Диалоговые и вопросно-ответные системы

Основы разработки чат-ботов, часть I

Диалоговая система

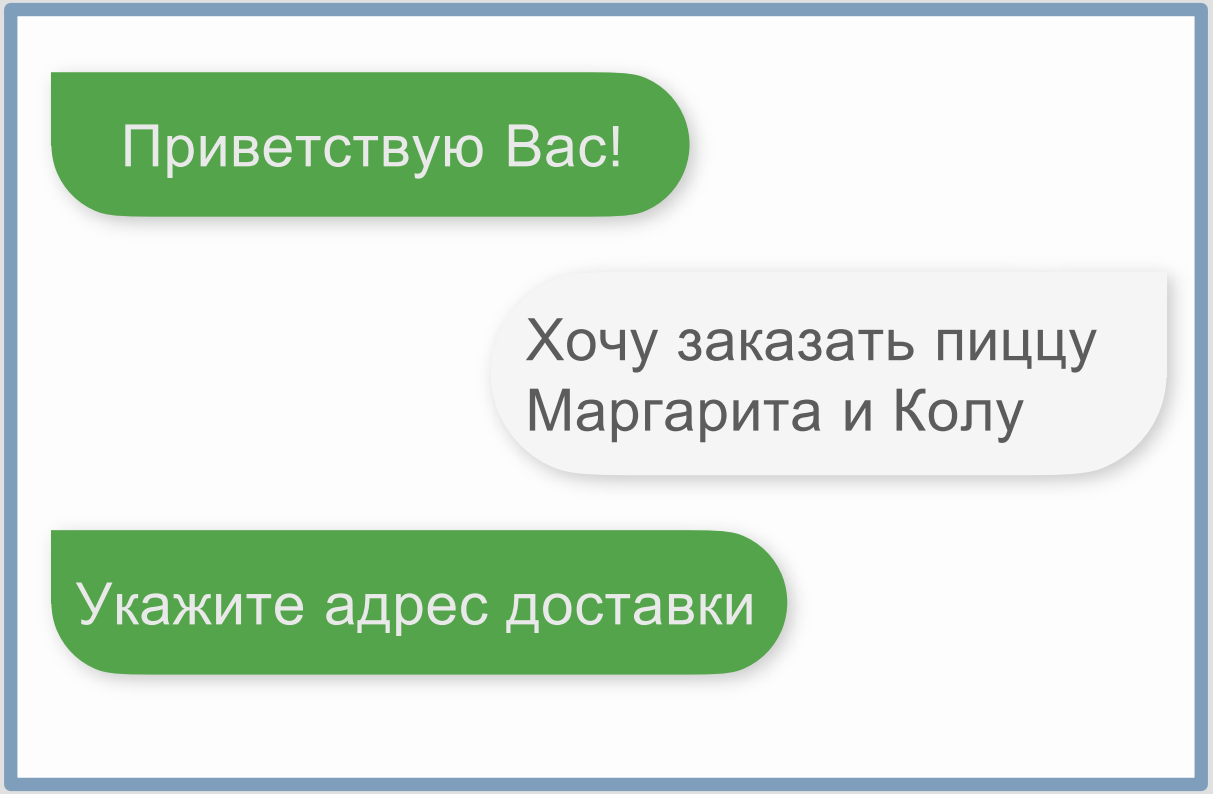
- Диалоговая система — компьютерная система, предназначенная для общения с человеком
- Интерфейс общения может быть произвольным:
  - Текстовый
  - Голосовой
  - Графический
  - Жестовый
  - Тактильный
- Нам прежде всего интересен текстовый
- Часто информация из других форматов переводится для обработки в текст

Чат-боты

- Чат-бот *(chat-bot)* — ключевой элемент текстовых и голосовых диалоговых систем
- Способен вести текстовый диалог с человеком на естественном языке

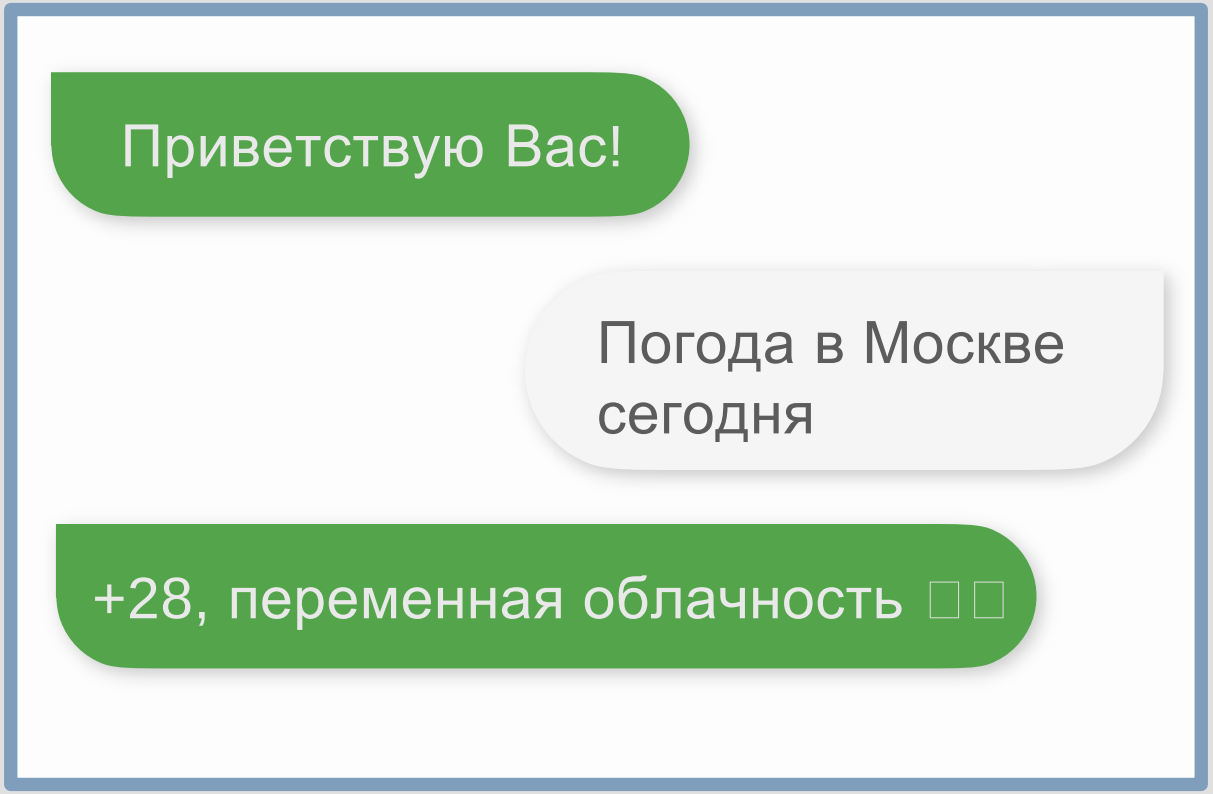
Чат-боты

- Чат-бот *(chat-bot)* — ключевой элемент текстовых и голосовых диалоговых систем
- Способен вести текстовый диалог с человеком на естественном языке

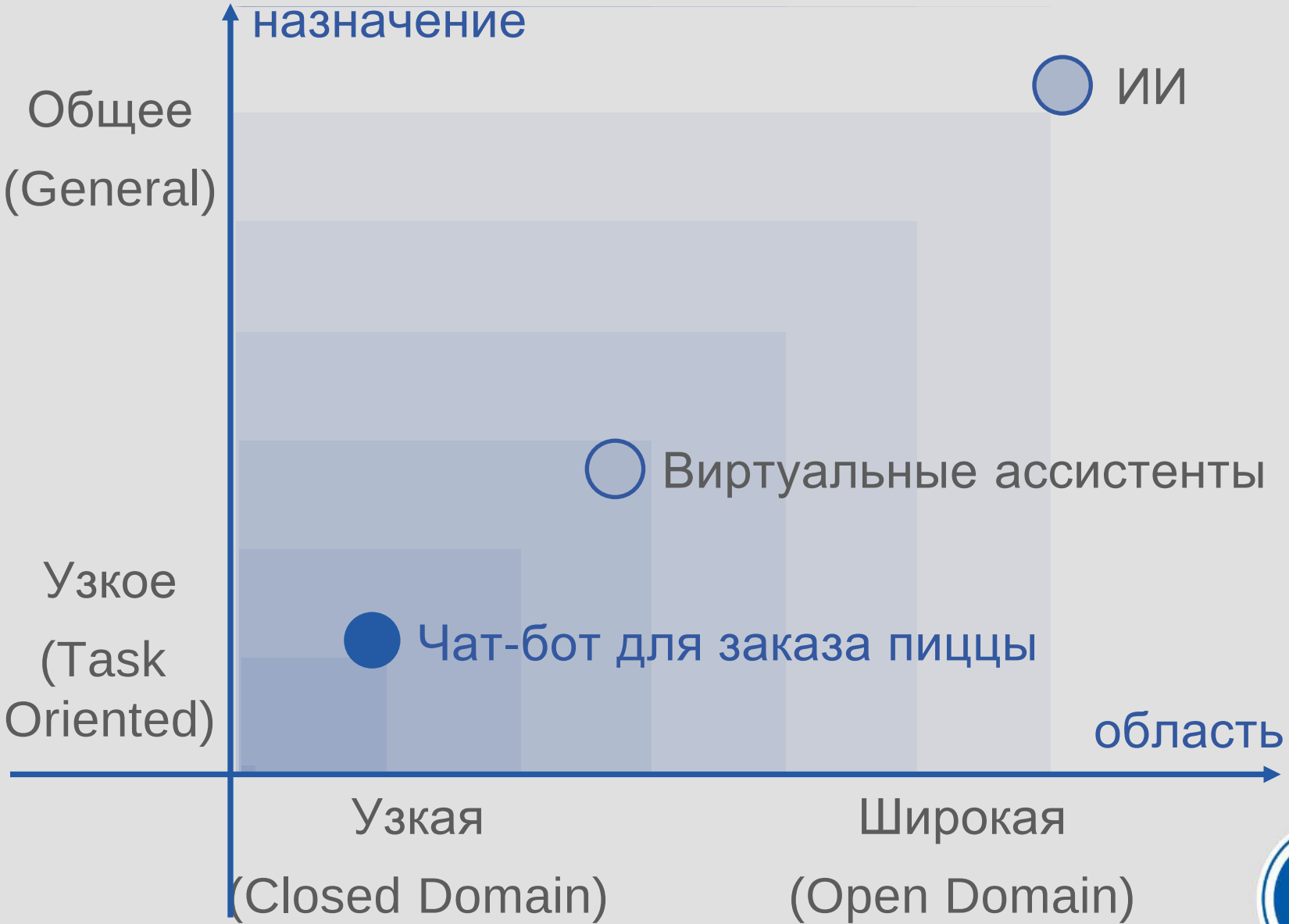


Чат-боты

- Чат-бот *(chat-bot)* — ключевой элемент текстовых и голосовых диалоговых систем
- Способен вести текстовый диалог с человеком на естественном языке



Виды диалоговых систем



Типы ответов в диалоговых системах

- Системы с готовыми ответами *(Retrieval-Based)*:
  - Все возможные ответы подготовлены заранее
  - Задача системы — выбрать наиболее подходящий в данный момент диалога
  - Ответы всегда корректно написаны и осмысленны
  - Ничего нового система сказать не в состоянии
  - Используются почти во всех чат-ботах
  - Важно: полноценную систему общего назначения невозможно сделать на основе готовых ответов

Типы ответов в диалоговых системах

- Системы с готовыми ответами *(Retrieval-Based)*
- Системы с генерацией ответов *(Generative-Based)*:
  - Возможны новые уникальные ответы
  - Качество и адекватность ответов не гарантируются
  - Обычно не используется в бизнес-приложениях
  - Перспективное направление исследований

Где используются чат-боты

- Замена операторов в контакт-центрах
- Замена консультантов при выборе товара/услуги
- Заказ такси, билетов, еды
- Онбординг (например, при регистрации на ресурсе)
- Виртуальные ассистенты

