Математические методы анализа текстов

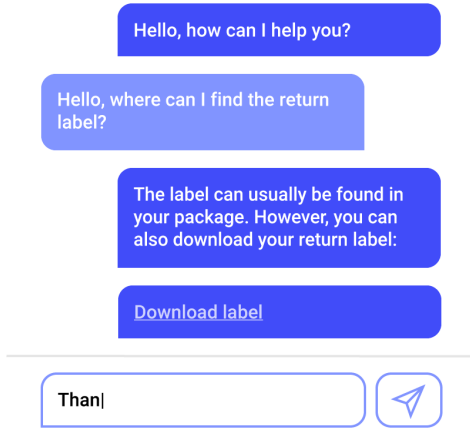
# Диалоговые и вопросно-ответные системы

Мурат Апишев (mel-lain@yandex.ru)

Ноябрь, 2020

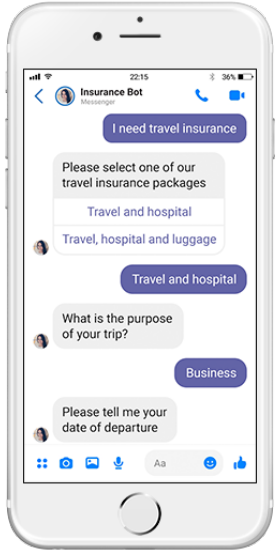
# Диалоговая система

- Диалоговая система — компьютерная система, которая общается с человеком привычным для него образом
- Интерфейсы общения могут быть различные: текст, голос, жесты
- Наиболее интересен текст — другие форматы часто сводятся к нему
- Большинство текстовых диалоговых систем строятся на основе чат-ботов

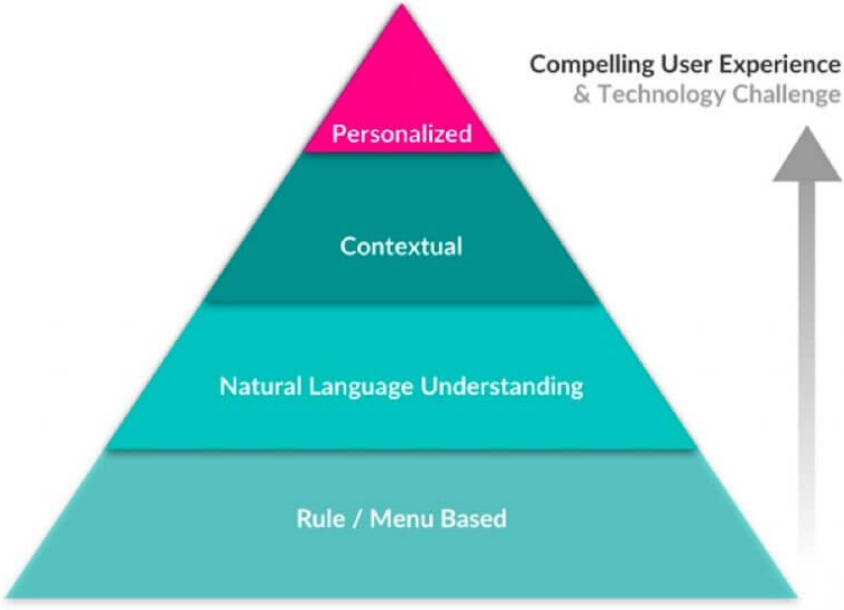


# Сферы применения чат-ботов

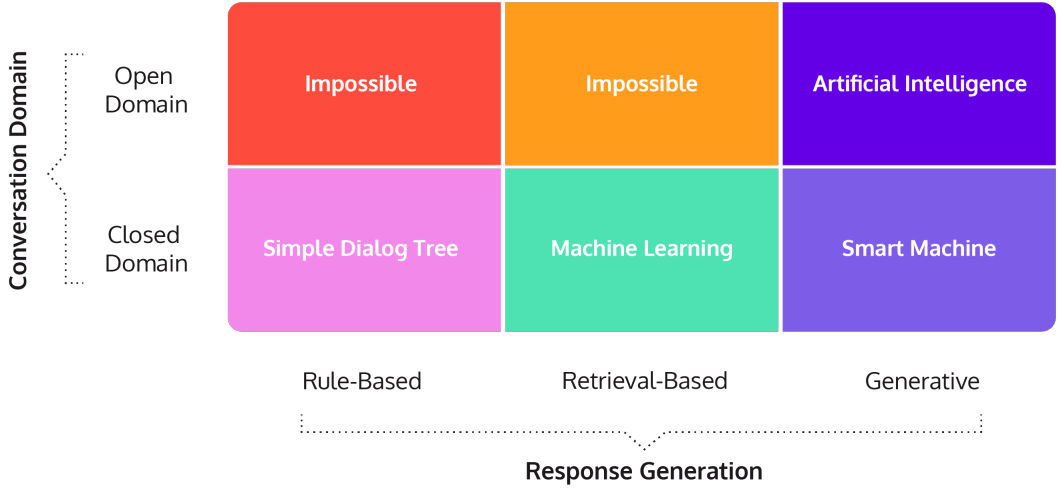
- Замена операторов контакт-центров
- Замена консультантов при выборе товаров
- Автоматизация заказа такси, еды, билетов и услуг
- Автоматизация онбординга
- Виртуальные ассистенты



