товарах или услугах, качестве обслуживания.
4) Работа внутри компании. Чат-боты помогают оптимизировать в работе такие процессы как: бронирование переговоров, информирование

сотрудников о датах отпуска, расписание корпоративного транспорта, сроки зарплаты и т.п.

5) И многое другое. Возможности безграничны.

История чат-ботов

Несмотря на стремительное распространение, чат-боты – это не современное изобретение, хотя кажется, что появились они относительное недавно. Первые чат-боты появились более полувека назад, и развивались на протяжении многих лет, а началось все с Алана Тьюринга.

Отцом основателем чат-ботов можно смело назвать известного английского математика и криптографа Алана Тьюринга. Многие знают его, как человека, который взломал немецкую шифровальную машину «Энигма». Алан Тьюринг еще в далеком 1941 году начал заниматься теорией машинного интеллекта, которая к 1947 году превратилась в «компьютерный интеллект» – прообраз современного искусственного интеллекта.

Именно он в 1950 году написал научную статью под названием «Вычислительные машины и интеллект», в которой заявил, что компьютерная программа может думать и говорить как человек, а чтобы доказать это, предложил эксперимент под названием «Имитационная игра», на сегодняшний день известный как тест Тьюринга. В этом эксперименте Тьюринг предлагал человеку, которого назначали судьей, провести беседу по компьютеру с человеком и машиной, которых нельзя было увидеть. Задача судьи заключалась в том, чтобы отличить компьютер от реального человека. Если судья не может сказать, какие ответы принадлежат компьютеру, а какие человеку, то это докажет, что компьютер способен имитировать человеческий язык. Тьюринг полагал, что к 2020 году машины смогут легко пройти его испытания.

Статья и идеи легендарного математика вдохновили многих ученых по всему миру – различные институты и университеты начали разрабатывать собственные машины, способные пройти тест Тьюринга. Одним из таких ученых был Джозеф Вейценбаум, который совместно с компьютерной лабораторией Массачусетского технологического института создал в 1966 году революционную программу-бота ELIZA.

Программа была виртуальным собеседником с обработкой естественного языка – простыми словами первым чат-ботом. ELIZA пародировала диалог с психотерапевтом. И хотя очевидно, что она не обладала искусственным интеллектом, программа выделяла определенные слова-якори и на основе их строила свои вопросы и последующий диалог, который достаточно точно повторял действия врача. Диалог с машиной велся путем

ввода запросов на телепринтере – предке современной клавиатуры и ноутбук. Когда пациент говорил: «Моя мама любит цветы», Элиза отвечала: «Расскажи мне больше о своей матери». Таким образом Элиза побуждала людей больше говорить. Это также создало впечатление, что она понимает человеческую речь так же, как люди.

В 1972 году уже в другом, не менее известном американском университете – Стэнфордском, ученым психиатром Кеннетом Колби был создан еще один виртуальный собеседник – PARRY. Данный чат-бот симулировал диалог человека больного параноидной шизофренией. В этом боте была намного более совершенная диалоговая часть. PARRY проходил несколько серьезных проверок и экспериментов, одним из таких экспериментов стало исследование, в котором принимало участие несколько десятков известных психиатров. Суть исследования была в том, чтобы определить, где настоящий больной, а где машина – в половине случаев доктора не смогли дать правильный ответ.

Интересный факт: виртуальный доктор и виртуальный больной несколько раз встречались и даже «общались». В 1972 году PARRY и ELIZA были подключены друг к другу по Арпанету (предку Интернета) и между ними был проведен виртуальный диалог.

Следующим этапом в развитии чат-ботов стала возможность самообучения, внедрения искусственного интеллекта и более продвинутого НЛП. Еще в начале 80-х началась разработка программы Jabberwacky, ее создатель британский ученый Ролло Карпентер ставил цель пройти тест Тьюринга. Программа была ни чем иным, как виртуальным собеседником, способным поддерживать диалог на различные темы, можно сказать, что это был один из первых развлекательных ботов в истории. Мир этот чат-бот увидел в 1997 году, когда его «выпустили» в онлайн.

В 1995 году программисты, вдохновленные работой Джозефа Вейценбаума и его ботом ELIZA, создали программу A.L.I.C.E. На протяжении практически двух десятилетий это был самый совершенный виртуальный собеседник, который мог вести абсолютно естественный диалог с человеком на более чем 40 000 различных тем. Эта программа несколько раз становилась самой «человечной» программой в мире и получала премию Лебнера.

В наши же дни с появлением современных виртуальных ассистентов (Apple представил Siri в 2011) стало абсолютно ясно, что теперь чат-боты не просто стали частью нашей жизни, а еще и обрели голос. Несмотря на это, в то время мы на самом деле не осознавали, насколько эта индустрия изменится с появлением мессенджеров.

