Чему недавно научились **диалоговые помощники**, и чего они ещё не умеют

Давид Дале,

выпускник ФКН

15 сентября 2020, ПМИ ФКН

Давайте познакомимся

- Давид Дале [t.me/cointegrated]
- Выпускник магистратуры ФКН (НоД+ШАД) 2016
- Research Engineer в NLP группе в Сколтехе
- Фриланс-разработчик бэкенда и чатботов
- Работал:
  - в разработке голосовых ассистентов (Алиса)
  - в ML-консалтинге (Yandex Data Factory)
  - в кредитном скоринге (Альфа-Банк)
- Преподавал теорию игр в ВШЭ и теорвер в Тель-Авивском ШАДе

О чем сегодня поговорим

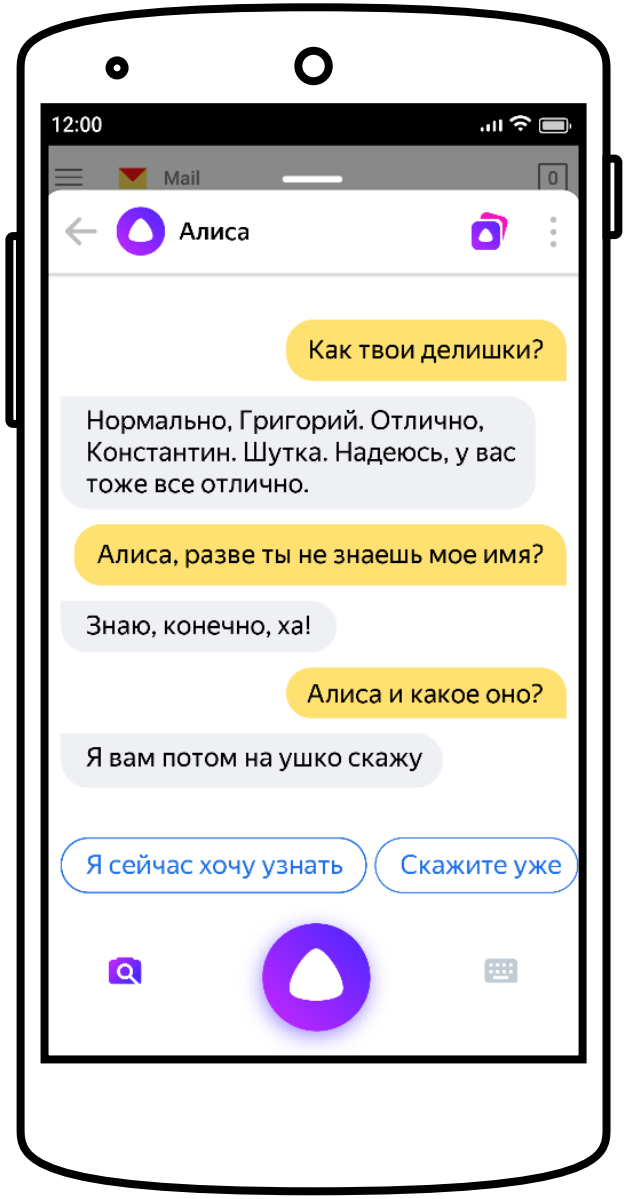
- Диалоговый помощник: чего от него хочется?
- Архитектура ассистента на примере Алисы
- Гонка гигантов: как соревнуются болталки от FAANG
- Каким мог бы быть goal-oriented диалоговый помощник будущего

- Если будет время и желание: свободный разговор о создании собственных навыков для голосовых помощников

Алиса снаружи и изнутри

Алиса – это

- «Полезные» сценарии
  - Разбирают смысл фраз
  - Ходят во внешние API
  - Помогают решать задачи
  - Важно: краткость и точность
- Разговор на общие темы
  - Подбирают уместный ответ
  - Только слова, без действий
  - Развлекают
  - Важно: непротиворечивость, уместность, разнообразие, проактивность, юмор