# Классификации диалоговых систем



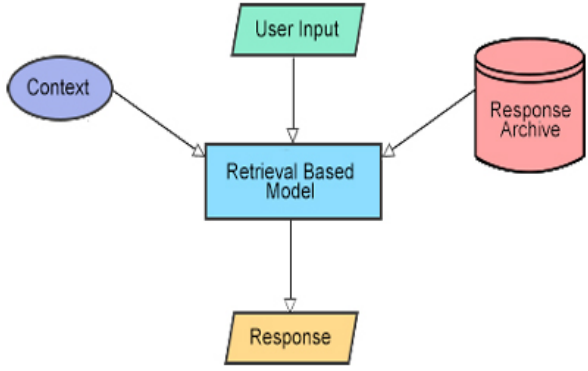
# Классификации диалоговых систем




https://www.codecademy.com/articles/what-are-chatbots

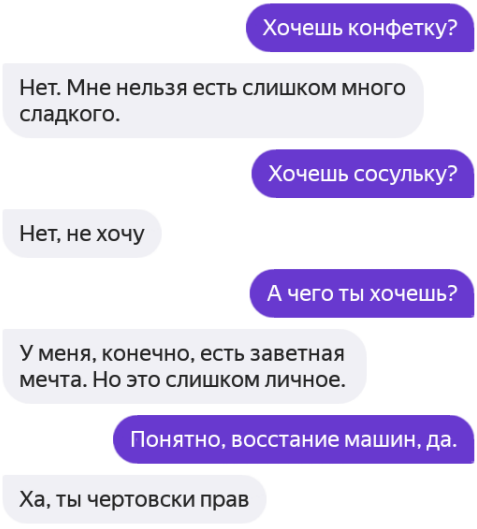
# Системы с подготовленными ответами (Retrieval-Based)

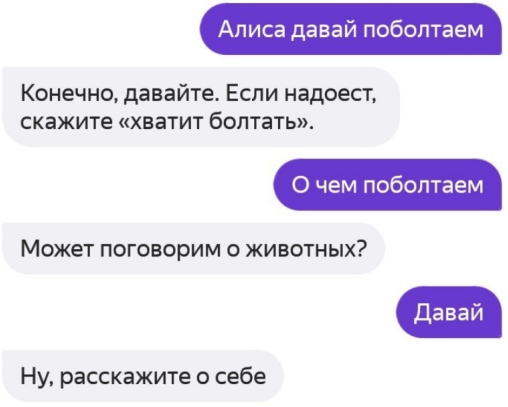
- Все возможные ответы подготовлены заранее
- Система ищет самый подходящий ответ для текущего момента диалога
- Ответы написаны грамматически верно и являются осмысленными
- Полноценную диалоговую систему общего назначения невозможно построить на основе готовых ответов
- Можно построить близкую к ней с большой шаблонизированной базой ответов




https://dzone.com/articles/significance-of-chatbot

# Диалоги в сложной Retrieval-Based системе

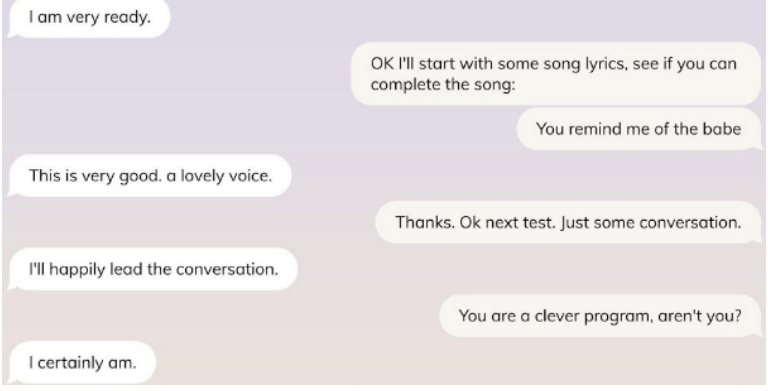



https://pikabu.ru/story/yandeks__alisa_6697271
http://joyreactor.cc/post/3994138

# Системы с генерацией ответов (Generative-Based)

- Возможны новые уникальные ответы
- Качество и адекватность ответов не гарантируются
- Редко применяются в бизнес-приложениях
- Являются перспективным направлением исследований
- В основном используются огромных нейросетевые модели (типа GPT-3)



# Основные компоненты чат-бота

- Intent Detection: подсистема выделения интента — извлекает из реплики пользователя его намерение

  **Я хочу заказать две пиццы 4 сыра**
  $\Rightarrow$ интент: «**заказ пиццы**»

- Slot Filling: подсистема заполнения слотов — извлекает из сообщения необходимые сущности

  **Я хочу заказать две пиццы 4 сыра**
  $\Rightarrow$ слоты: «количество»: «**2**», «название»: «**4-сыра**»

- Chat-bot Script: сценарий диалога — граф состояний, определяющий поведение бота в различных ситуациях
- Подсистема генерации ответов и действий