WhatsApp появился в 2009, Kik и Viber – в 2010, WeChat и Facebook Messenger в 2011, Telegram в 2013. Появление чат-ботов на этих платформах было всего лишь делом времени.

Проектирование основного тела чат-бота

Как уже говорилось ранее, чат-боты бывают двух основных видов: декларативные и предиктивные. Одни используют методы NLP и являются менее гибкими, но более специализированными, а другие наоборот – не способны решать прикладные задачи и отвечать на узкоспециализированные вопросы, но способны самообучаться и подстраиваться под своего пользователя. В рамках обработки естественного языка нам подходит первый тип, о нем и пойдет дальше речь. Мы научимся создавать простых поисковых чат-ботов используя методы NLP, а также знания, полученные из предыдущих занятий. Подключим сразу все библиотеки и функции, которые нам понадобятся:

```
import string
import nltk
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.corpus.reader.plaintext import PlaintextCorpusReader
nltk.download('punkt')
try:
  import natasha as nt
except:
  !pip install natasha
  import natasha as nt
```

Начнем с самого простого – приветствия собеседника. Как мы уже выяснили из истории чат-ботов, каждый из них в большей или меньшей степени старается имитировать поведение реального человека. А что мы делаем обычно перед началом разговора? Приветствуем собеседника. Так и наш бот должен отвечать на реплики-приветствия. Если пользователь поприветствует нашего бота, то он поздоровается в ответ. ELIZA использует простое сопоставление ключевых слов. Будем использовать ту же идею и создадим функцию приветствия:

```
#Функция приветствия
def welcome(user_response):
  welcome_response = ["Приветствую!", "Привет!", "Здравствуйте",
 "Привет, я чат-бот! Готов ответить на ваши вопросы!"]
  for word in user_response.split():
      if word in welcome_input:
          return random.choice(welcome_response)
```

В ней пропишем список, который будет хранить ответы нашего бота на возможные приветствия пользователя. В том случае, если в запросе пользователя будет какое-либо слово из списка welcome_input, который хранит слова «привет», «приветствую» и так далее, то он вернет случайный ответ из предоставленных.

Аналогичным образом можно создать функцию прощания, которая одновременно будет являться командой прекращения работы нашего бота. Обычно чат боты, которых часто можно встретить на различных сайтах или мессенджерах не имеют данной особенности, ведь их задача заключается в том, чтобы работать на постоянной основе и в любой момент быть готовыми ответить на запрос пользователя, однако, может возникнуть ситуация, когда данная функция необходима:

```
#Функция прощания
def goodbye(user_resonse):
  goodbye_response = ["С нетерпением буду ждать вас!", "До свида
ния!", "Хорошего дня!"]
  for word in user_response.split():
      if word in goodbye_input:
          return random.choice(goodbye_response)
```

На этом формальности подошли к концу и можно приступить к реализации нашей главной цели – научить чат-бота отвечать на вопросы пользователя, касаемые определенной темы. В качестве примера будем использовать статью из википедии, которая посвящена русской императрице Екатерине II. Пусть наш пользователь будет спрашивать какие-нибудь факты из её биографии, а наш чат-бот стараться ответить на них цитатой из текста.

Для этого, в первую очередь, нужно написать функцию нормализации текста для русского языка, используя все доступные и известные нам методы:

```
#Функция нормализации текста
def Normalize(text):
  #Инициализируем вспомогательные объекты библиотеки natasha
  segmenter = nt.Segmenter()
  morph_vocab = nt.MorphVocab()
  emb = nt.NewsEmbedding()
  morph_tagger = nt.NewsMorphTagger(emb)
  ner_tagger = nt.NewsNERTagger(emb)

  #Убираем знаки пунктуации из текста
  word_token = text.translate(str.maketrans("", "", string.punct
uation)).replace("—", "")

  #Преобразуем очищенный текст в объект Doc и
  doc = nt.Doc(word_token)
  doc.segment(segmenter)
```

```
  doc.tag_morph(morph_tagger)
  doc.tag_ner(ner_tagger)

  #Приводим каждое слово к его начальной форме
  for token in doc.tokens:
    token.lemmatize(morph_vocab)
  resDict = {_.text: _.lemma for _ in doc.tokens}

  #Возвращаем результат в виде списка
  return [resDict[i] for i in resDict]
```

В качестве параметра этой функции будем передавать текст нашего будущего корпуса, который нам предстоит собрать, а на выходе получать список токенов, которые уже прошли процесс очистки от пунктуации и процесс лемматизации. Убирать стоп-слова из предложений в этой функции мы не будем, ведь они автоматически удалятся преобразователем `TfidfVectorizer()`.