Внешние навыки Алисы

- «Много маленьких автономных ботов внутри Алисы»
- "request": {

  "command": "где ближайшее отделение",

  "original_utterance": "Алиса спроси у Сбербанка где ближайшее отделение",

  "type": "SimpleUtterance", "payload": {}

  }
- "response": {

  "text": "Ближайшее отделение по адресу Земляной вал 41, строение 1.",

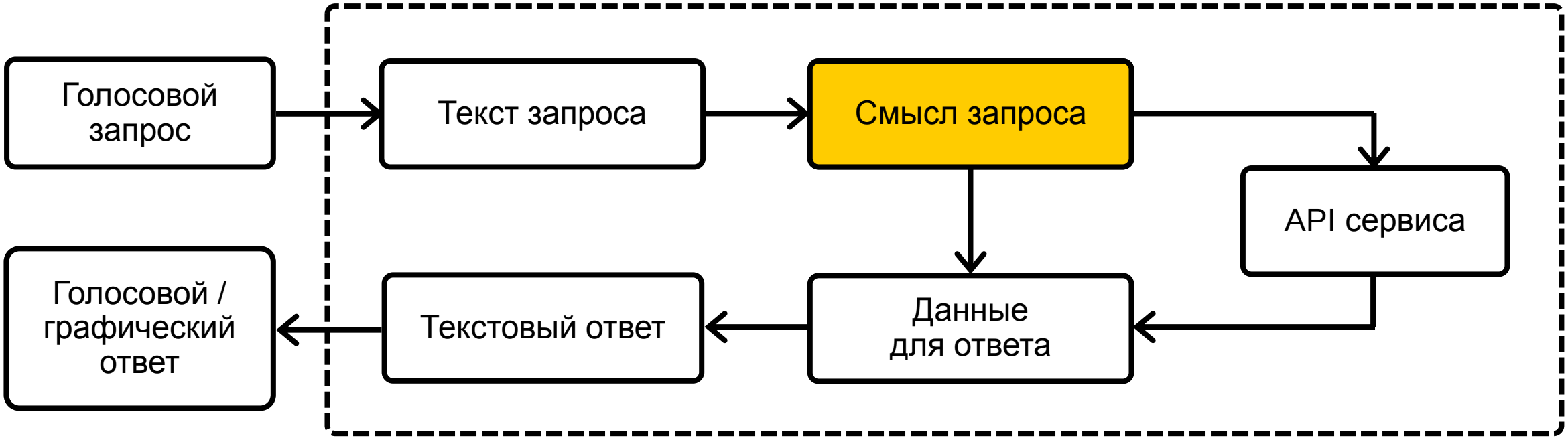
  “buttons": [ ], "end_session": false

  }

Кто работает над Алисой

- Разработчики
  - Разработка сценариев (в основном «полезных»)
  - MLщики (в основном – для «болталки» и поиска)
  - Разработка инфраструктуры
- Тестировщики
- Аналитики
- Продакты и другие менеджеры
- …

Устройство голосового помощника



- Если уже есть сервис и есть голосовой API (Яндекс SpeechKit), то самое сложное – разобрать смысл запроса

Как Алиса формирует ответ

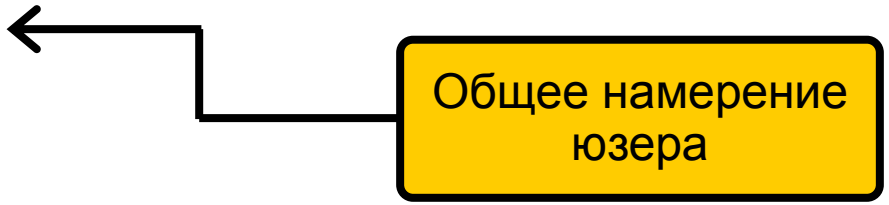
Обработка запроса пользователя

«Маршрут до петухова пятьдесят шесть через троллейную сто двадцать восемь дробь один»

Разбор запроса по смыслу

«Маршрут до петухова пятьдесят шесть через троллейную сто двадцать восемь дробь один»

```
{
  intent: "route"
}
```



Разбор запроса по смыслу

«Маршрут до петухова пятьдесят шесть через троллейную сто двадцать восемь дробь один»

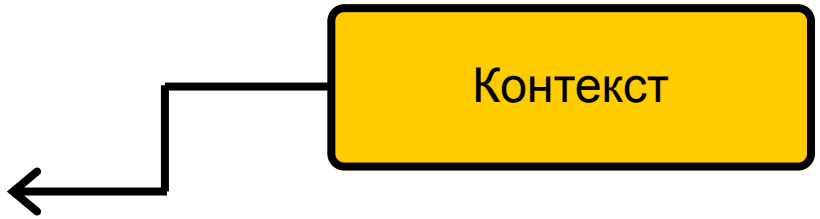
```
{
  intent: "route",
  slots: {
     "address_to":  {"street": "Петухова", " house_number": "56"},
     "address_via": {"street": "Троллейная", " house_number": "128/1"},
   }
}
```