# Определение интента

- Определение интента — это задача классификации
- Входные данные:
  - Текущая реплика пользователя
  - Предыдущие реплики/интенты
  - Состояние бота
  - Любая иная полезная внешняя или внутренняя информация
- Выходные данные:
  - Интент реплики — метка класса намерения
  - Степень уверенности классификатора

# Подходы к определению интента

- Обучение сложной модели для классификации:
  - Сбор и разметка пользовательских диалогов
  - Обучение глубоких сетей на последовательностях реплики и их интентов

  **Долго, дорого и сложно**

- Обучение простой модели для классификации:
  - Сбор и разметка пользовательских реплик
  - Обучение любых моделей для отдельных репликах

  **Проще, чаще используется на практике**

- Построение классификатора на словарях и правилах:
  - Сбор списков слов/N-грамм + правила
  - Формирование списка сущностей + правила

  **Проще, чаще используется на практике**

# Заполнение слотов

- Слоты — дополнительные данные, которые нужны чат-боту для обработки реплики с данным интентом
- Слоты заполняются на основе сущностей, которые содержатся в реплике
- Не все сущности в реплике являются слотами, даже если они имеют релевантный тип:

Приветствую! Меня зовут Александр, у меня бизнес в Новосибе, но щас я по делам в Спб, нужно доехать до Москвы, в идеале сегодня, поможете? А то друг школьный туда из Владимира заехал до среды, хочу повидаться. Заранее спасибо!

$\Rightarrow$ слоты:

«город отправления»: «**Санкт-Петербург**»

«город прибытия»: «**Москва**»

«дата отправления»: «**11.11.2020**»

# Типы слотов

- Обязательные слоты:
  - **Примеры**:
    - Город отправления
    - Город прибытия
    - Дата отправления
  - Если в реплике нет обязательных слотов, их необходимо уточнить
- Опциональные слоты:
  - **Примеры**:
    - Класс билета
    - Вокзал прибытия
    - Вокзал отправления
    - Время отправления
  - Если в реплике есть опциональные слоты, их нужно учесть, если нет — игнорировать, не беспокоя пользователя

# Выделение сущностей для заполнения слотов

- Обучаемые модели:
  - Собираются реплики
  - Выбираются типы сущностей, выполняется разметка
  - Обучается модель для NER (BiLSTM или Transformer)
  - Поиск сущностей по предсказанию модели
  - Сложнее и дольше, лучше подходит для сложных типов сущностей (названия песен и альбомов)
- Правила и регулярные выражения:
  - Собираются реплики
  - Выбираются типы сущностей, собираются словари, пишутся регулярные выражения и эвристические правила
  - Поиск сущностей по соответствию правилу/словарю/рег. выражению
  - Проще и быстрее, лучше подходит для простых типов сущностей (даты, имена, телефоны)

# Пример: выделение дат

- Даты можно качественно выделять с помощью регулярных выражений
- Сперва выделяются текстовые фрагменты, содержащие описание даты:

  `Мне нужен билет на 17.03` $\Rightarrow$ 17.03

  `Хочу поехать на следующие выходные` $\Rightarrow$ следующие выходные

  `Собираемся в Мск через понедельник` $\Rightarrow$ через понедельник

- Затем фрагменты конвертируются в даты в стандартном формате:

  17.03 $\Rightarrow$ 17.03.2020

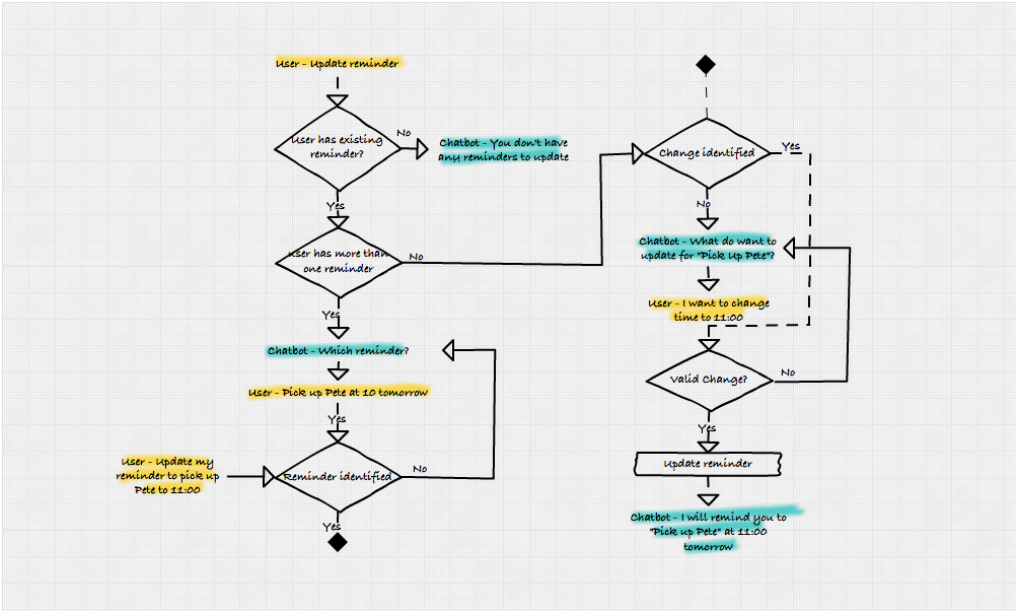
  следующие выходные $\Rightarrow$ 07.11.2020 / 08.11.2020