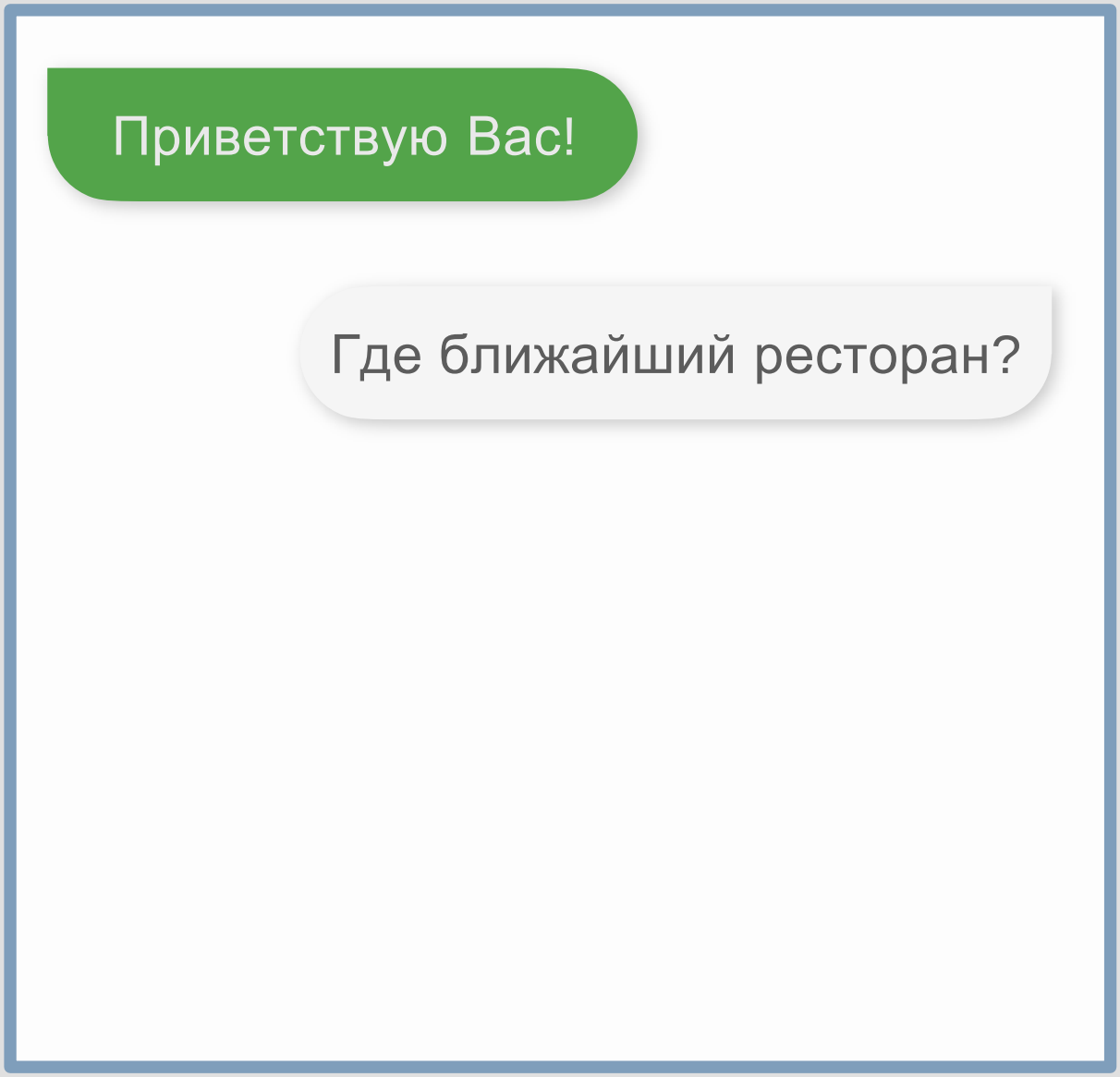
Основные компоненты чат-бота

- Подсистема определения интента *(Intent Detection)* — выделяет из сообщения намерение пользователя
- Подсистема заполнения слотов *(Slot Filling)* — выделяет из сообщения необходимые сущности
- Граф сценария
- Подсистема генерации ответов и действий

Определение интента

- Определения интента — это задача классификации
- Входные данные:
  - Текущая пользовательская реплика
  - Любые другие полезные данные: прошлый интент, текущие сущности, состояние бота, …
- Выходные данные:
  - Интент — метка класса намерения
  - Опционально: степень уверенности классификатора

Определение интента



Определение интента

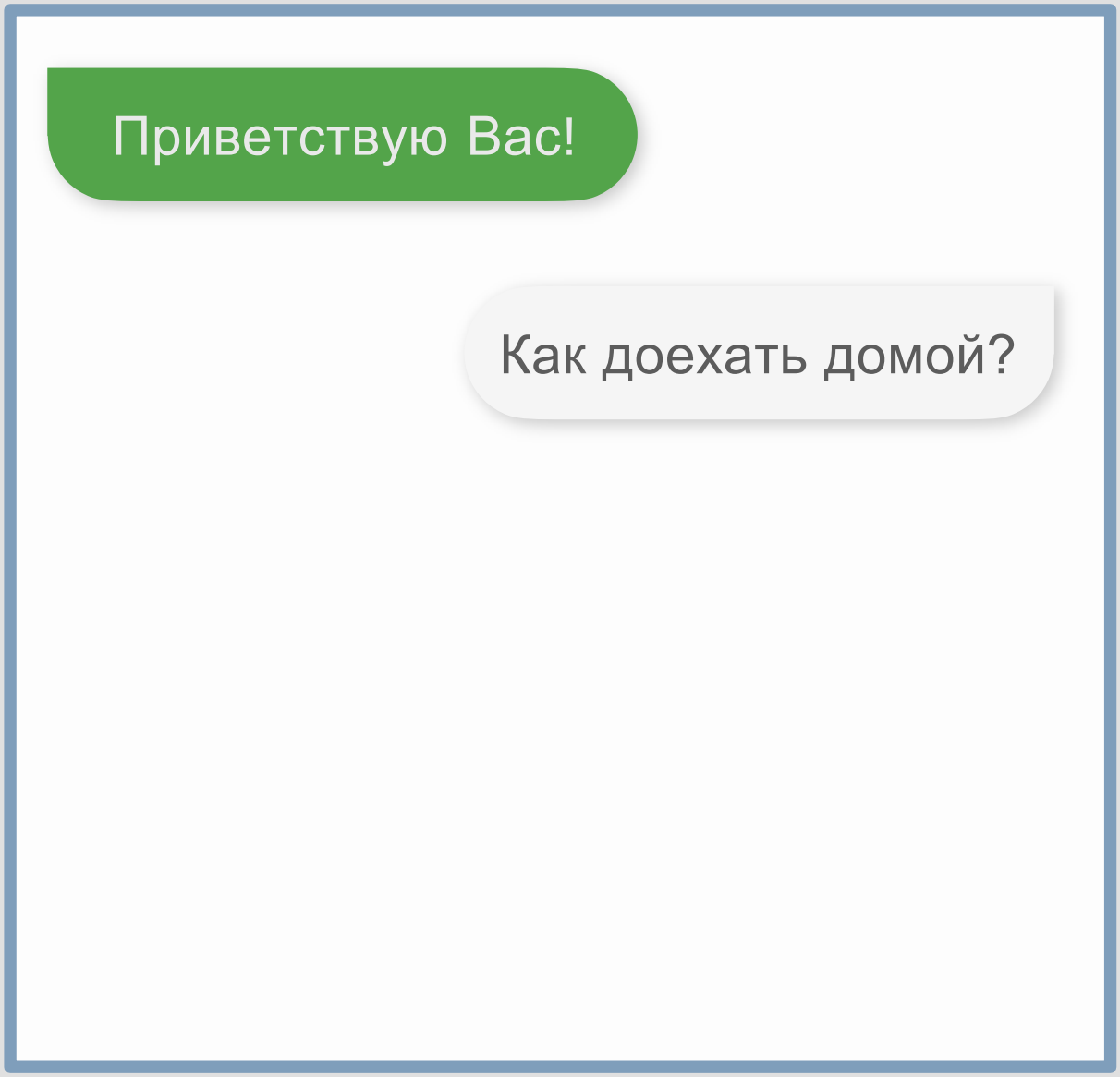
Приветствую Вас!

Где ближайший ресторан?

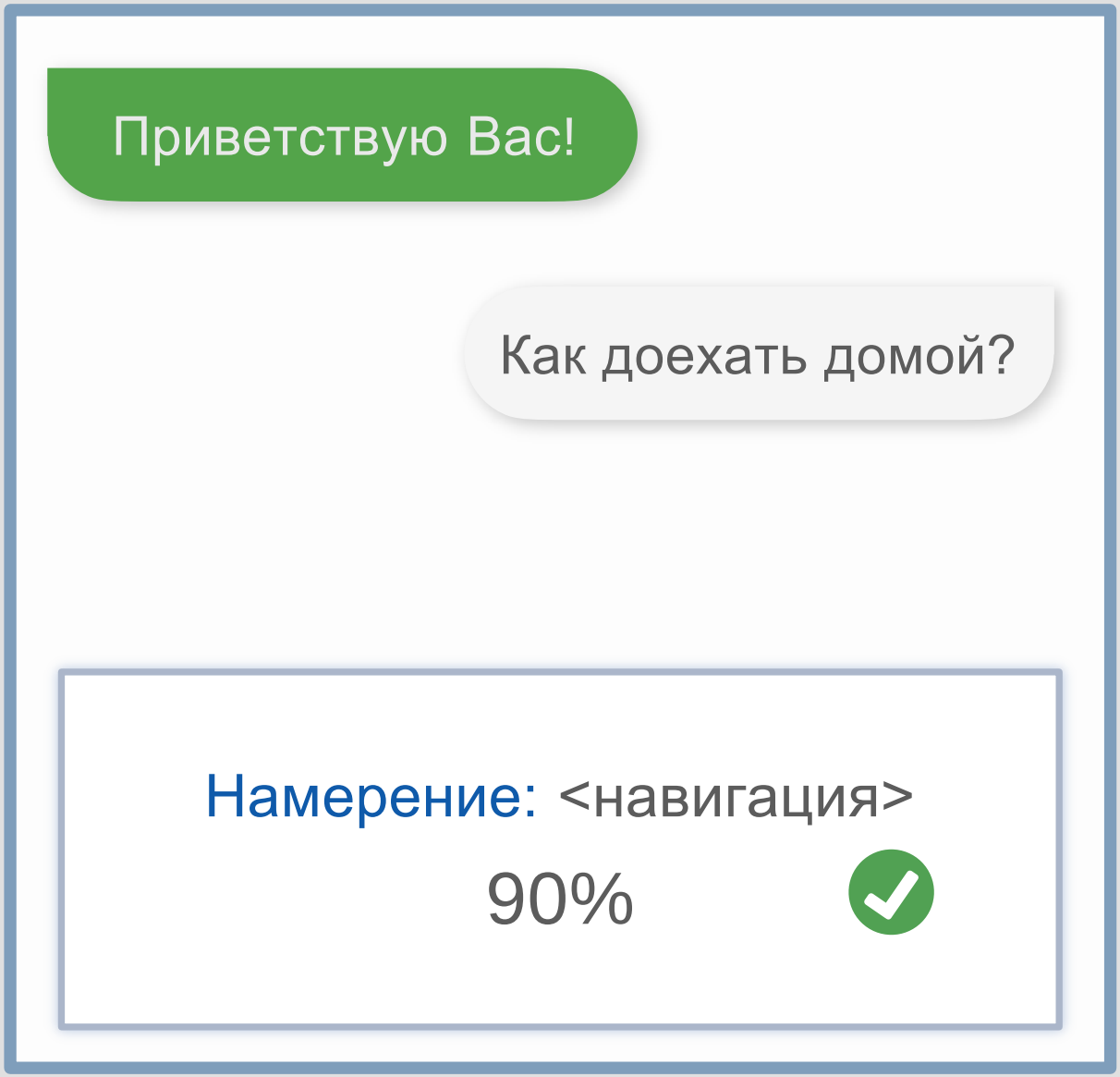
Намерение: <адрес ресторана>

80%

Определение интента



Определение интента



Определение интента

Приветствую Вас!

Какого цвета мой правый носок?

Определение интента

Приветствую Вас!

Какого цвета мой правый носок?

Намерение: <покупка одежды>

5%

Выбор классификатора интента

- Можно строить сложный классификатор:
  - Собрать очень много пользовательских диалогов
  - Разметить интенты
  - Обучать глубокие сети на последовательности токенов реплик с учётом их интентов

Выбор классификатора интента

- Можно строить сложный классификатор:
  - Собрать очень много пользовательских диалогов
  - Разметить интенты
  - Обучать глубокие сети на последовательности токенов реплик с учётом их интентов
- Можно делать проще:
  - Работать только с текущей и прошлой репликами и предыдущим интентом
  - Или работать только с текущей репликой и предыдущим интентом
  - Или работать только с текущей репликой