Дело осталось за малым и скоро наш чат-бот начнет свою полноценную работу, однако на данный момент он не умеет сравнивать запрос пользователя с имеющимся у него корпусом, а в этом нам поможет метод косинусного сходства:

```
#Функция ответа на запрос
def response(user_response):
    robo_response=''#Будущий ответ нашего бота
    sent_tokens.append(user_response)#Временно добавим запрос по
льзователя в наш корпус.
    TfidfVec = TfidfVectorizer(tokenizer=Normalize)#Вызовем вект
оризатор TF-IDF
    tfidf = TfidfVec.fit_transform(sent_tokens)#Создадим вектора
    vals = cosine_similarity(tfidf[-
1], tfidf)#Через метод косинусного сходства найдем предложение с
 наилучшим результатом
    idx=vals.argsort()[0][-2]#Запомним индексы этого предложения
    flat = vals.flatten()#сглаживаем полученное косинусное сходс
тво
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0): #Если сглаженное значение будет равно 0, т
о ответ не был найден
        robo_response=robo_response+"Извините, я не нашел ответа
..."
        return robo_response
    else:
        robo_response = robo_response+sent_tokens[idx]
        return robo_response
```

Стоит отметить, что функция `argsort()` сортирует весь список, содержащий косинусные сходства векторов, от меньшего к большему, а так как последнее значение у нас будет равно 1 (это и есть запрос пользователя), то брать нужно предпоследнее значение.

На этом создание вспомогательных функций подошло к концу. Осталось лишь собрать корпус и запустить нашего бота, не забыв создать списки, которые будут хранить варианты возможных приветствий и прощаний пользователя:

```
#Создадим корпус
newcorpus = PlaintextCorpusReader('newcorpus/', r'.*\.txt')

data = newcorpus.raw(newcorpus.fileids())
sent_tokens = nltk.sent_tokenize(data)

#Зададим списки с возможными приветствиями и прощаниями пользова
теля
welcome_input = ["привет", "ку", "прив", "добрый день", "доброго
 времени суток","здравствуйте", "приветствую"]
goodbye_input = ["пока", "стоп", "выход", "конец", "до свидания"
]

#Основное тело бота
flag=True #Флаг, отвечающий за работу бота

print("Чат-
бот: Привет, меня зовут Боб и я готов рассказать вам интересные
факты об императрице Екатерине II!")
print("Чат-
бот: Если вы устанете общаться со мной, то напишите мне ключевое
 слово 'стоп'.")

while(flag==True):
    user_response = input()
    user_response = user_response.lower()
    if user_response in welcome_input:
      print("Чат-бот: "+welcome(user_response))
    elif user_response in goodbye_input:
      flag = False
      print("Чат-бот: "+goodbye(user_response))
    else:
      print("Чат-бот: ",end="")
      print(response(user_response))
      sent_tokens.remove(user_response)#Удаляем запрос из корпус
a
```

Вот и все, наш первый чат-бот создан и готов работать!

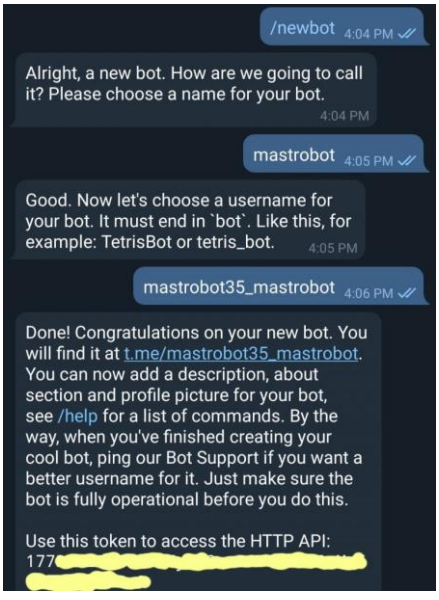
Немного пообщавшись с ним, можно убедится, что функционирует он вполне хорошо. Главное правильно задавать вопросы:

```
Чат-бот: Привет, меня зовут Боб и я готов рассказать вам интересные факты об императрице Екатерине II!
Чат-бот: Если вы устанете общаться со мной, то напишите мне ключевое слово 'стоп'.
Привет
Чат-бот: Привет, я чат-бот! Готов ответить на ваши вопросы!
Когда родилась Екатерина?
Чат-бот: София Фредерика Августа Ангальт-Цербстская родилась 21 апреля (2 мая) 1729 года в немецком городе Штеттин
Как Екатерина взошла на престол?
Чат-бот: Она взошла на престол, следуя прецеденту, созданному Екатериной I, сменившей своего мужа Петра Великого в
Пока
Чат-бот: Хорошего дня!
```

Интеграция чат-бота в социальные сети

Теперь, когда у нас появился полноценный чат-бот его можно пускать на просторы интернета. Для этого существуют множество способов: от внедрения его в социальные сети, до размещения на каком-либо сайте в качестве виртуального помощника. В нашем курсе мы покажем вам как интегрировать созданного бота в Telegram и для этого нам понадобится библиотека `aiogram`. `Aiogram` – это простой и полностью асинхронный фреймворк для Telegram Bot API, написанный на Python 3.7 с помощью `asyncio` и `aiohttp`. Вдаваться в подробности этой библиотеки мы не будем, а воспользуемся только самыми необходимыми для работы нашего бота функциями.