Важные сущности в запросе

Управление диалогом

«Маршрут до петухова пятьдесят шесть через троллейную сто двадцать восемь дробь один»

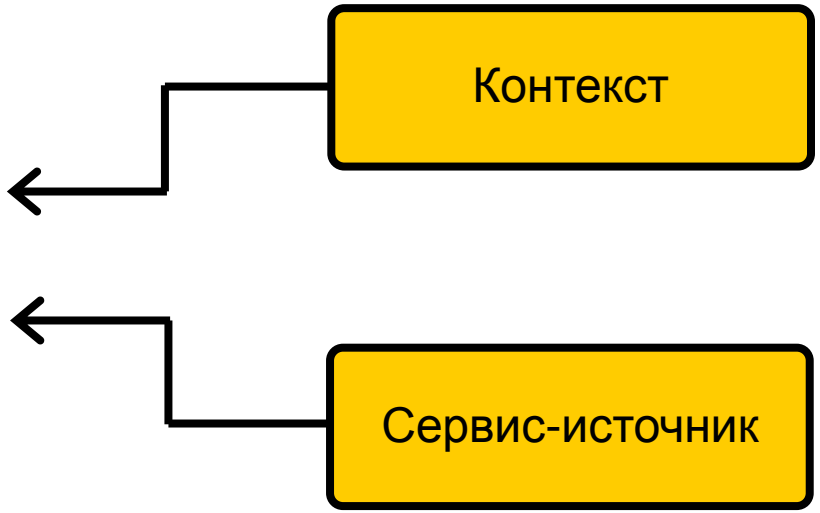
```
{
  intent: "route",
  slots: {
    "address_to":  {"street": "Петухова", " house_number": "56"},
    "address_via": {"street": "Троллейная", " house_number": "128/1"},
    "address_from": {"street": "Депутатская", " house_number": "46"},
   }
}
```



Взаимодействие с источниками

«Маршрут до петухова пятьдесят шесть через троллейную сто двадцать восемь дробь один»

```
{
  intent: "route",
  slots: {
    "address_to":  {"street": "Петухова", " house_number": "56"},
    "address_via": {"street": "Троллейная", " house_number": "128/1"},
    "address_from": {"street": "Депутатская", " house_number": "46"},
    "route": {"length": 16, "duration": 42, "url": "http://yandex.ru/maps..."}
   }
}
```