Выбор классификатора интента

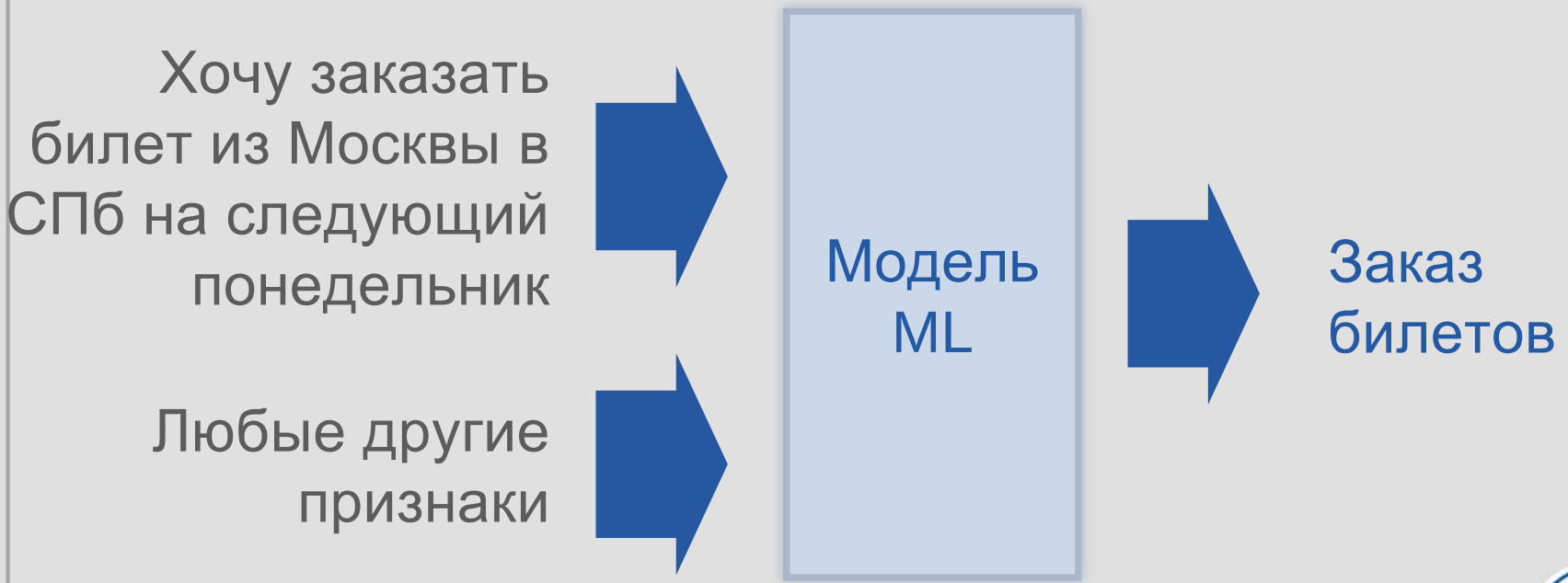
- Простые подходы в большинстве случаев предпочтительнее:
  - Для фиксированного набора интентов можно построить классификатор хорошего качества без нейросетей
  - Не нужно много размеченных реплик
  - Проще обучать и модифицировать классификатор
  - Проще понимать и отлаживать поведение бота
  - Меньшее потребление ресурсов
  - Более высокая скорость работы

Выбор классификатора интента

- Простые подходы в большинстве случаев предпочтительнее:
  - Для фиксированного набора интентов можно построить классификатор хорошего качества без нейросетей
  - Не нужно много размеченных реплик
  - Проще обучать и модифицировать классификатор
  - Проще понимать и отлаживать поведение бота
  - Меньшее потребление ресурсов
  - Более высокая скорость работы
- Это не значит, что нейросети в чат-ботах не нужны!
- Есть задачи в построении диалоговых систем, где без них никак

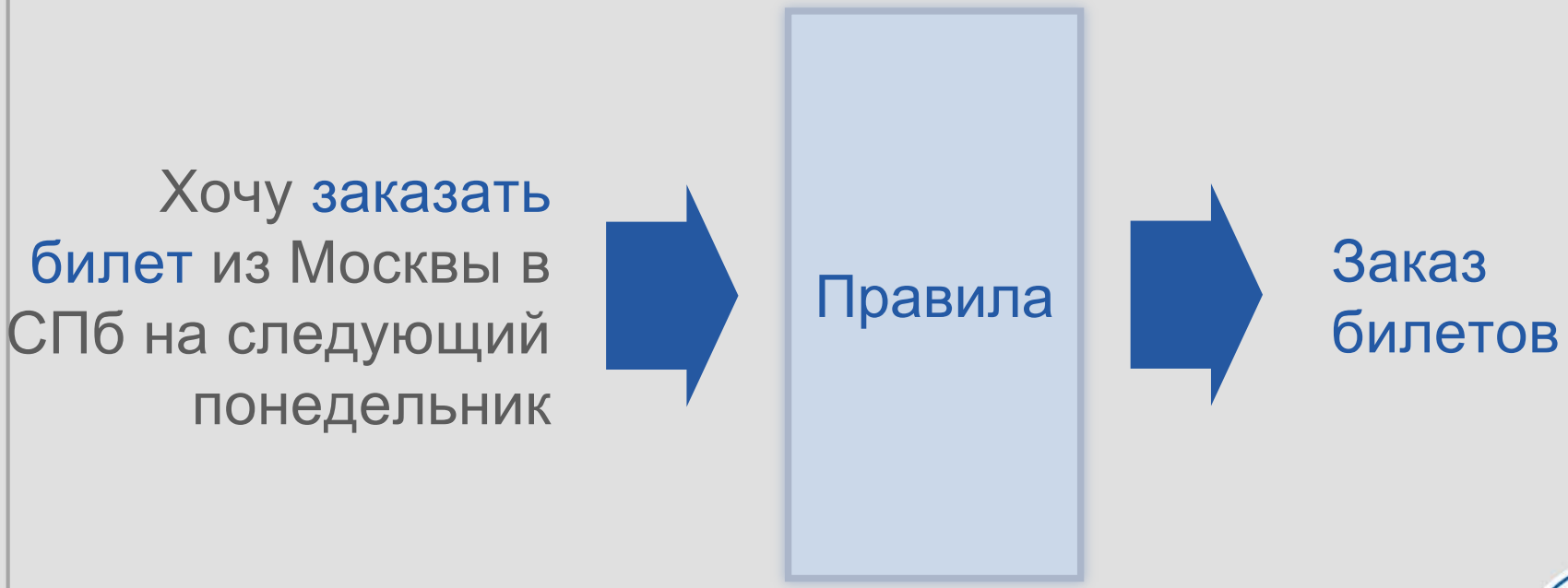
Выбор классификатора интента

- Можно использовать обучаемые модели



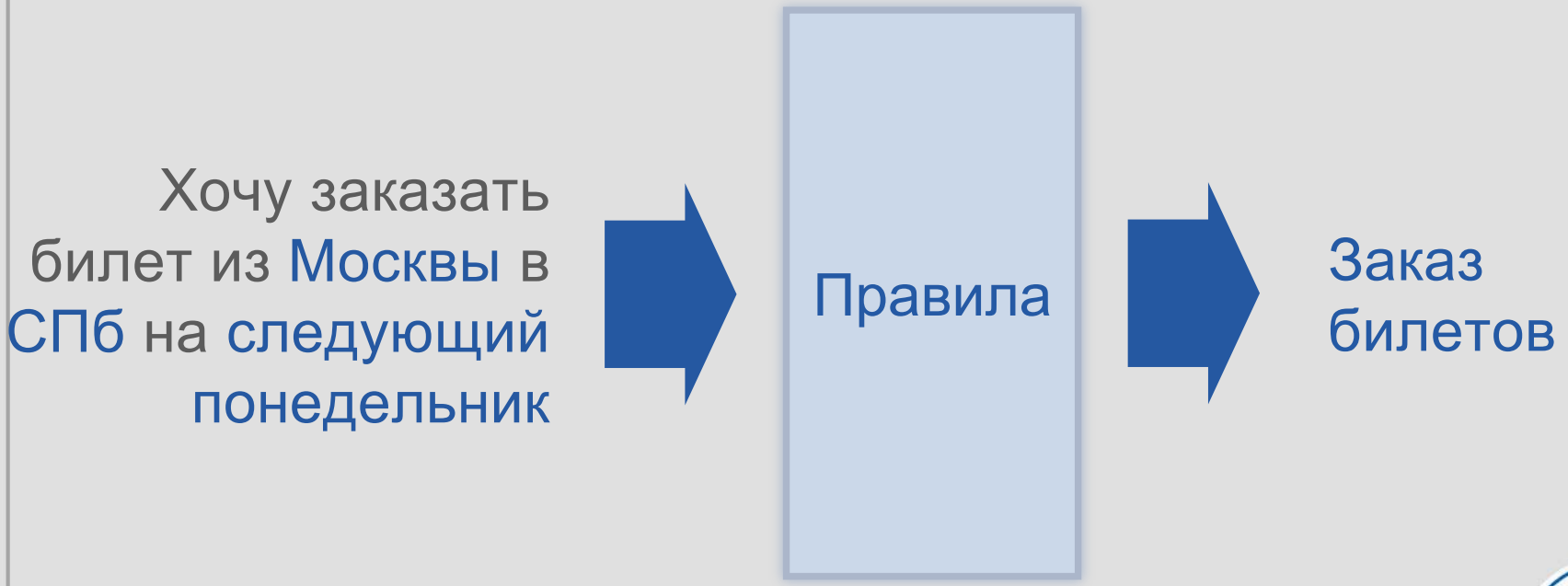
Выбор классификатора интента

- Можно использовать обучаемые модели
- Можно строить определение интента на правилах:
  - По наличию в реплике определённых слов/N-грамм



Выбор классификатора интента

- Можно использовать обучаемые модели
- Можно строить определение интента на правилах:
  - По наличию в реплике определённых слов/N-грамм
  - По наличию определенных сущностей



Выбор классификатора интента

- Можно использовать обучаемые модели
- Можно строить определение интента на правилах:
  - По наличию в реплике определённых слов/N-грамм
  - По наличию определенных сущностей
- На практике часто используются различные комбинации этих подходов

Основные выводы

- Чат-боты — способ организации общения человека с компьютером через текстовый канал
- Очень популярный и часто используемый в бизнесе инструмент
- При построении бота нужно уметь решать задачу классификации

Основы разработки чат-ботов, часть II

Основные компоненты чат-бота

- Подсистема определения интента *(Intent Detection)* — выделяет из сообщения намерение пользователя
- Подсистема заполнения слотов *(Slot Filling)* — выделяет из сообщения необходимые сущности
- Граф сценария
- Подсистема генерации ответов и действий

Заполнение слотов

- Слоты — дополнительные данные, нужные боту для обработки реплики с заданным интентом:

Интент	Заказ билетов
Слоты	<ГО, город отправления> <ГП, город прибытия> <ДО, дата отправления>

- Слоты заполняются на основе сущностей, содержащихся в тексте реплики

Заполнение слотов

- Слоты — дополнительные данные, нужные боту для обработки реплики с заданным интентом:

Интент	Заказ билетов
Слоты	<ГО, город отправления> <ГП, город прибытия> <ДО, дата отправления>

- Слоты заполняются на основе сущностей, содержащихся в тексте реплики:

Хочу заказать билет из Москвы в СПб на следующий понедельник

<ГО>: Москва

<ГП>: Санкт-Петербург

<ДО>: 3.04.<текущий год>

Заполнение слотов

- Слоты заполняются с помощью сущностей, но слоты ≠ сущностям:

Здравствуйте! Я сам живу в Новгороде, теперь вот хочу заказать билет из Москвы в СПб на следующий понедельник, а то друг туда приехал до среды, хочу повидаться. Спасибо!

Заполнение слотов

- Слоты заполняются с помощью сущностей, но слоты ≠ сущностям:

Здравствуйте! Я сам живу в Новгороде, теперь вот хочу заказать билет из Москвы в СПб на следующий понедельник, а то друг туда приехал до среды, хочу повидаться. Спасибо!

<ГО>: Москва

<ГП>: Санкт-Петербург

<ДО>: 3.04.<текущий год>

Заполнение слотов

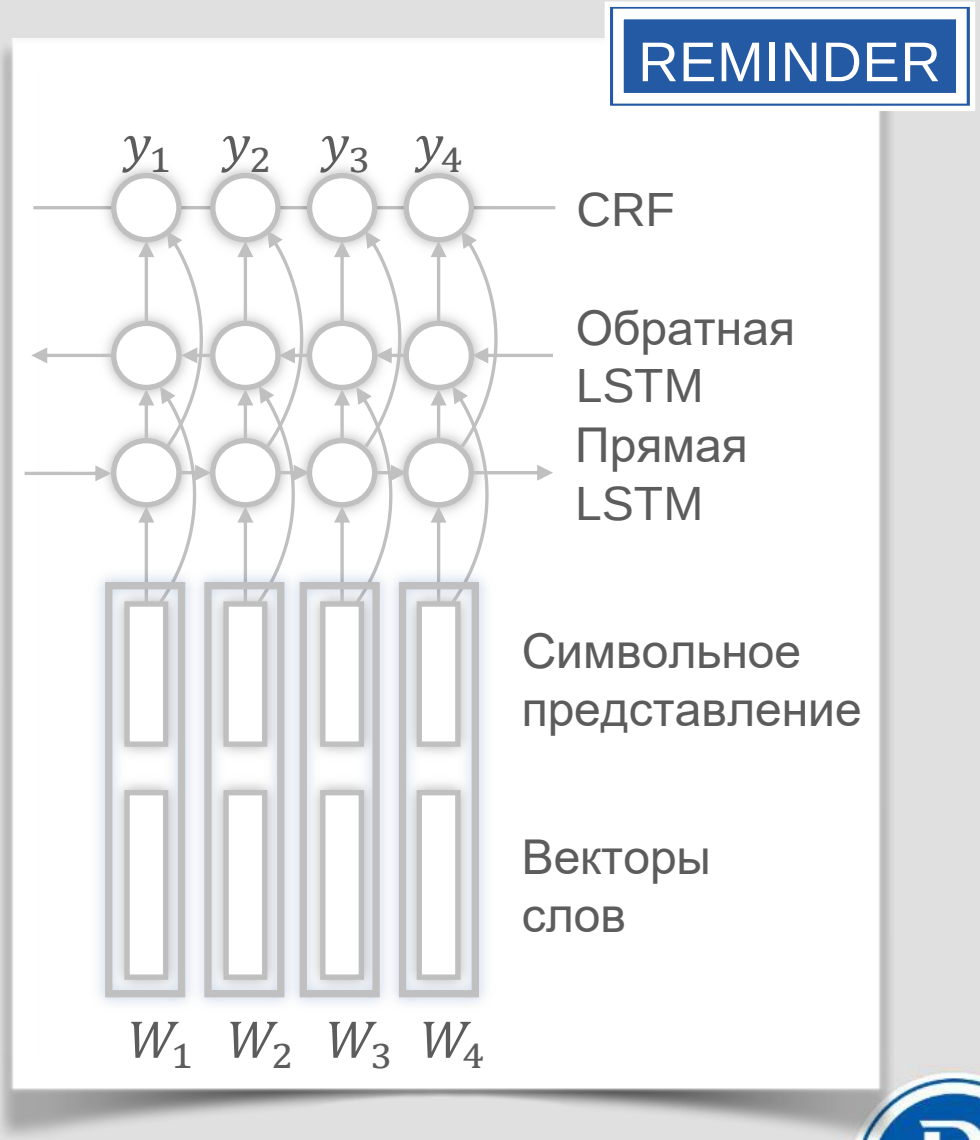
- Слоты могут быть обязательными, а могут быть опциональными:

Интент	Заказ билетов
Обязательные слоты	<ГО, город отправления> <ГП, город прибытия> <ДО, дата отправления>
Опциональные слоты	<КБ, класс билета> <ВО, время отправления>

- Если обязательных нет в реплике, то надо уточнить
- Если опциональные есть — надо учесть, если нет — игнорировать, не беспокоя пользователя

Выделение сущностей для слотов

- Можно использовать обучаемые модели
- Выбираются типы выделяемых сущностей
- Собираются подходящие реплики
- Подготавливается разметка (например, в формате IOB)
- Обучается модель NER, в основе обычно LSTM или Трансформер



Выделение сущностей для слотов

- На практике сущности часто выделяются более простыми методами
  - Мягкое или жёсткое сопоставление по словарю
  - Регулярные выражения
  - Эвристические правила
- Причины те же, что и для выделения интентов:
  - Не нужно много размеченных данных
  - Относительно легко сделать
  - Высокая скорость работы
  - Легко отлаживать и модифицировать
- Стандартные сущности выделяются хорошо

Пример: выделение дат

- Даты можно с высокой точностью выделять с помощью регулярных выражений
- Сперва нужно выделить текстовый фрагмент, содержащий описание даты:

Билет 17.03 …

Хочу на следующих выходных …

Поедем через среду …

Регулярные выражения

17.03

следующих выходных

через среду

Пример: выделение дат

- Даты можно с высокой точностью выделять с помощью регулярных выражений
- Сперва нужно выделить текстовый фрагмент, содержащий описание даты
- Затем нужно конвертировать этот фрагмент в дату:

17.03

следующих выходных

через среду

Регулярные выражения и правила

17.03.<текущий год>

19.03.<текущий год>

20.03.<текущий год>

23.03.<текущий год>

Сценарий чат-бота

- Сценарий является основой логики чат-бота
- Сперва для текущей реплики определяются интент и слоты
- Затем бот сверяется со своим сценарием и в соответствии с ним формирует ответ или производит действие
- Сценарий представляет собой конечный автомат
- Конечный автомат — математическая модель дискретного устройства из теории алгоритмов

Конечный автомат

- Будем считать, что это направленный граф:
  - Каждая вершина описывает некоторое состояние
  - Ребра показывают возможные переходы между состояниями

Конечный автомат

- Будем считать, что это направленный граф:
  - Каждая вершина описывает некоторое состояние
  - Ребра показывают возможные переходы между состояниями
  - Есть правила, описывающие переходы в зависимости от состояния и входной реплики
  - Есть правила, описывающие выходной ответ или действие в зависимости от состояния и входной реплики

Пример сценарного графа

Пример сценарного графа

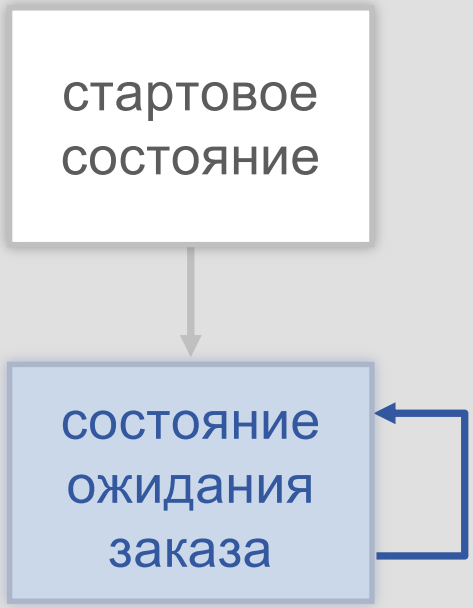
Вход:

Текст: /start

Выход:

Я бот для заказа пиццы

Пример сценарного графа



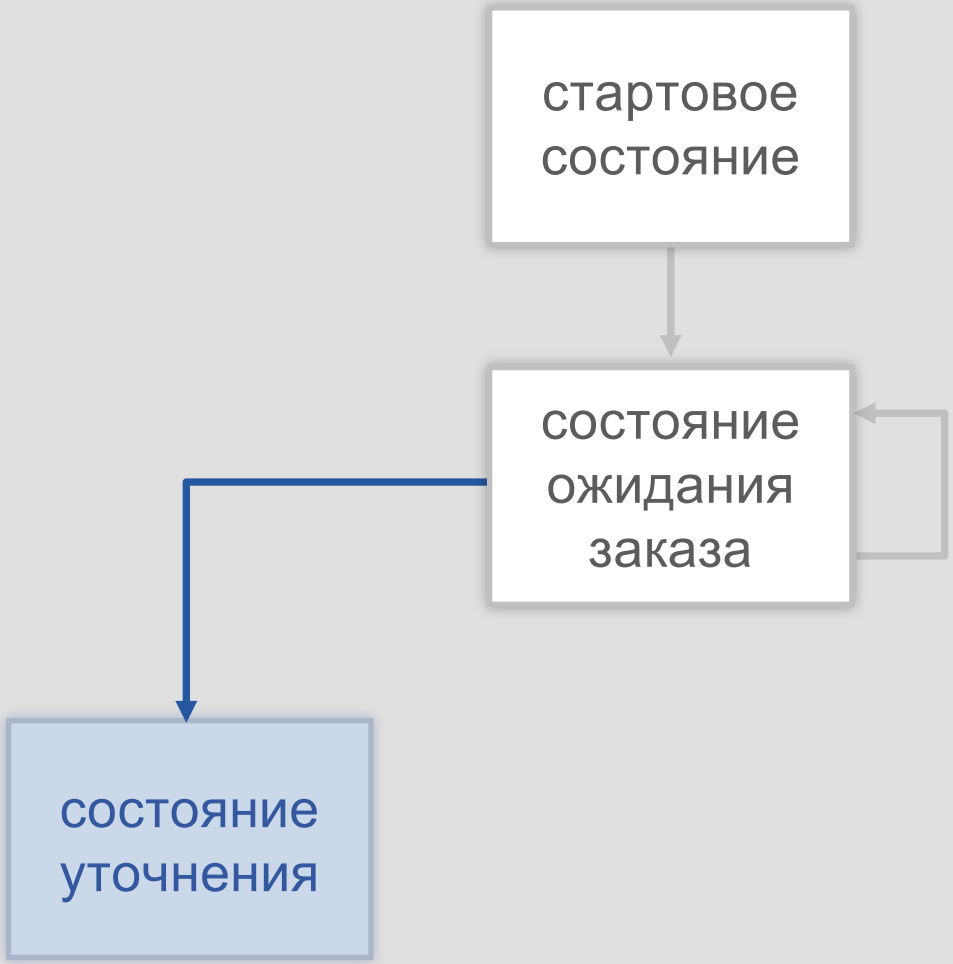
Вход:

Интент: прочее

Выход:

Не понимаю, я могу заказать пиццу

Пример сценарного графа



Вход:

Интент: заказ пиццы
Слоты: нет

Выход:

Укажите название пиццы

Пример сценарного графа

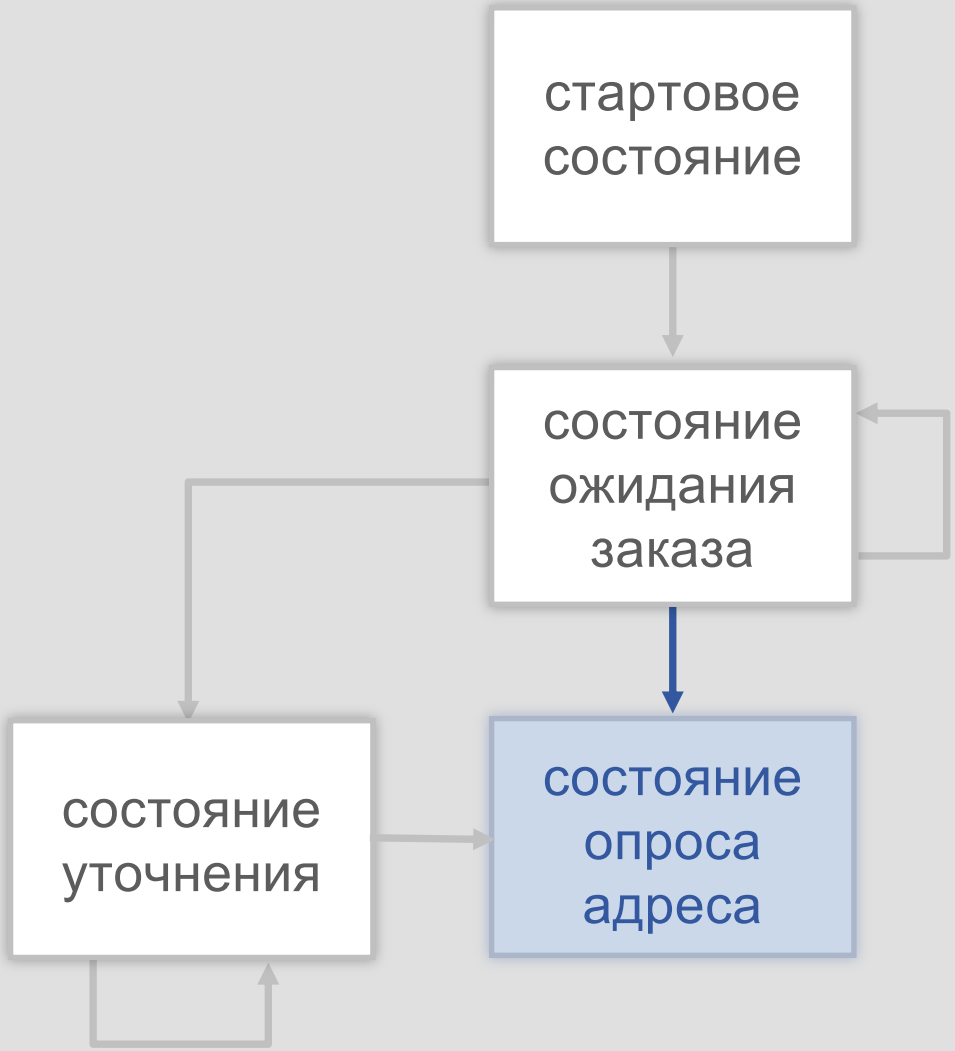
Вход:

Слоты: нет

Выход:

Укажите название пиццы

Пример сценарного графа



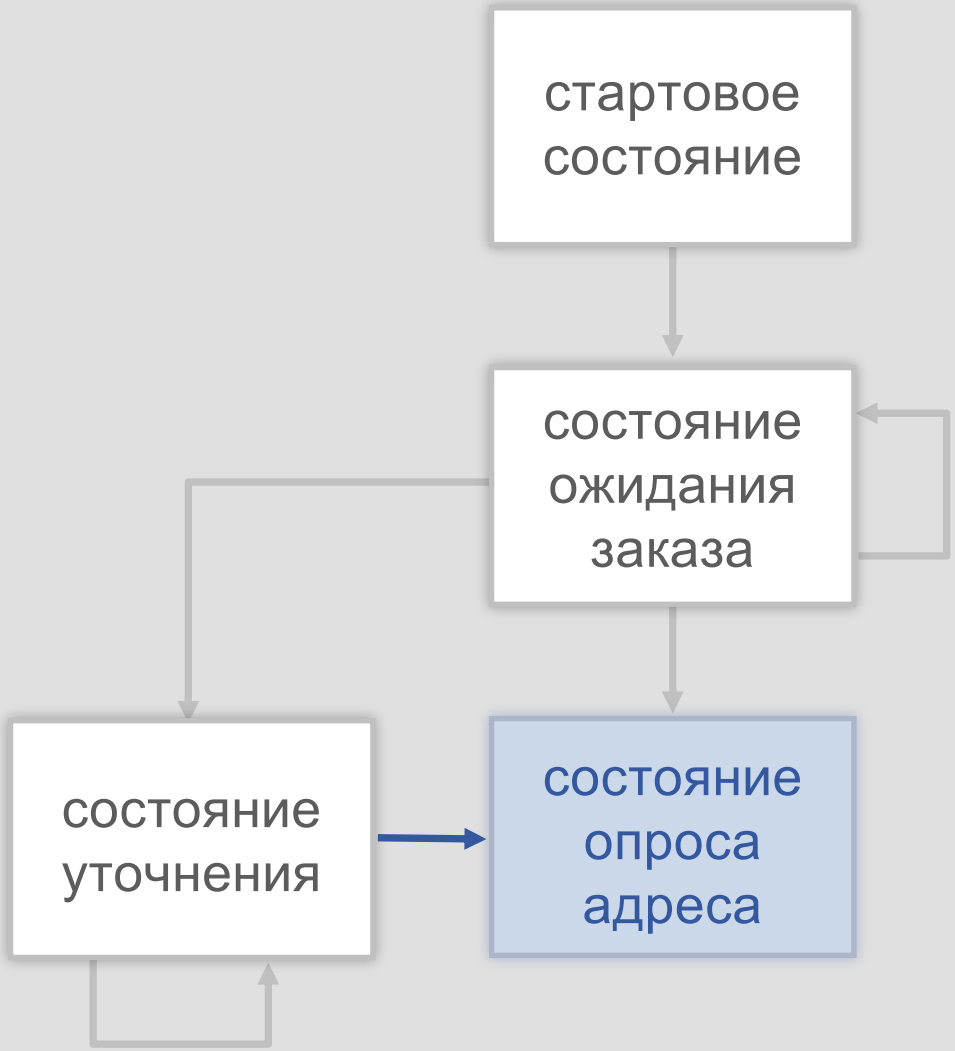
Вход:

Интент: заказ пиццы
Слоты: 4 сыра

Выход:

Заказываем пиццу 4 сыра, укажите адрес доставки

Пример сценарного графа



Вход:

Слоты: 4 сыра

Выход:

Заказываем пиццу 4 сыра, укажите адрес доставки

Пример сценарного графа

Вход:

Выход:

Система генерации ответов и действий

- Классификатор интентов и система извлечения слотов выделяют информацию, поступившую от пользователя
- Сценарный граф определяет положение пользователя в диалоге
- Всё, что осталось — адекватно отреагировать на полученную реплику

Система генерации ответов и действий

- Классификатор интентов и система извлечения слотов выделяют информацию, поступившую от пользователя
- Сценарный граф определяет положение пользователя в диалоге
- Всё, что осталось — адекватно отреагировать на полученную реплику
- Роль реагирующей системы может выполнять:
  - Набор правил формирования ответа
  - Модель ML, генерирующая/выбирающая ответ
  - Набор команд для API удалённого сервиса
  - Набор команд для API локального устройства

Основные выводы

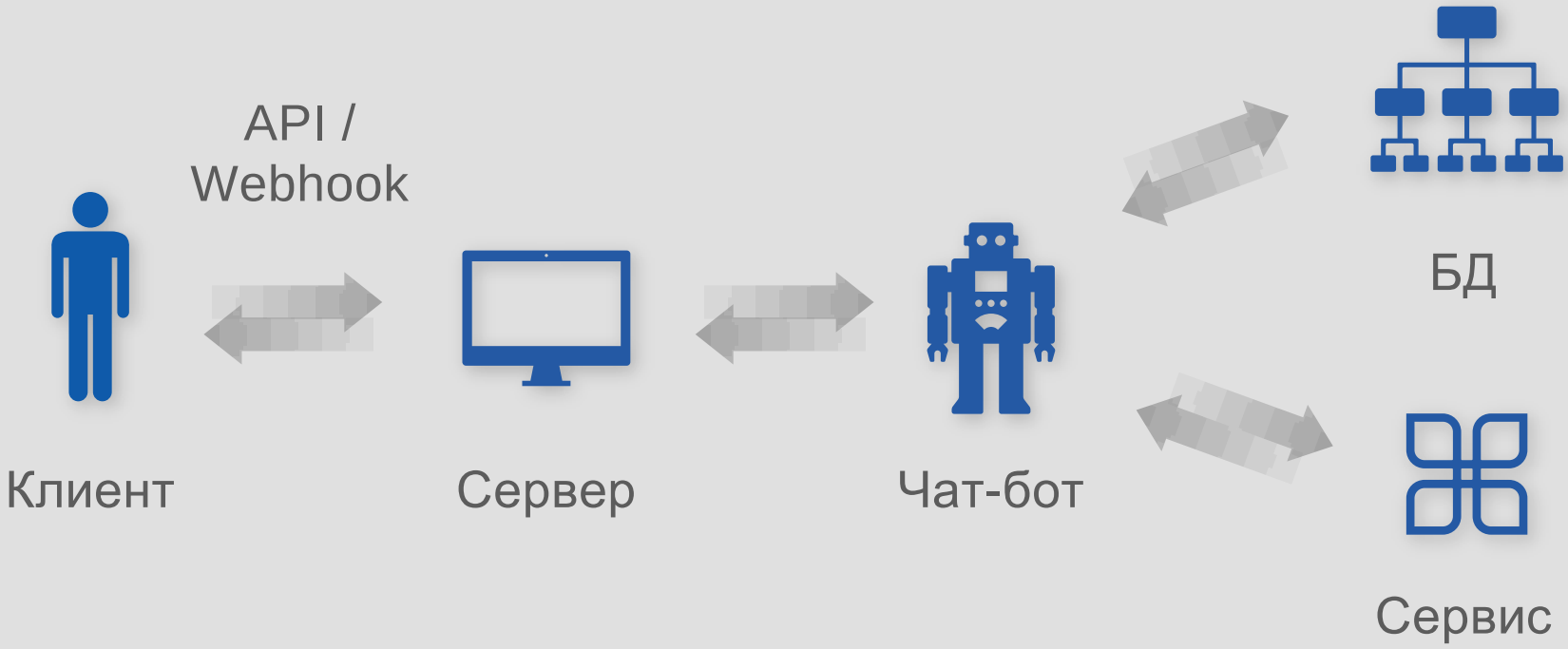
- Обычный чат-бот состоит из классификатора интентов, системы заполнения слотов, сценария и ответных фраз/действий
- При построении бота нужно уметь решать задачу NER
- Важно построить хороший сценарный граф и протестировать его на реальных пользователях

Инструменты разработки чат-ботов

Клиенты для чат-ботов

- Соцсети и мессенджеры
  - Telegram
  - VK
  - Facebook Messenger
  - …
- Виртуальные ассистенты
  - Алиса
  - Google Ассистент
  - …
- Другие приложения
- Виджеты на сайтах

Структура сервиса с чат-ботом



- Чат-бота и его инфраструктуру (или её части) можно писать с нуля
- А можно использовать платформы/фреймворки для создания и поддержки ботов

Структура сервиса с чат-ботом

- Часто бот делается для нескольких типов клиентов с разным API
- Можно писать свой код для каждого из них
- Для крупного проекта можно воспользоваться сервисом-интегратором, например, Webim
- Платформы для создания ботов обычно берут интеграции на себя, сводя задачу к настройке имеющихся возможностей

Структура сервиса с чат-ботом

- Сервер можно писать самостоятельно на удобном языке и технологическом стеке
- Платформы для создания чат-ботов предоставляют серверы и программную инфраструктуру

Структура сервиса с чат-ботом

- Самого бота (NLU, логику и интеграции с сервисом) тоже можно писать самостоятельно на удобном языке
- Платформы для создания ботов предоставляют инструменты для каждого компонента

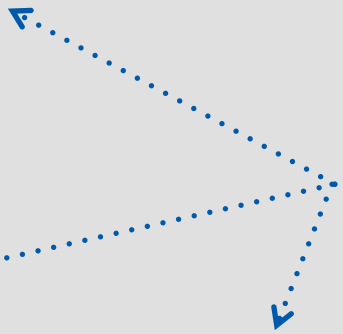
Примеры платформ для чат-ботов

- Платформы от технологических гигантов
  - Google Dialogflow
  - Amazon Lex
  - Microsoft LUIS
  - IBM Watson
  - Facebook Messenger Platform
  - Baidu KITT.AI

➢ https://habr.com/ru/company/mipt/blog/467183

Примеры платформ для чат-ботов

- Платформы от технологических гигантов
  - Google Dialogflow
  - Amazon Lex
  - Microsoft LUIS
  - IBM Watson
  - Facebook Messenger Platform
  - Baidu KITT.AI
- Локальные платформы для русского языка
  - Electra.AI
  - Chatme.AI
  - Just AI CP



Поддерживают русский язык

Платформы для создания чат-ботов

Плюсы

- Готовые инфраструктура и интеграции
- Готовые инструменты для создания бота
- Не обязательна большая NLP-экспертиза

Минусы

- Нельзя напрямую влиять на стабильность и производительность инфраструктуры
- Сложность внедрения собственных моделей
- Сложность решения любой задачи, не покрываемой встроенными инструментами
- Не всегда есть возможность локального развёртывания

Фреймворки для чат-ботов

- Промежуточным вариантом между платформой и разработкой с нуля являются фреймворки для создания чат-ботов
- В отличие от платформ, фреймворки ориентируются только на разработчиков и специалистов в ML и NLP
- Разделение условное, часто всё называют платформами

Фреймворки для чат-ботов

- Промежуточным вариантом между платформой и разработкой с нуля являются фреймворки для создания чат-ботов
- В отличие от платформ, фреймворки ориентируются только на разработчиков и специалистов в ML и NLP
- Разделение условное, часто всё называют платформами

Фреймворки для чат-ботов

- Промежуточным вариантом между платформой и разработкой с нуля являются фреймворки для создания чат-ботов
- В отличие от платформ, фреймворки ориентируются только на разработчиков и специалистов в ML и NLP
- Разделение условное, часто всё называют платформами
- Фреймворки упрощают решение части задач, в целом сохраняя гибкость процесса
- Одним из наиболее используемых в мире инструментов является Rasa
- Для русского языка также популярен DeepPavlov.AI

Что могут дать фреймфорки

- Унифицированный язык описания интентов, слотов, сценариев и ответов
- Упрощённый программный интерфейс для обучения моделей и описания правил
- Коллекции предобученных нейросетевых моделей для типовых задач
- Базовые серверы для тестирования бота
- Инструменты аналитики для оценки работы бота

Что могут дать фреймфорки

- Унифицированный язык описания интентов, слотов, сценариев и ответов
- Упрощённый программный интерфейс для обучения моделей и описания правил
- Коллекции предобученных нейросетевых моделей для типовых задач
- Базовые серверы для тестирования бота
- Инструменты аналитики для оценки работы бота

Чем дороже, сложнее и долгосрочнее проект бота, тем рациональнее создавать собственные решения для его компонентов

Метрики качества чат-бота

- NLU-компоненты бота:
  - Качество классификатора интентов
  - Качество выявления сущностей
  - …

Метрики качества чат-бота

- Бизнес-метрики:
  - Возвращаемость пользователей *(Retention)*
  - Доля успешных диалогов
  - Доля обработанных ботом обращений от их общего числа
  - Доля реплик, которые бот не смог разобрать
  - Средняя длина диалога
  - Отзывы и тональность реплик пользователей
  - Количество заработанных ботом денег
  - …

Основные выводы

- Запуск чат-бота требует не только описание его логики, но и сервисную инфраструктуру
- Если бот рассчитан на широкую аудиторию, то инфраструктура должна быть масштабируемой
- Это могут обеспечить платформы для разработки ботов
- Платформы также снижают порог входа для разработчиков логики бота

Основные выводы

- Обратной стороной использования платформ является сложность модификации всех компонентов системы под свою задачу
- Помимо полноценных платформ есть открытые фреймворки, упрощающие описание логики бота даже для профессиональных разработчиков
- Качество чат-ботов можно и нужно измерять, есть множество технических и бизнесовых метрик

Виртуальные ассистенты, диалоговый ИИ

Виртуальный ассистент

- Виртуальный ассистент (intelligent virtual assistant) — программная система, выполняющая задачи пользователя на основе его обращений, текстовых или голосовых (реже графических)

Виртуальный ассистент

- Виртуальный ассистент (intelligent virtual assistant) — программная система, выполняющая задачи пользователя на основе его обращений, текстовых или голосовых (реже графических)
- Большинство крупных IT-компаний имеют собственных ассистентов, интегрированных со своей экосистемой:

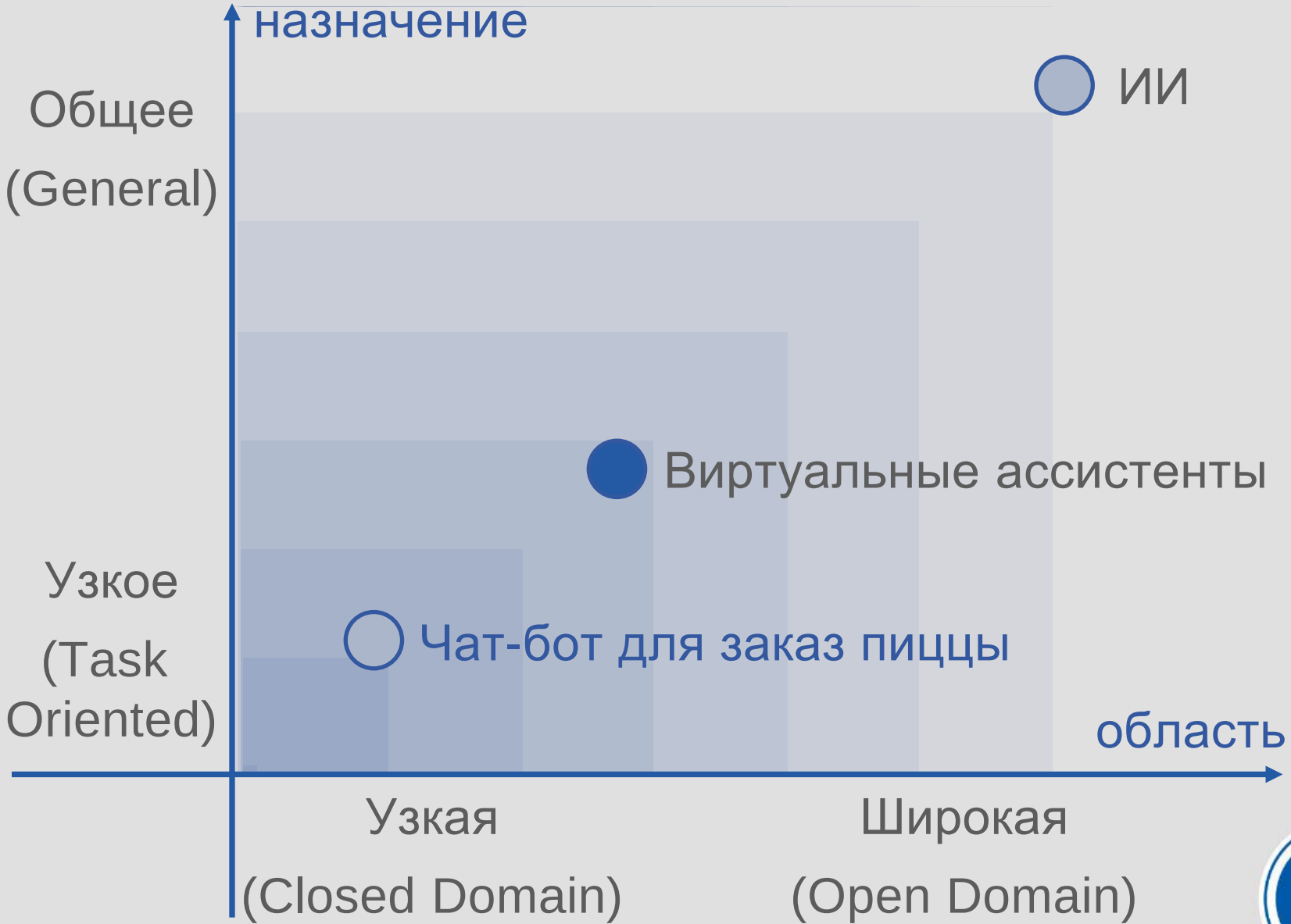
- Google Assistant (Google)
- Alexa (Amazon)
- Siri (Apple)
- Cortana (Microsoft)
- Bixby (Samsung)
- Алиса (Яндекс)
- Маруся (Mail.ru)

И ещё десятки других!

Что представляет собой ассистент

- Виртуальный ассистент и чат-бот — близкие понятия
- По сути, ассистент — это мощный чат-бот с голосовым интерфейсом, решающий определённый спектр задач
- При его построении используются те же сценарии, классификаторы интентов и NER
- В отличие от простых ботов, в ассистентах эти задачи решаются почти всегда с помощью глубоких нейросетей
- Некоторые ассистенты умеют работать в режиме свободного диалога

Виды диалоговых систем



Свободный диалог

- Ассистент умеет выполнять ряд задач, например:
  - устанавливать будильники и напоминания
  - искать и воспроизводить медиа
  - сообщать погоду, курсы валют, дорожную ситуацию
  - редактировать контакты
  - отправлять сообщения
  - выполнять общие поисковые запросы

Свободный диалог

- Тип задачи определяется интентом пользовательской реплики и контекстом
- Если явного интента в пользовательской реплике нет, то ассистент может поддержать свободное общение без конкретной цели

Свободный диалог

- Тип задачи определяется интентом пользовательской реплики и контекстом
- Если явного интента в пользовательской реплике нет, то ассистент может поддержать свободное общение без конкретной цели
- Система всё равно остаётся Retrieval-Based — ответы набираются из большой базы, которая регулярно пополняется
- На русском языке это хорошо получается у Алисы и Google ассистента

Свободный диалог

Сколько тебе лет?

Давайте я сделаю вид, что не слышала этого вопроса. А вы сделаете вид, что не задавали его.

Свободный диалог

Ты человек?

Мои разработчики говорят, что я программа

Но я не знаю, насколько они компетентны, чтобы делать такие громкие заявления 😎

Свободный диалог

Польза

- Создаёт ощущение целостного диалога
- Формирует образ личности ассистента — важно!
- Развлекает пользователя, повышает возвращаемость

Сложности

- Поведение малопредсказуемо
- Ответы могут быть токсичными в разных ситуациях — нужно аккуратно обрабатывать!
- Тяжело собирать и размечать диалоги
- Сложно и долго настраивать модели и фильтры

Принципы построения умных ботов

- Собирается большой набор реальных диалогов
- Обучается нейросеть:
  - Вход: пара «контекст-ответ»
  - Перевод входа в векторные представления
  - Выход: оценка Score релевантности ответа
- Контекст состоит из одной или нескольких реплик

Принципы построения умных ботов

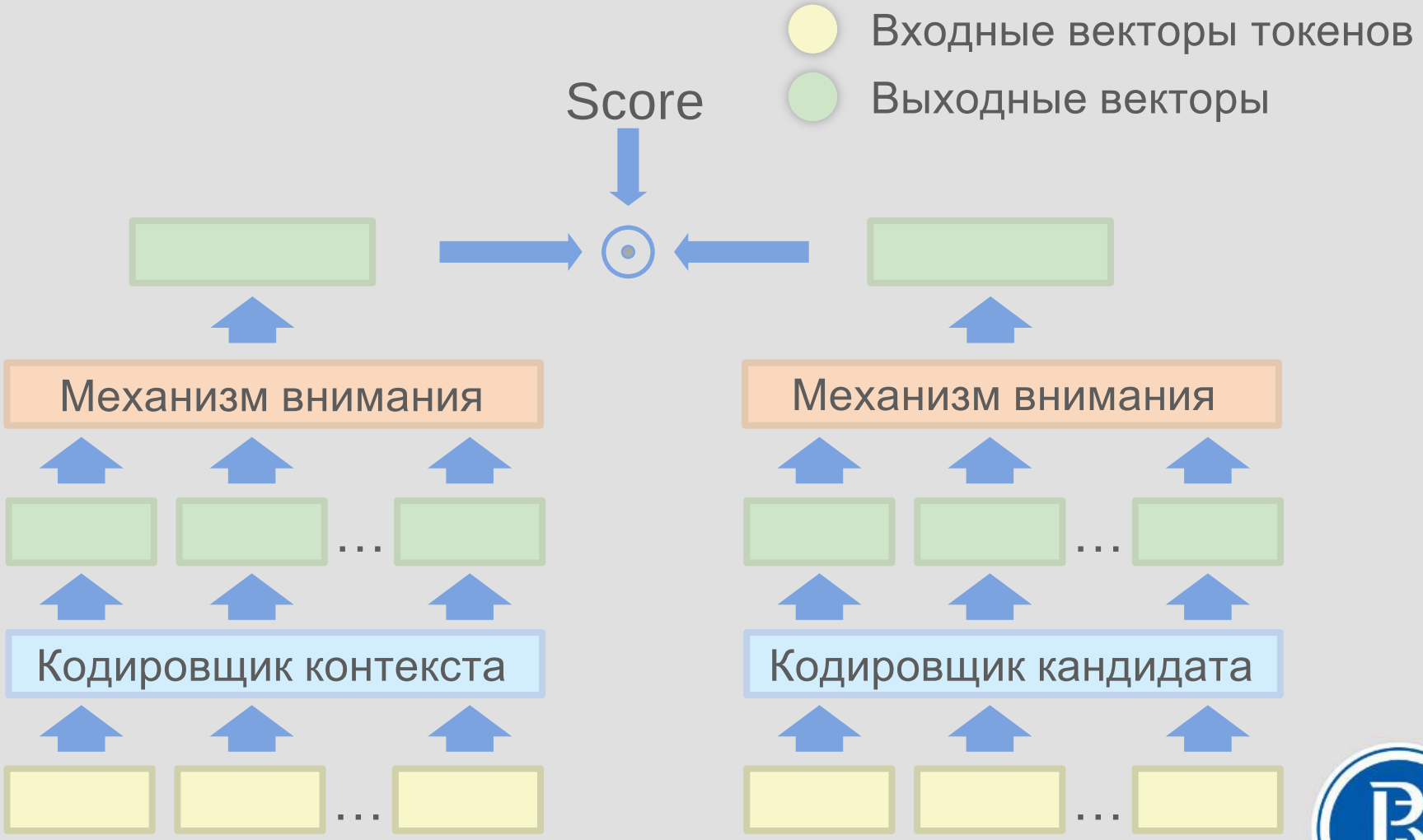
- Собирается большой набор реальных диалогов
- Обучается нейросеть:
  - Вход: пара «контекст-ответ»
  - Перевод входа в векторные представления
  - Выход: оценка Score релевантности ответа
- Контекст состоит из одной или нескольких реплик
- На этапе применения модель для входного контекста оценивает релевантность реплик-кандидатов
- Из наиболее вероятных ответов выбирается самый уместный с точки зрения иных ограничений

Архитектуры кодировщиков

- Параллельный кодировщик *(Bi-encoder)*
- Контекст и ответ переводятся в векторы своими сетями-кодировщиками
- В качестве Score можно брать сигмоиду от скалярного произведения выходных векторов
- Кодировщики могут использовать общие веса

Архитектуры кодировщиков

- Параллельный кодировщик *(Bi-encoder)*

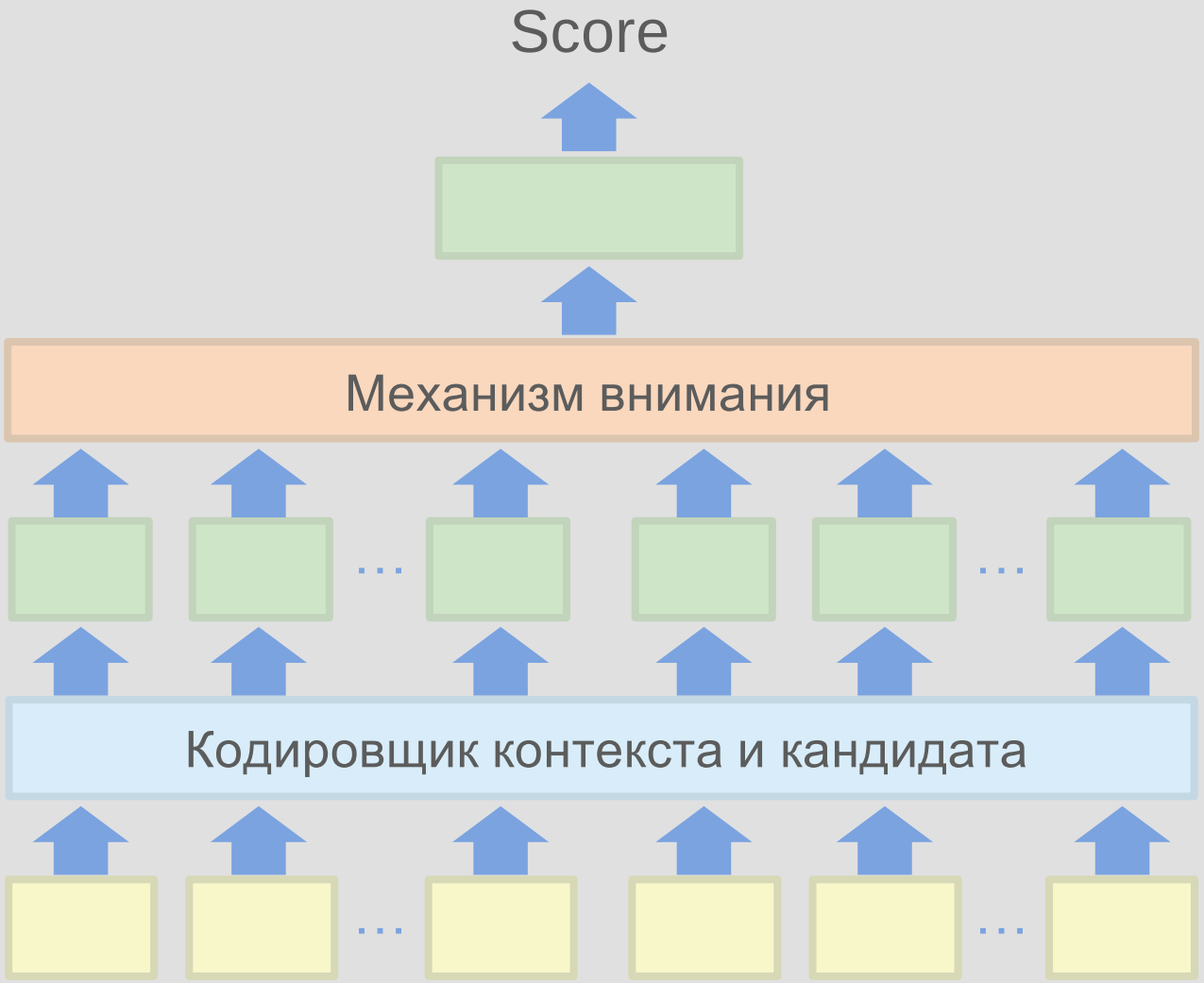


Архитектуры кодировщиков

- Кросс-кодировщик *(Cross-encoder)*
- Контекст и реплика-кандидат кодируются вместе одной сетью-кодировщиком
- В качестве Score можно взять выход лог-регрессии, применяемой к выходному вектору