В первую очередь нам необходимо зарегистрировать нашего бота в той социальной сети, в которой мы собрались его разместить. К сожаленью, простое создание нового аккаунта нам не подойдет, ведь для ботов процесс их регистрации проходит иначе, чем у «живых» пользователей. Для этого зайдите в телеграмм и найдите там канал @BotFather, именно он отвечает за регистрацию новых ботов, и напишите команды /start и /newbot последовательно. Должно получится вот так:



После этого дадим нашему боту никнейм В примере его назвали mastrobot, и имя пользователя, допустим mastrobot35_mastrobot. Первое имя будет высвечиваться при общении с нашим ботом, а второе пойдет в ссылку, по которой мы найдем личный чат. Помимо ссылки, мы так же получим и API токен нашего бота, вот он то нам как раз и нужен при написании различных команд.

У BotFather можно также запросить множество других интересных вещей. Например, изменить изображение профиля бота, добавить описание и многое другое, но это уже на ваше усмотрение. Мы же приступим к основному коду нашего первого телеграмм бота.

Давайте создадим такого чат-бота, который будет как попугай повторять сообщения, которые мы ему отправляем. Для этого подключим необходимые функции из библиотеки (ВНИМАНИЕ В GOOGLE COLLAB ДАННЫЙ КОД НЕ РАБОТАЕТ):

```
try:
  from aiogram import Bot, Dispatcher, executor, types
except:
  !pip install aiogram
  from aiogram import Bot, Dispatcher, executor, types
```

Далее инициализируем нашего бота в системе, т.е., говоря простым языком, говорим ему «зайти» на свой аккаунт. Для этого понадобится наш API токен. Обычно его нельзя показывать, как и пароли, а потому хранится он в отдельном файле config.py, но, чтобы вам было удобнее сдать будущую практическую работу, опустим данный момент:

```
#Инициализация бота
bot = Bot(token = '1810420403:AAHa0MiYvgvYyaxupRg6hyntLyA65mkstH
g')
dp = Dispatcher(bot)
```

Осталось лишь прописать функцию, которая будет считывать полученное сообщение из личного чата и обратно возвращать его пользователю. Реализуется все это через асинхронные функции. Они очень сильно похожи на привычные нам функции, однако их главное преимущество в том, что выполняются они вне зависимости от друг друга, т.е. одна из функций не будет ждать, когда закончится работа другой. А также, при «регистрации» функции как обработчик получаемых сообщений, необходимо навесить на нее декоратор:

```
#Хэндлер для функции
@dp.message_handler()
async def echo(message: types.message):
  await message.answer(message.text)#Возвращаем текст обратно по
льзователю
```

Однако, что, если мы захотим сделать функцию, которая будет вызываться в определенный момент времени по специальной команде? В таком случае при создании и «регистрации» функции необходимо в декораторе прописать то самое ключевое слово, на которое будет реагировать наш чат-бот. Выглядеть это будет следующим образом:

```
#Хэндлер для функции
@dp.message_handler(commands = ['hi'])
async def hi_func(message: types.message):
  await message.answer('Привет, напиши мне что-нибудь!')
```

Обратите внимание также на то, в каком порядке у вас расположены функции в программном коде. Если эхо-функция, выполняющая роль попугая у нашего бота будет стоять перед функцией приветствия, то последняя корректно работать не будет. Дело в том, что первая будет перехватывать полученное сообщение и вне зависимости от его содержания возвращать пользователю отправленное им же сообщение, минуя другие существующие функции. Чтобы этого не случалось, нужно придерживаться некоторого приоритета выполнений, который вы устанавливаете сами!

Последний шаг в разработке нашего бота – это сам его запуск. Данные две строчки позволяют запустить нашего бота и заставить работать до тех пор, пока мы его не выключим.

```
if __name__ == '__main__':
  executor.start_polling(dp, skip_updates=True) #Запуск бота
```

На этом разработка нашего первого чат-бота подошла к концу. Можете начать с ним вести бесконечную зеркальную беседу и проверить работоспособность его функций.

Поздравляю с последним заданием изучаемого курса ДПО!!!

Удачи на защите выпускных проектов!