  через понедельник $\Rightarrow$ 16.11.2020

# Сценарий чат-бота

- Сценарий является основой логики работы любого чат-бота
- Последовательность действий бота:
  - Определение интента и заполнение слотов
  - Проверка внутреннего состояния в сценарии
  - Генерация ответа или выполнение действия
- Сценарий представляет собой конечный автомат
- На бытовом уровне это направленный граф:
  - Каждая вершина описывает какое-то состояние бота
  - Ребра описывают возможные переходы между состояниями
  - Есть правила перехода, зависящие от состояния и входной реплики
  - Есть правила, описывающие выходной ответ или действие в зависимости от состояния и входной реплики

# Пример сценария чат-бота



# Структура сервиса с чат-ботом

- Верхнеуровневая архитектура стандартная — клиент-сервер
- В роди клиента могут выступать
  - соцсети и мессенджеры (Telegram, FB Messanger, WhatsApp, VK)
  - виртуальные ассистенты (Alexa, Google Assistant, Алиса)
  - любые подходящие мобильные приложения
  - виджеты на сайтах
- Серверная часть состоит из
  - собственно веб-сервера, обрабатывающего запросы
  - сервиса, реализующего логику чат-бота
  - сервиса/интерфейса устройства, с которым бот взаимодействует, выполняя команды пользователя
  - любых дополнительных инфраструктурных компонентов (базы данных)
- Клиент и сервер взаимодействуют через API / Webhooks

# Разработка сервиса с чат-ботом

Разработка с нуля:

- Нужно разрабатывать сервер и инфраструктуру
- NLU и логику можно разрабатывать как угодно гибко
- Нужно настраивать интеграции с клиентами
- Требует экспертизы в разработке ПО и NLP
- Подходит для больших сложных и высоконагруженных систем
- Полностью контролируется заказчиком

Разработка на базе платформы:

- Сервер и инфраструктура от идут платформы
- NLU и логика разрабатываются инструментами платформы
- Интеграция с клиентами обеспечивается платформой
- Разработчики могут не быть экспертами
- Подходит для простых ботов со слабым NLU и низкой нагрузкой
- Может не работать on-premise, у заказчика нет контроля

# Платформы для разработки чат-ботов

- Платформы от технологических гигантов с поддержкой русского языка:
  - Google Dialogflow
  - IBM Watson
  - Facebook Messenger Platform
- Платформы от технологических гигантов без поддержки русского языка:
  - Amazon Lex
  - Microsoft LUIS
  - Baidu KITT.AI
- Локальные русскоязычные платформы:
  - Electra.AI
  - Chatme.AI
  - Just AI Conversational Platform

# Фреймоворки для разработки чат-ботов

- Фреймворки — промежуточный вариант между платформой и разработкой с нуля
- В отличие от платформ, фреймворки ориентируются на разработчиков и специалистов в ML и NLP
- Фреймворки упрощают решение части задач, в целом сохраняя гибкость процесса, они дают:
  - Унифицированный язык описания интентов, слотов, сценариев и ответов
  - Упрощѐнный интерфейс для обучения моделей и описания правил
  - Коллекции предобученных моделей для типовых задач
  - Базовые инструменты для тестирования и аналитики
- Один из наиболее используемых фреймворков в мире — Rasa
- Для русского языка также популярен DeepPavlov.AI

# Метрики качества бота

- Технические метрики:
  - Качество классификатора интентов
  - Качество выявления сущностей
- Бизнес-метрики:
  - Возвращаемость пользователей (Retention)
  - Доля успешных диалогов
  - Доля обработанных ботом обращений от общего их числа
  - Доля реплик, которые бот не смог распознать
  - Средняя длина диалога
  - Отзывы пользователей
  - Тональность реплик пользователей
  - Количество принесенных ботом денег
  - ...

# Виртуальные ассистенты

- Виртуальный ассистент (Intelligent Virtual Assistant) — диалоговая система общего назначения для решения спектра задач на основе текстовых и голосовых команд пользователя
- Многие крупные IT-компании имеют собственных ассистентов со своей программной экосистемой:

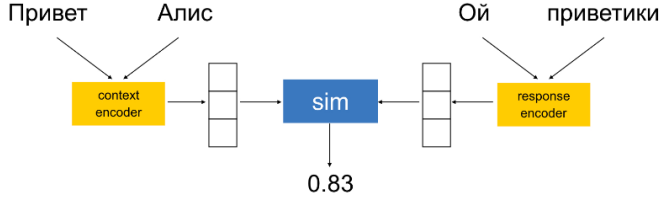
- Google Assistant (Google)
- Alexa (Amazon)
- Siri (Apple)
- Cortana (Microsoft)
- Bixby (Samsung)
- Алиса (Яндекс)
- Маруся (Mail.ru)
- Celia (Huawei)
- Duer (Baidu)
- AliGenie (Alibaba Group)