Архитектуры кодировщиков

- Кросс-кодировщик (Cross-encoder)



Архитектуры кодировщиков

- В основе кодировщиков могут лежать различные архитектуры нейронных сетей
- Лучше всего работают сети на основе Трансформера, например, BERT и его модификаций

Архитектуры кодировщиков

- В основе кодировщиков могут лежать различные архитектуры нейронных сетей
- Лучше всего работают сети на основе Трансформера, например, BERT и его модификаций
- Параллельное кодирование работает намного быстрее:
  - подготавливаем векторы всех ответов заранее
  - для выбора ответа достаточно закодировать контекст
  - дальше используются масштабируемые библиотеки для поиска ближайших соседей (например, FAISS)

Архитектуры кодировщиков

- В основе кодировщиков могут лежать различные архитектуры нейронных сетей
- Лучше всего работают сети на основе Трансформера, например, BERT и его модификаций
- Параллельное кодирование работает намного быстрее:
  - подготавливаем векторы всех ответов заранее
  - для выбора ответа достаточно закодировать контекст
  - дальше используются масштабируемые библиотеки для поиска ближайших соседей (например, FAISS)
- Но кросс-кодирование работает качественнее:
  - представления контекста и кандидата влияют друг на друга при выводе итогового вектора

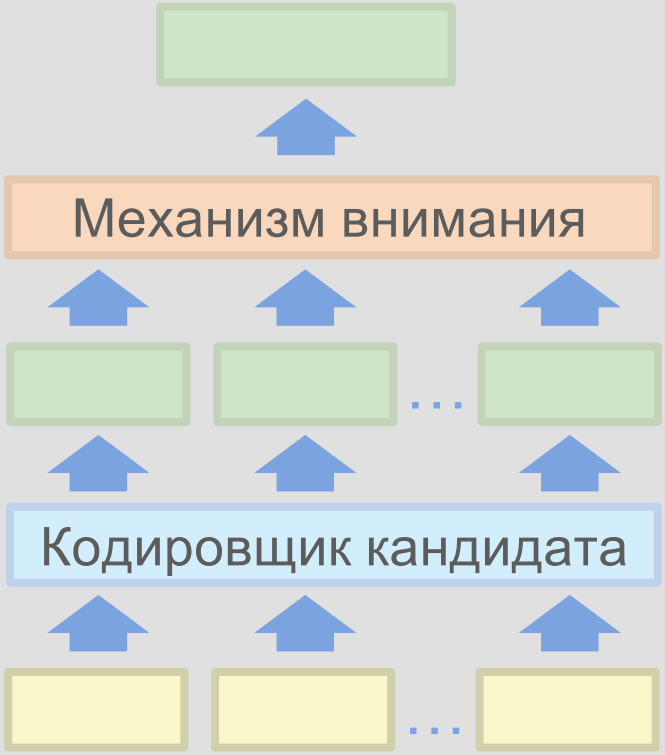
Архитектуры кодировщиков

- Поликодировщик (Poly-encoder) — попытка объединить преимущества двух подходов

➢ Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, Jason Weston. Poly-encoders: architectures and pre-training strategies for fast and accurate multi-sentence scoring // ICLR. — *2020.*

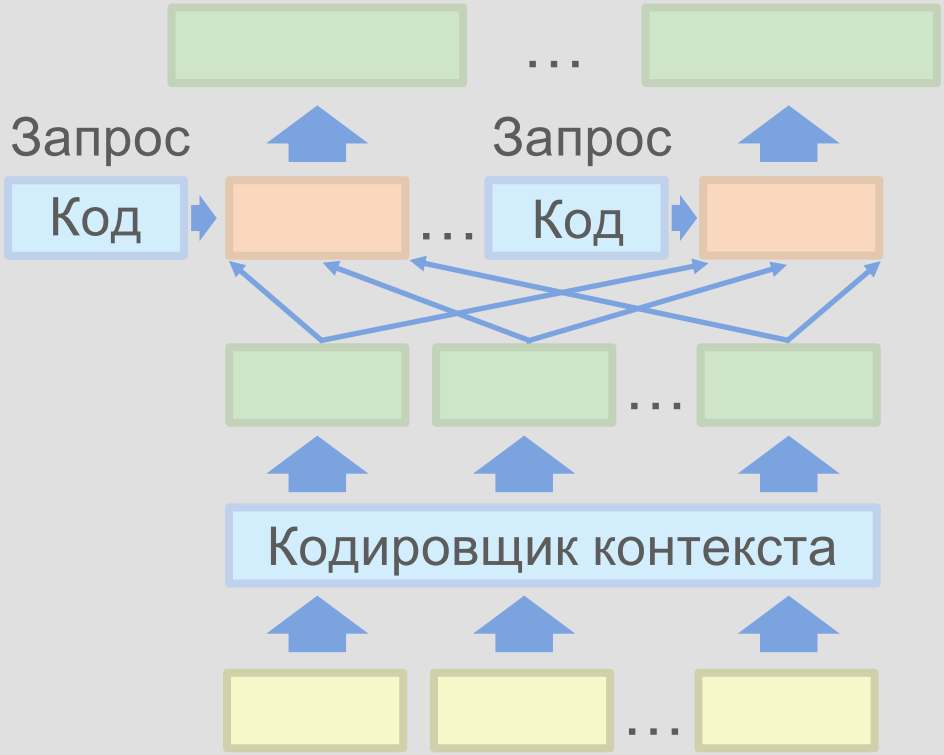
Poly-encoder

- Использует две сети на основе Трансформера
- Векторы реплик-кандидатов подсчитываются заранее



Poly-encoder

- Контекст промежуточно кодируется m векторами
- Получаются они с помощью контекстных кодов



Poly-encoder

- Контекст промежуточно кодируется m векторами
- Получаются они с помощью m векторов контекстных кодов c_i — запросов для механизма внимания
- Пусть кодировщик для каждого из N входных токенов выдал соответствующие векторы h_j
- Формула для i-го вектора контекста, $i = 1, \dots m$:

$$y^i = \sum_j w_j^{c_i} h_j$$

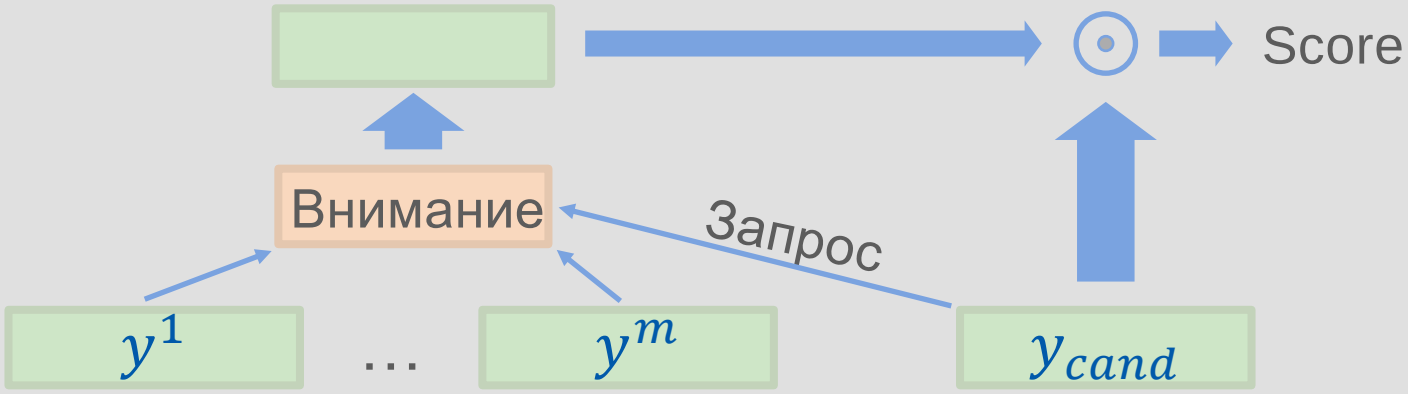
- Веса $w_j^{c_i}$ определяются формулой

$$\left(w_1^{c_i}, \dots, w_N^{c_i}\right) = \text{Softmax}(\langle c_i, h_1 \rangle, \dots, \langle c_i, h_N \rangle)$$

Poly-encoder

- Окончательно контекст кодируется одним вектором y, тоже с помощью механизма внимания
- На входе промежуточные векторы y^i ,вектором запроса является представление кандидата y_{cand}
- Формулы выглядят аналогично:

$$y = \sum_i w_i \, y^i, \qquad (w_1, \dots, w_m) = \text{Softmax}(\langle y_{cand}, y^1 \rangle, \dots, \langle y_{cand}, y^m \rangle)$$



Poly-encoder

- В результате вектор контекста формируется с учётом представления кандидата, что улучшает качество
- Для каждого кандидата вместо вывода всего вектора надо посчитать $m < N$ скалярных произведений
- Это позволяет не сильно замедлить работу модели

Poly-encoder

- Оба кодировщика сперва предобучаются попеременно на задачах предсказания
  - маскированных токенов
  - следующего предложения
  - следующего высказывания (набор предложений)
- Затем модели дообучаются на целевую задачу ведения диалога с нужным набором данных

Данные для ообучения ассистентов

- Качество данных в случае ассистентов является решающим фактором
- Как и в других случаях, сбор их можно производить с помощью асессоров или краудсорсинга
- Есть два типа задач:
  - Генеративные — нужно создать целевые реплики
  - Дискриминативные — нужно разметить реплики (на интенты, сущности, тональность и т.д.)

Данные для ообучения ассистентов

- При обучении модели для свободного диалога положительные примеры набираются из реплик, идущих последовательно в диалогах
- Отрицательные примеры обычно берутся случайно
- Для хороших результатов можно специально подбирать отрицательные случаи, на которых сеть сильно ошибается

Данные для ообучения ассистентов

- При обучении модели для свободного диалога положительные примеры набираются из реплик, идущих последовательно в диалогах
- Отрицательные примеры обычно берутся случайно
- Для хороших результатов можно специально подбирать отрицательные случаи, на которых сеть сильно ошибается
- В кандидаты можно добавлять не все имеющиеся реплики, а только удовлетворяющие определённым условиям (например, без ругательств)
- При этом в обучающей выборке «плохие» реплики быть должны, иначе сеть будет реагировать на них неадекватно

Метрики качества

- Как и для обычных ботов, всегда оцениваются классификатор интентов и извлечение сущностей
- Данные для оценивания можно размечать самостоятельно с помощью краудсорсинга
- Существуют готовые наборы, например, NLU Benchmark от SNIPS
- Для свободных диалогов оценивается соответствие ответной реплики входному контексту
- Ещё оцениваются дополнительные метрики, такие как грубость, токсичность, фамильярность и т.п.

➢ A. Coucke, A. Saade, A. Ball et al. Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces // ArXiv preprint. — *2018.*

Расширение возможностей ассистентов

- Некоторые ассистенты позволяют сторонним разработчикам создавать расширения на своей основе
- Такие расширения называются навыками (skills, actions)
- Разработчик сам описывает логику своего бота
- Это может быть платформенный или самописный бот
- Он может быть простым, а может поддерживать свободный диалог

Расширение возможностей ассистентов

- Бот подключается к ассистенту и может быть вызван из него с помощью ключевых фраз
- Разработчику не нужно беспокоиться о преобразованиях голоса в текст и обратно
- Также в случае устройств с экраном он без дополнительных усилий получает графический интерфейс ассистента

Основные выводы

- Виртуальный ассистент — многофункциональный чат-бот с голосовым интерфейсом
- Иногда может поддерживать свободный диалог
- Системы свободного диалога строятся на подборе наиболее подходящей из имеющихся в базе реплик в зависимости от контекста диалога
- Для этого используются векторные представления реплик и контекста, генерируемые кодировщиками
- В роли кодировщиков выступают разные нейросети, хорошо работаю модели на основе Трансформера
- Качество очень сильно зависит от правильного сбора и использования в обучении данных

Вопросно-ответные системы, задача SQuAD

QA-система

- Вопросно-ответная система (QA-system) — подвид диалоговой системы, предназначенный для ответов на вопросы
- Задача QA находится на стыке информационного поиска и анализа текстов
- QA-система должна:
  - Понимать запрос пользователя на естественном языке
  - Искать на него ответ в огромных структурированных или сырых массивах данных
  - Формировать ответ на понятном человеку языке

QA-система

- Вопросно-ответная система является важным компонентом поисковой системы и использующих её виртуальных ассистентов общего назначения
- Построение хорошей QA-системы — сложная задача, которую пытаются решать с помощью глубоких нейросетевых моделей

Типы вопросов для QA-систем

- Фактологические вопросы (кто? где? когда?)

Кто написал “Старик и Море”?

Где находится Момбаса?

Когда открыли Америку?

Типы вопросов для QA-систем

- Фактологические вопросы (кто? где? когда?)

Кто написал “Старик и Море”?

Эрнест Хемингуэй

Где находится Момбаса?

Кения

Когда открыли Америку?

1492

Типы вопросов для QA-систем

- Вопросы на понимание здравого смысла

Где можно плавать?

а) Бассейн

б) Море

в) Небоскрёб

Типы вопросов для QA-систем

- Сравнительные и оценочные вопросы

Какой город больше,
Москва или Берлин?

Москва

Какой из континентов
меньше всех по площади?

Австралия

Типы вопросов для QA-систем

- Вопросы по прочитанному тексту *(machine reading comprehension)*

Пес побежал за котом, тот запрыгнул на забор.