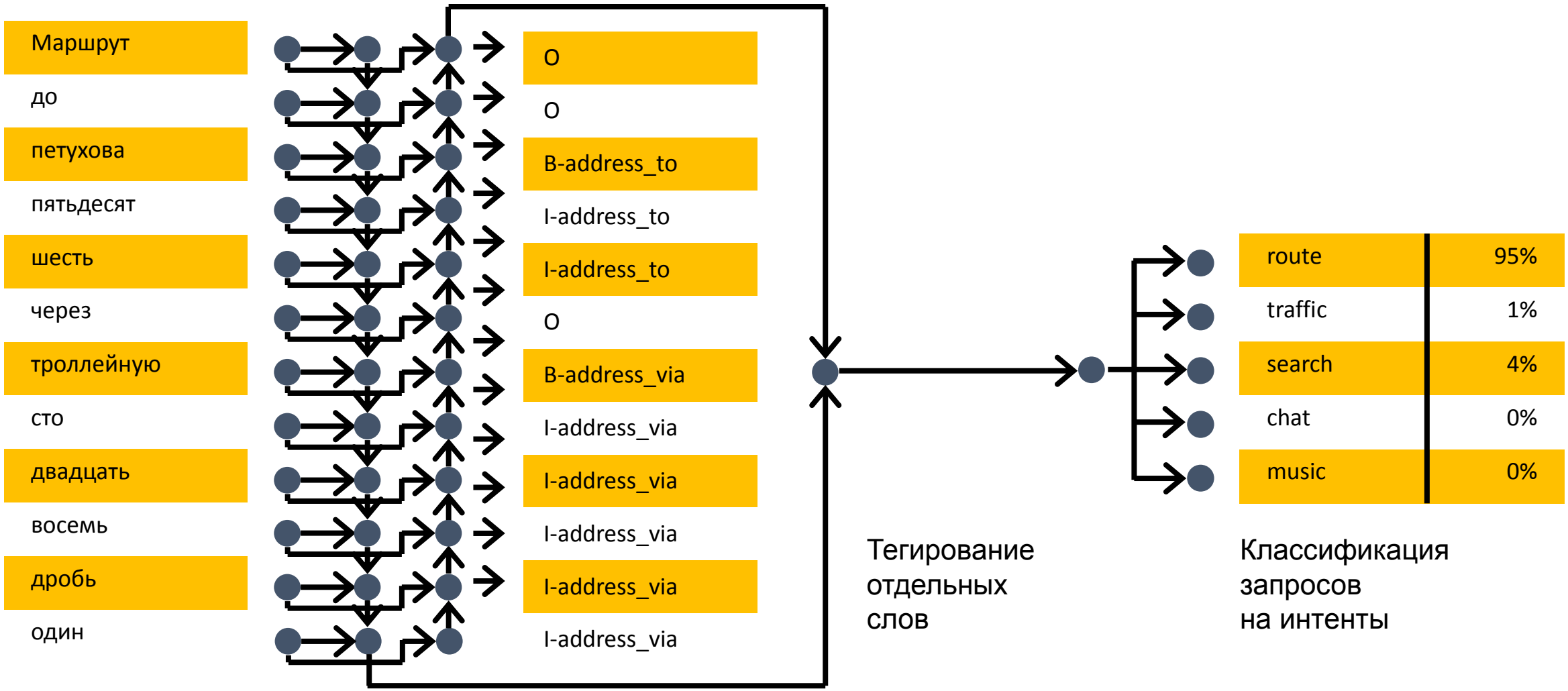


Генерация ответа

«Маршрут до петухова пятьдесят шесть через троллейную сто двадцать восемь дробь один»

«Дорога займёт 42 минуты на транспорте»

Решение с помощью нейронных сетей



Тегирование слов

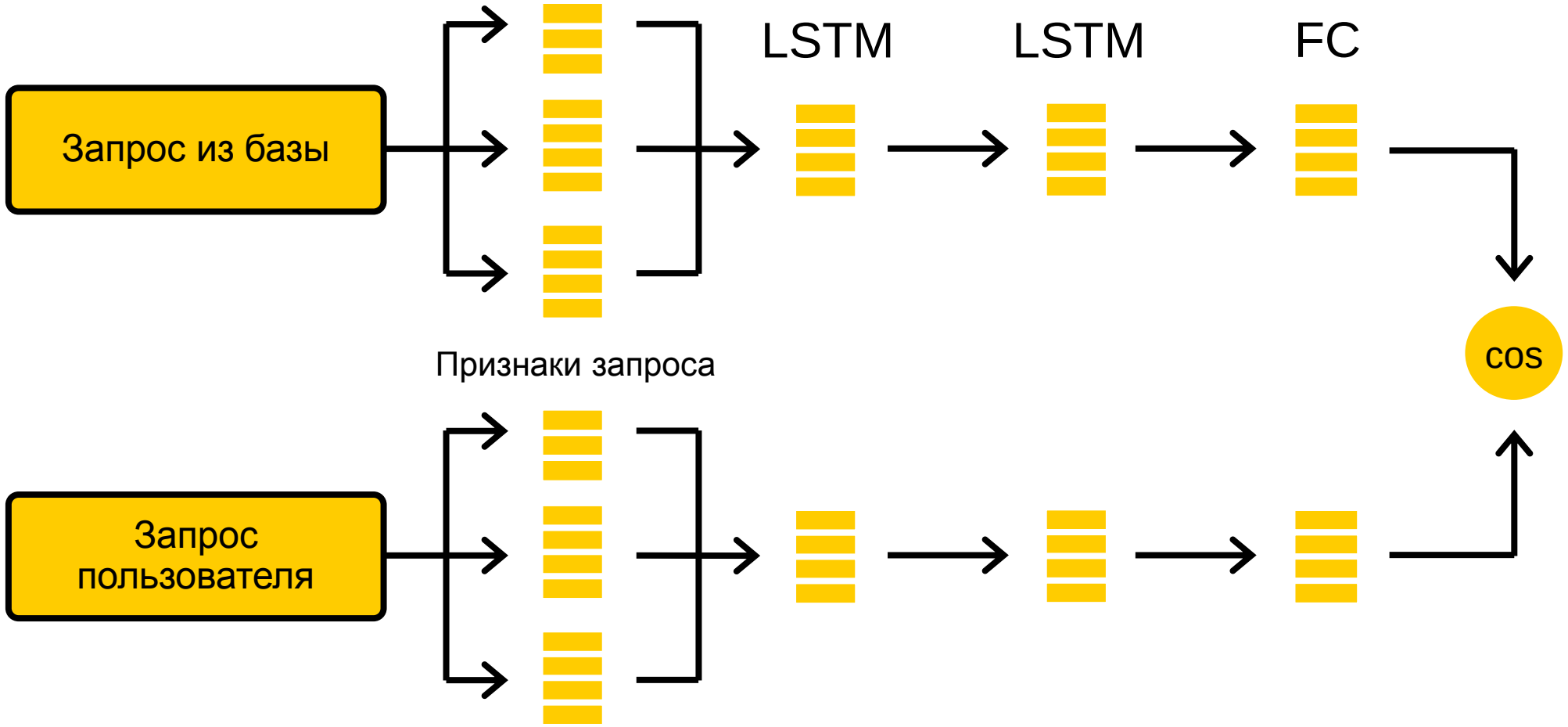
- «Мне пожалуйста биг мак фанту ноль пять и большую картошку с собой»