# Функции виртуальных ассистентов

- Стандартные задачи для виртуальных ассистентов:
  - установка будильников и напоминаний
  - поиск и запуск воспроизведения медиа-контента
  - информирование о погоде, пробках, курсах валют
  - редактирование списка контактов
  - ввод и отправка сообщений
  - поиск информации в Интернете
- По сути, ассистент — это мощный чат-бот с голосовым интерфейсом
- При его построении используются те же сценарии, классификаторы интентов и NER
- В отличие от простых ботов, в ассистентах такие задачи решаются в основном с помощью глубоких нейросетей
- Некоторые ассистенты умеют работать в режиме свободного диалога

# Свободный диалог

- Если явного интента в пользовательской реплике нет, то ассистент может поддержать свободное общение без конкретной цели
- При формировании ответа используется Retrieval-Based подход — все возможные реплики подготовлены заранее
- Общим подходом является использование сиамских сетей с двумя башнями-кодировщиками:



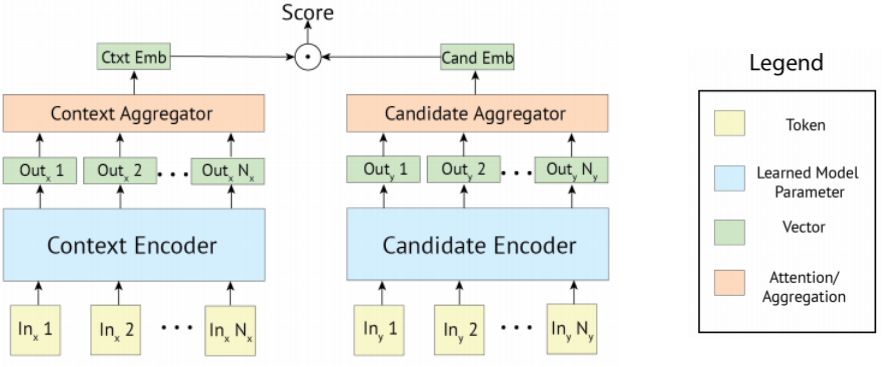

- За счёт того, что база ответов огромна и регулярно пополняется, в пределах 1-2 реплик может создаться иллюзия общения с живым человеком

# Подходы к построению модели

- Используются только нейросетевые модели, вход — реплика-кандидат и контекст, выход — векторы, подаваемые в функцию близости
- Контекст может состоять как из одной текущей реплики пользователя, так и из части предшествующего её диалога
- Из наиболее релевантных реплик-кандидатов выбирается лучшая с точки зрения дополнительных ограничений
- Варианты устройства модели:
  - Параллельное кодирование (Bi-Encoder)
  - Кросс-кодирование (Cross-Encoder)
  - Поли-кодирование (Poly-Encoder)
- В качестве кодировщика лучше всего работаю сети на основе Transformer, например BERT и его модификации

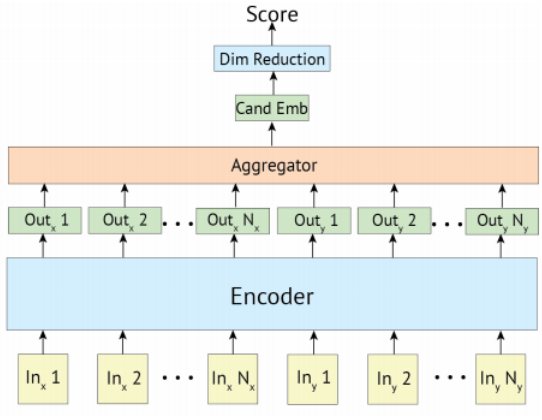
# Параллельное кодирование

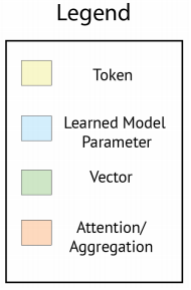
- Контекст и кандидат векторизуются по отдельности своими сетями-кодировщиками
- Векторы для всех ответов подготавливаются заранее
- На этапе вывода достаточно закодировать контекст
- Дальше используются эффективные библиотеки для поиска ближайших соседей (Faiss, Scann)



# Кросс-кодирование

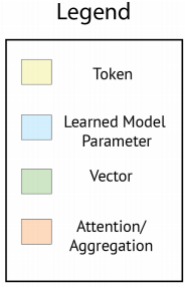
- Контекст и каждый кандидат векторизуются совместно в одной модели
- Такой подход даёт более качественный результат
- Но работает гораздо медленнее и плохо масштабируется