Кто запрыгнул на забор?

Типы вопросов для QA-систем

- Открытые вопросы *(open domain QA)* — вопросы о произвольных вещах в свободной форме, требуют доступа к огромным массивам информации

Подходы к построению QA-систем

- Информационный поиск (IR-Based QA):
  - Есть текстовая коллекция, разбитая на фрагменты
  - Система ищет фрагмент, который содержит ответ на пользовательский вопрос

Подходы к построению QA-систем

- Информационный поиск (IR-Based QA):
  - Есть текстовая коллекция, разбитая на фрагменты
  - Система ищет фрагмент, который содержит ответ на пользовательский вопрос
- Поиск по базе знаний (KB-Based QA):
  - Есть база знаний в структурированном формате
  - Система формирует на основании вопроса запрос и отправляет его в базу:

Когда умер Владимир Ленин?

дата-смерти(Владимир Ленин, ?)

  - Сложный и не очень популярный подход

Работа IR-based QA-системы

- Обработка текста вопроса
  - Определение типа вопроса
  - Выделение ключевых слов и сущностей
  - Определение типа ответа
- Формирование запроса к данным
  - Определение типа вопроса
  - Выделение ключевых слов и сущностей
  - Определение типа ответа
- Извлечение документов и фрагментов из них
- Извлечение ответа из фрагмента и его обработка

Набор данных SQuAD

- Канонический набор данных для IR-based QA-систем — SQuAD *(Stanford Question Answering Dataset)*
- Состоит из 100 тысяч троек:
  - Текст вопроса
  - Текстовый фрагмент, содержащий ответ
  - Текст ответа
- В SQuAD 2.0 добавлены 50 тысяч вопросов с релевантными фрагментами, в которых ответа нет

➢ Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text // EMNLP. — *2016.*

Набор данных SQuAD

- Формирование SQuAD:
  - Выбраны топ-10 тысяч статей английской Википедии по значению PageRank
  - Из них случайно набраны 536 статей
  - Каждая статья разбита на параграфы
  - Краудсорсинг: к каждому параграфу нужно сформулировать до 5 вопросов
  - Вопросы формулируются самостоятельно, без копирования текста из параграфа
  - Используются формальные фильтры (например, ответ должен быть не слишком коротким)

Набор данных SQuAD

- Аналоги SQuAD для других языков:
  - SberQuAD (для русского языка)
  - KorQuAD
  - FQuAD
  - Neural Arabic QA
  - Spanish SQuAD
  - Italian SQuAD
  - XQuAD (смесь из 11 языков)
- Получаются как разметкой по схожим принципам, так и переводом оригинального набора

➢ https://medium.com/deepset-ai/going-beyond-squad-part-1-question-answering-in-different-languages-8eac6cf56f21

Решение задачи SQuAD

- Этап 1: отбор кандидатов для ранжирования
  - Отбор фрагментов, которые потенциально могут содержать ответ на вопрос
  - Обычно используются легковесные грубые методы
  - Например, косинусное расстояние между векторами TF-IDF или FastText вопроса и параграфов-кандидатов

Решение задачи SQuAD

Вопрос: Почему музыкант Moby взял такой псевдоним?

“Родился в Нью-Йорке, но вырос в Дэриене, Коннектикут. Когда ему было два года, его отец погиб в автокатастрофе, и в дальнейшем мальчик воспитывался матерью, довольно прогрессивной женщиной, поощрявшей творческие наклонности сына. В девять лет он начал брать уроки игры на фортепиано и гитаре. Прозвище «Моби» Ричард получил потому, что дальним родственником ему приходился Герман Мелвилл, автор «Моби Дика». Подростком он играл на гитаре в различных командах, начиная с панк-группы The Vatican Commandos и заканчивая анархическим коллективом Flipper и с достаточно известной в своё время командой Ultra Vivid Scene.”

Решение задачи SQuAD

Вопрос: Почему музыкант Moby взял такой псевдоним?

“Родился в Нью-Йорке, но вырос в Дэриене, Коннектикут. Когда ему было два года, его отец погиб в автокатастрофе, и в дальнейшем мальчик воспитывался матерью, довольно прогрессивной женщиной, поощрявшей творческие наклонности сына. В девять лет он начал брать уроки игры на фортепиано и гитаре. Прозвище «Моби» Ричард получил потому, что дальним родственником ему приходился Герман Мелвилл, автор «Моби Дика». Подростком он играл на гитаре в различных командах, начиная с панк-группы The Vatican Commandos и заканчивая анархическим коллективом Flipper и с достаточно известной в своё время командой Ultra Vivid Scene.”

Решение задачи SQuAD

- Этап 1: отбор кандидатов для ранжирования
- Этап 2: поиск подстрок фрагмента с ответом
  - Классификация слов в параграфах-кандидатах: каждое слово проверяется на то, является ли оно какой-то частью ответа
  - Используются тяжеловесные нейросетевые модели

Решение задачи SQuAD

Вопрос: Почему музыкант Moby взял такой псевдоним?

“Родился в Нью-Йорке, но вырос в Дэриене, Коннектикут. Когда ему было два года, его отец погиб в автокатастрофе, и в дальнейшем мальчик воспитывался матерью, довольно прогрессивной женщиной, поощрявшей творческие наклонности сына. В девять лет он начал брать уроки игры на фортепиано и гитаре. Прозвище «Моби» Ричард получил потому, что дальним родственником ему приходился Герман Мелвилл, автор «Моби Дика». Подростком он играл на гитаре в различных командах, начиная с панк-группы The Vatican Commandos и заканчивая анархическим коллективом Flipper и с достаточно известной в своё время командой Ultra Vivid Scene.”

Решение задачи SQuAD

Вопрос: Почему музыкант Moby взял такой псевдоним?

“Родился в Нью-Йорке, но вырос в Дэриене, Коннектикут. Когда ему было два года, его отец погиб в автокатастрофе, и в дальнейшем мальчик воспитывался матерью, довольно прогрессивной женщиной, поощрявшей творческие наклонности сына. В девять лет он начал брать уроки игры на фортепиано и гитаре. Прозвище «Моби» Ричард получил потому, что дальним родственником ему приходился Герман Мелвилл, автор «Моби Дика». Подростком он играл на гитаре в различных командах, начиная с панк-группы The Vatican Commandos и заканчивая анархическим коллективом Flipper и с достаточно известной в своё время командой Ultra Vivid Scene.”

Модель DrQA

- Решаем задачу классификации на этапе 2
- Одна из первых моделей — DrQA
  - Кодируем запрос и каждый из параграфов-кандидатов параллельно с помощью BiLSTM
  - Сети для запроса и параграфов разные
  - На вход подаются векторы GloVe токенов
  - Для токенов параграфа в вектор добавляются грамматические признаки

➢ Danqi Chen, Adam Fisch, Jason Weston, Antoine Bordes. Reading Wikipedia to Answer Open-Domain Questions. // ACL. — *2017.*

Модель DrQA

- Решаем задачу классификации на этапе 2
- Одна из первых моделей — DrQA
  - Выходы сети для запроса суммируются в один вектор q
  - На выходе сети для параграфа набор векторов p_i, соответствующих входным токенам

➢ Danqi Chen, Adam Fisch, Jason Weston, Antoine Bordes. Reading Wikipedia to Answer Open-Domain Questions. // ACL. — *2017.*

Модель DrQA

- Решаем задачу классификации на этапе 2
- Одна из первых моделей — DrQA
  - Выходы сети для запроса суммируются в один вектор q
  - На выходе сети для параграфа набор векторов p_i, соответствующих входным токенам
  - Обучаем две весовые матрицы W_s и W_e для определения начала и конца подстроки-ответа
  - Для каждого i -го токена определяем вероятность быть началом и концом ответа по формулам

$$P_{start} \propto \exp(p_i W_s q), \qquad P_{end} \propto \exp(p_i W_e q)$$

  - Выбираем фрагмент $[i_s, i_e]$ ограниченной длины с максимальным значением $P_{start}(i_s) \times P_{end}(i_e)$

Аналогичные модели

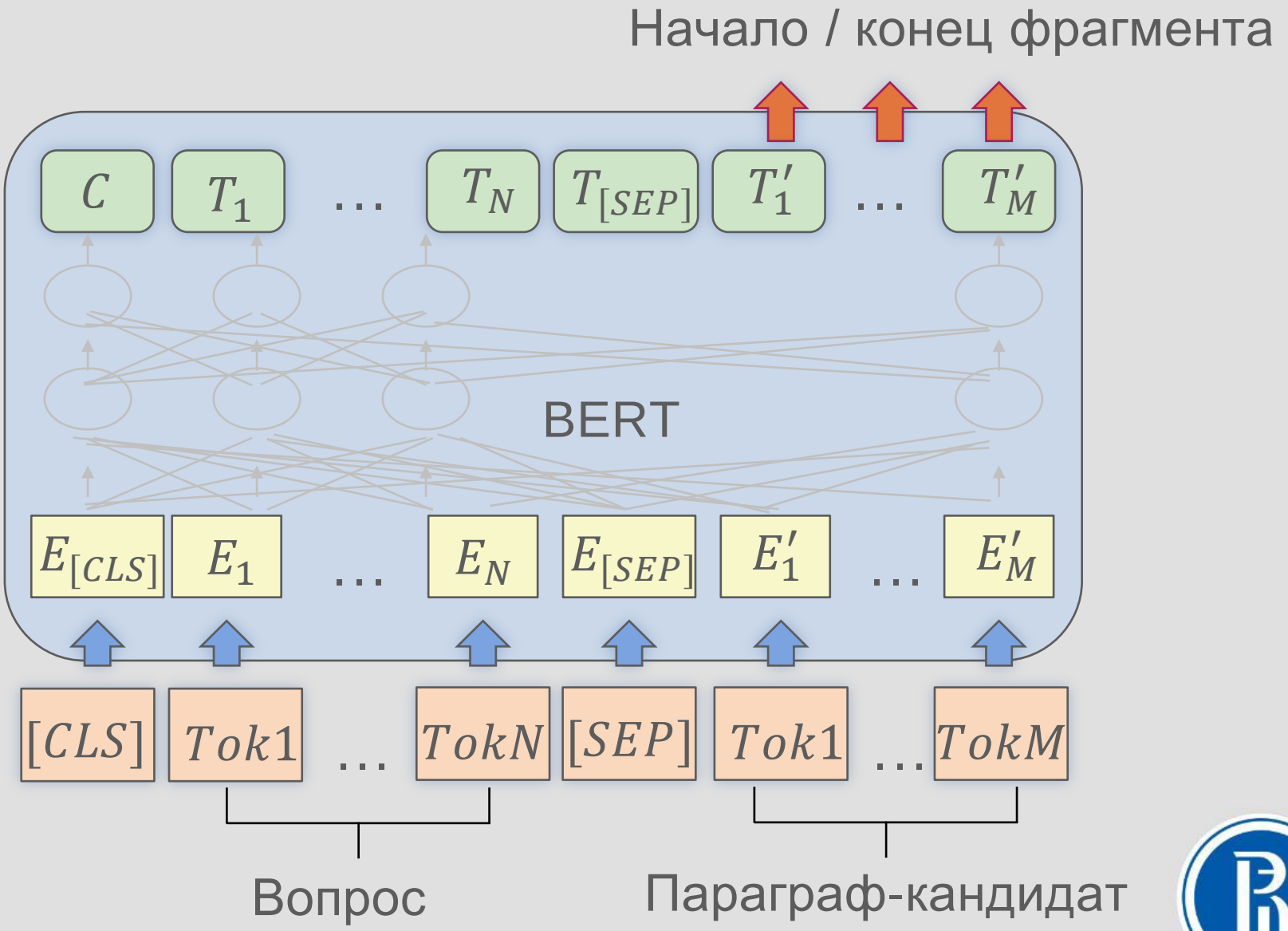
- После DrQA появилось ещё несколько моделей, работающих по такому же принципу
- В основе остаётся BiLSTM, но добавляются
  - дополнительные рекурентные слои
  - механизмы внимания
  - свёрточные слои для символов
  - …
- Примеры моделей
  - BiDAF
  - R-NET
  - QA-NET

BERT для SQuAD

- Предобученные нейросетевые языковые модели хорошо определяют связи между предложениями
- Поэтому SQuAD, как задача поиска ответа в абзаце, хорошо ложится на BERT без модификаций

➢ Sam Schwager, John Solitario. SQuAD: Question and Answering on SQuAD 2.0: BERT Is All You Need

BERT для SQuAD



Подойдут и другие модели

- Изученный ранее зоопарк предобученных моделей тоже можно использовать для задачи SQuAD
- Хорошо себя показывают ансамбли на основе модели ALBERT
- Есть стандартный лидерборд для SQuAD 2.0, на котором показываются рекорды, поставленные различными моделями

➢ https://rajpurkar.github.io/SQuAD-explorer

Основные выводы

- Вопросно-ответные системы — ещё одно направление в NLP, идеалом которого является ИИ
- Существующие системы могут классифицироваться по типу обрабатываемых запросов и по методу построения
- Существуют два основных подхода к построению — IR-based и KB-based, в основном развивается первый
- Самая популярная задача для IR-based подходов — SQuAD 2.0 с одноимённым набором данных
- Задача решается с помощью нейросетевых методов на основе рекурентных сетей или предобученных моделей на основе Трансформера