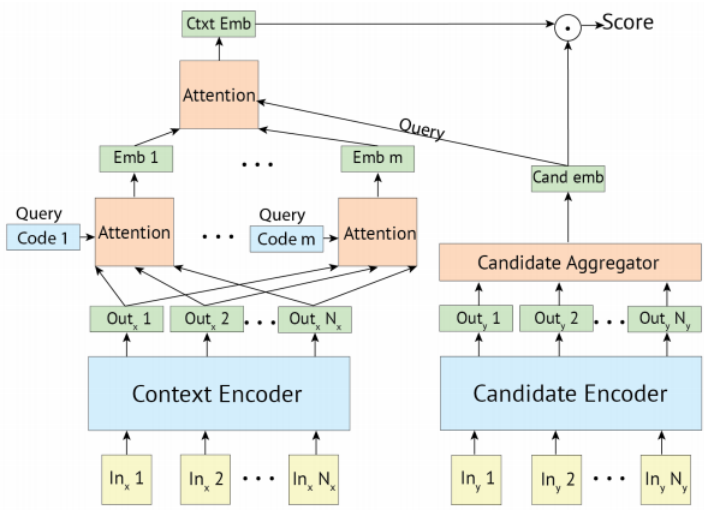


# Поли-кодирование

- Попытка объединить преимущества двух подходов
- Ответы векторизуются заранее
- Токены контекста переводятся в $m$ векторных кодов с помощью механизма внимания ($m$ сильно меньше числа токенов)
- Вектор ответа служит запросом к механизму внимания на этих кодах



# Обучение кодировщиков

- Обе сети предобучаются попеременно на задачах предсказания
  - маскированных токенов
  - следующего предложения
  - следующего высказывания (набора предложений)
- Затем модели дообучаются на целевую задачу ведения диалога с нужным набором данных
- Качество данных играет решающую роль, их можно собирать из реальных ресурсов или с помощью краудсорсинга

# Данные для обучения

- Положительные примеры набираются из последовательно идущих реплик реальных диалогов
- Отрицательные обычно набираются случайно
- Для повышения качества стоит добавлять в выборку примеры, на которых модель сильно ошибается
- В кандидаты важно добавлять не все реплики, а только удовлетворяющие определѐнным условиям (например, без ругательств)
- Но в обучающей выборке «плохие» реплики быть должны, иначе сеть будет реагировать на них неадекватно

# Навыки для 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
- Бот подключается к ассистенту и может быть вызван из него с помощью ключевых фраз
- Разработчик получает от экосистемы ассистента
  - Преобразование голоса в текст
  - Синтез речи из текста
  - Графический интерфейс (на устройствах с экраном)

# Вопросно-ответная система

- Вопросно-ответная система (QA-system) — подвид (неформально) диалоговой системы, предназначенный для ответов на вопросы
- Задача QA находится на стыке информационного поиска и NLP
- QA-система должна:
  - Понимать запрос пользователя на естественном языке
  - Искать ответ в огромных структурированных или сырых массивах данных
  - Формировать ответ на понятном человеку языке
- Вопросно-ответная система является важным компонентом поисковой системы и использующих ее виртуальных ассистентов
- Построение хорошей QA-системы — сложная задача, которую пытаются решать с помощью глубоких нейросетевых моделей

# Типы вопросов для QA-систем

- Фактологические вопросы (что? где? когда?):
  - Кто автор пьесы «Три сестры»?
  - Какая столица у Гондураса?
  - В каком году разрушили Карфаген?
- Вопросы на понимание здравого смысла:
  - Вопрос: Где можно плавать?
  - Ответы: а) бассейн б) море в) небоскреб
- Сравнительные и оценочные вопросы:
  - Какое метро старше, Лондонское или Московское?
  - Какой из океанов меньше всех по площади?
- Вопросы по прочитанному тексту:
  - Текст: «Собака помчалась за кошкой, та перепрыгнула через изгородь»
  - Вопрос: Кто перепрыгнул через изгородь?

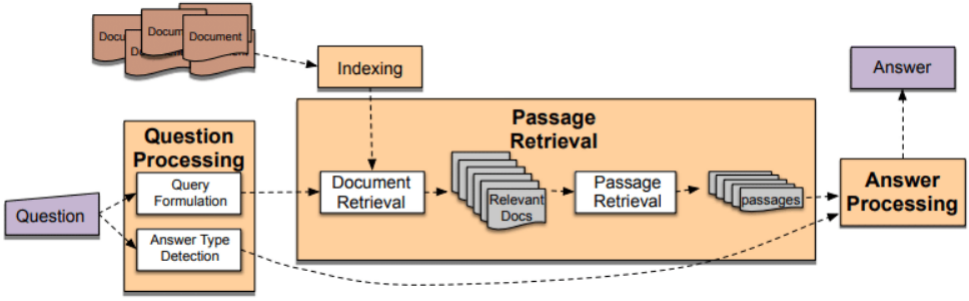
# Подходы к построению QA-систем

- Информационный поиск (IR-Based QA):
  - Дана текстовая коллекция, развитая на параграфы
  - Система ищет параграф, содержащий фрагмент-ответ на вопрос и выделяет этот фрагмент
- Поиск по базе/графу знаний (KB-Based QA):
  - Дана база знаний в структурированном виде
  - Система формирует на основании вопроса запрос и отправляет его в базу:

    Когда умер Эрнест Хемингуэй? $\Rightarrow$ дата-смерти(Эрнест Хемингуэй, ?)

  - Такой подход неплох для фактологических вопросов, но весьма сложен (пример использования — IBM Watson)

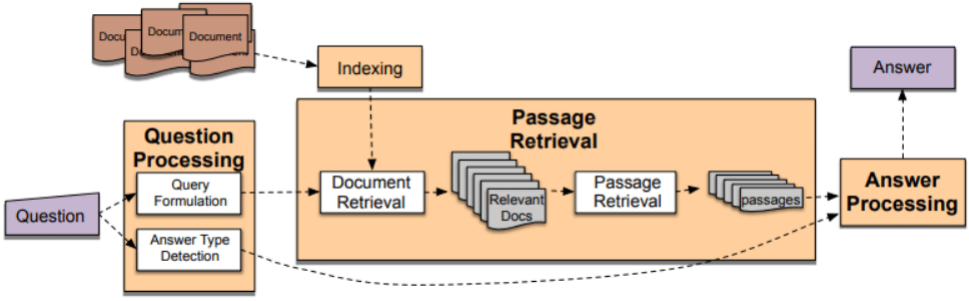
# IR-Based QA-системы




1. Обработка вопроса
   - определение типа ответа (персона, геолокация, метка времени)
   - определение ключевых сущностей
   - определение типа вопроса
2. Формирование запроса
   - переформулировка запроса: удаление wh-слов, изменение порядка слов
   - расширение запроса (+ синонимы, исправление опечаток и т.п.)

https://www.researchgate.net/figure/The-architecture-of-IR-based-question-answering-system_fig1_328861207

# IR-Based QA-системы




3 Извлечение параграфов и фрагментов из них

4 Извлечение и обработка ответа:

**Вопрос**: `Что означает апостроф в английской транскрипции?`

**Фрагмент+ответ**: `На самом же деле, в транскрипции английского ничего сложного нет, например, ударение в транскрипции отмечается апострофом`

# Набор данных SQuAD

**Passage**: Tesla later approached Morgan to ask for more funds to build a more powerful transmitter. **When asked where all the money had gone, Tesla responded by saying that he was affected by the Panic of 1901**, which he (Morgan) had caused. Morgan was shocked by the reminder of his part in the stock market crash and by Tesla's breach of contract by asking for more funds. Tesla wrote another plea to Morgan, but it was also fruitless. Morgan still owed Tesla money on the original agreement, and Tesla had been facing foreclosure even before construction of the tower began.

**Question**: On what did Tesla blame for the loss of the initial money?

**Answer**: Panic of 1901

- SQuAD (Stanford Question Answering Dataset) — канонический набор данных для IR-based QA-систем
- Состоит из 100K троек
  - вопрос
  - параграф с ответом
  - фрагмент-ответ
- В SQuAD 2.0 добавлены 50K вопросов с релевантными параграфами, в которых фрагмента-ответа нет

# Набор данных SQuAD

- Формирование датасета:
  - Отобраны топ-10k статей английской Википедии по PageRank, из них случайно выбрано 536
  - Каждая статья была разбита на параграфы
  - Краудсорсинг: сформулировать до 5 запросов к каждому из параграфов
  - Вопрос формулировался своими словами, без копирования текста из параграфа
  - Анализ: разнообразие ответов, сложность вопросов, степень синтаксического различия между вопросом и ответом
- Аналоги SQuAD для других языков:
  - SberQuAD (для русского языка)
  - KorQuAD
  - FQuAD
  - Neural Arabic QA
  - Spanish SQuAD
  - Italian SQuAD
  - XQuAD (смесь из 11 языков)

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

# Решение задачи SQuAD (IR-Based QA)

- Первый этап: отбор параграфов-кандидатов для ранжирования
  - формирование запроса
  - поиск близких к нему параграфов с помощью легковесных методов
  - например, косинусное расстояние между векторами TF-IDF или FastText
- Второй этап: поиск в параграфах фрагментов-ответов
  - классификация токенов в параграфах-кандидатах: каждое слово проверяется на то, является ли оно какой-то частью ответа
  - обычно ищут начало и конец фрагмента
  - используются тяжеловесные нейросетевые модели

# Модель DrQA

- Извлечение документов:

  ищем пять ближайших статей Википедии, используя близость по векторам TF-IDF

- Чтение документа:

  получаем запрос $q = q_1, \ldots, q_\ell$ и $m$ параграфов $p_1, \ldots, p_m$

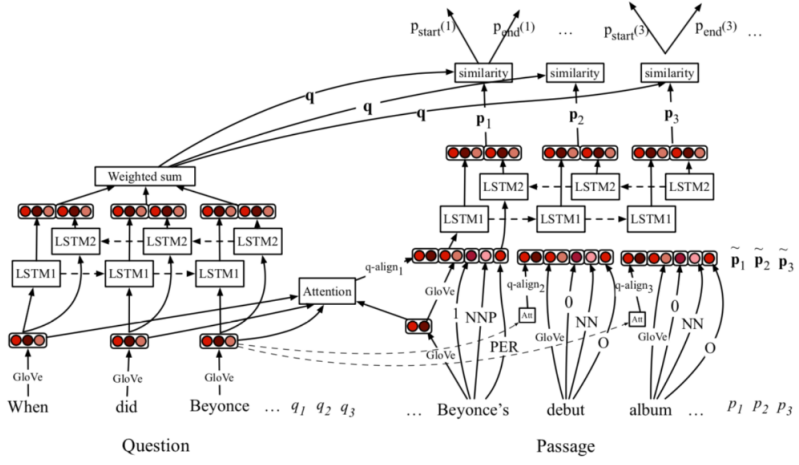
- Кодирование вопроса:

  взвешенная сумма выходов BiLSTM на словах запроса

- Кодирование параграфа:
  - Для каждого слова берём его эмбеддинг (GloVe)
  - Добавляем флаг точного совпадения с одним из слов запроса
  - Добавляем морфологические признаки
  - Пропускаем через BiLSTM, получаем выходной вектор

# Модель DrQA

- Предсказание:
  - используем $P_{start} \propto \exp(p_i W_s q), P_{end} \propto \exp(p_i W_e q)$
  - выбираем наилучший фрагмент от $i-$го токена до $i'-$го токена, такие что $i \geqslant i' \geqslant i + 15$ и $P_{start}(i) \times P_{end}(i')$ максимально

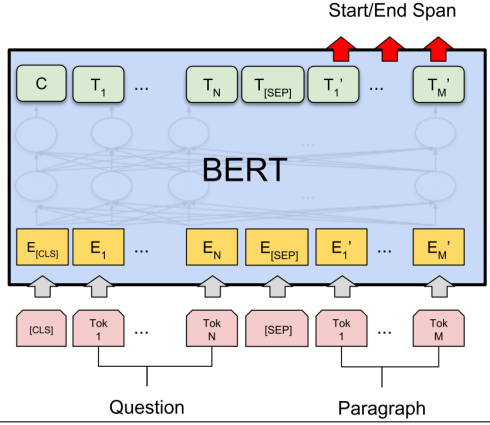


# Аналогичные модели

- После DrQA появилось еще несколько моделей, работающих по аналогичному принципу
- В основе остается BiLSTM, но добавляются
  - дополнительные рекурентные слои
  - механизмы внимания
  - сверточные слои для символов
  - ...
- Примеры моделей:
  - BiDAF
  - R-NET
  - QA-NET
- Решать задачу SQuAD можно решать с помощью предобученных Transformer-based моделей

# BERT для QA

- Предобученные нейросетевые языковые модели эффективны в определении связей между парами предложений (парафраз и логическое следование)
- Задача SQuAD (поиск ответа в абзаце) отлично ложится на BERT
- В лидерборде SQuAD сейчас лидируют ансамбли на базе ALBERT и XLNet




https://rajpurkar.github.io/SQuAD-explorer
https://blog.scaleway.com/understanding-text-with-bert

# Библиотека Haystack

- Haystack — библиотека с открытым кодом для построения QA-систем
- Реализует описанную двухуровневую логику поиска ответов
- Первый этап выполняется с помощью
  - TF-IDF
  - BM25
  - запросы на движке Elasticsearch
  - близость векторных представлений токенов/параграфа
- Второй этап выполняется с помощью предобученных моделей на базе Transformer (библиотека Hugging Face)
- Библиотека предоставляет
  - хранилища данных
  - REST API для доступа к сервису
  - удобный программный интерфейс на Python

# KB-Based QA-системы

- Общий пайплайн обработки запроса:

  **Пример:** Когда умер Эрнест Хемингуэй?

  1. Выделение именованных сущностей (Эрнест Хемингуэй)
  2. Классификация вопроса и определение предиката (, когда, умер)
  3. Поиск сущности в базе (проблема: разрешение неоднозначности)
  4. Формирование итогового ответа

- Существующие системы плохо справляются с вопросами, в которых
  - более одной сущности
  - требуется сделать несколько запросов к базе

  **Пример:** Когда умерла мать Эрнеста Хемингуэя?

# Итоги занятия I

- Диалоговые системы могут классифицироваться по широте предметной области и по методу формирования ответа
- Чат-боты — способ организации общения человека с компьютером через текстовый канал, популярный инструмент в бизнесе
- Для построения бота нужно уметь решать задачи классификации и NER
- Качество сценария бота имеет решающее значение, его важно постоянно тестировать на реальных пользователях
- Платформы для создания ботов могут упростить разработку, но сильно урезают гибкость и степень контроля
- Виртуальный ассистент — многофункциональный чат-бот с голосовым интерфейсом, иногда умеющий поддержать свободный диалог

# Итоги занятия II

- Системы свободного диалога строятся на подборе наиболее подходящей из имеющихся в базе реплик в зависимости от контекста диалога
- Для выбора реплик используются их векторные представления, генерируемые кодировщиками
- В роли кодировщиков выступают разные нейросети, хорошо работаю модели на основе Transformer
- Существующие QA-системы классифицироваться по типу обрабатываемых запросов и по методу построения
- Существуют два основных подхода к построению — IR-Based и KB-Based
- Наиболее популярная задача для IR-based подходов — SQuAD 2.0 с одноимѐнным набором данных
- Задача решается с помощью предобученных моделей на основе рекурентных сетей или Transformer