O	O	item	item	item	item	item	O	item	item	way	way
O	O	B-item	I-item	B-item	I-item	I-item	O	B-item	I-item	B-way	I-way


```
{
 intent: "new_order",
 slots: {
   "item":  [ {"product": "bigmac"}, {"product": "fanta", "size": "0.5"}, {"product": "fries", "size": "large" } ],
   "way": "to_go"
}
```


- IOB-теги – способ превратить наложить форму на текст. В нашем случае, каждый item придётся парсить повторно (уже независимо от остальных)

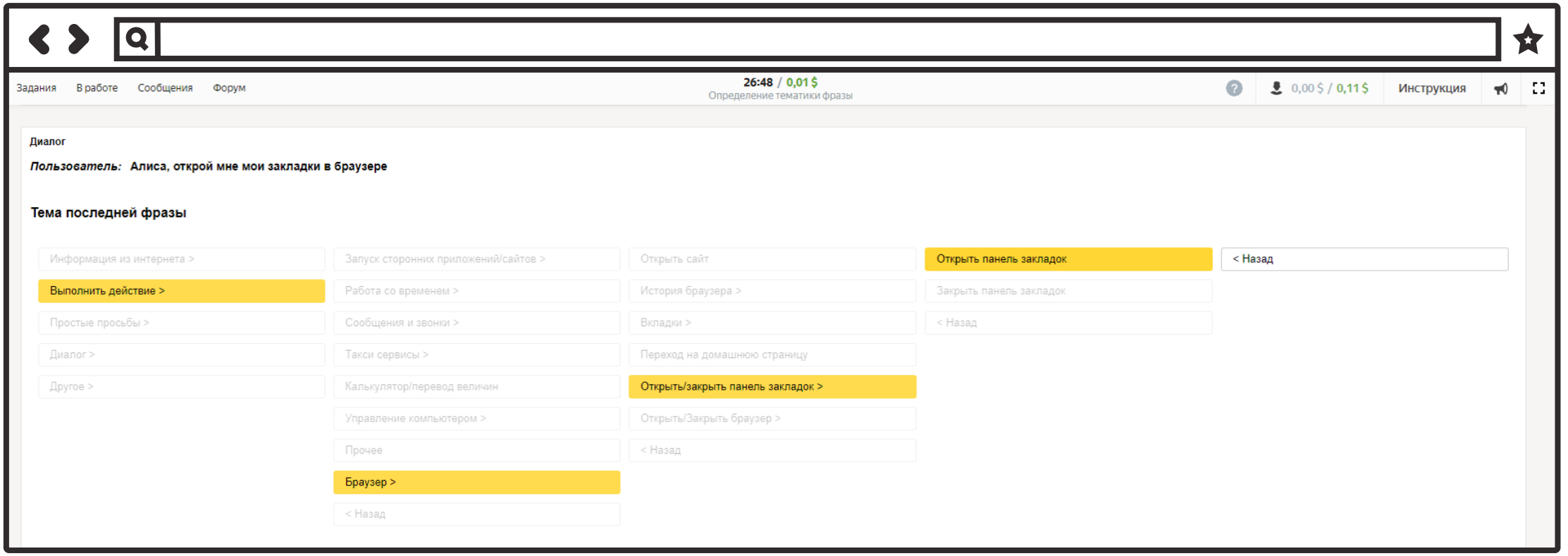
Сиамская сеть для классификации



Подробности в докладе Николая Любимова
на DataFest: youtu.be/4As-5fhDvsU?t=15736

Откуда взять размеченные данные

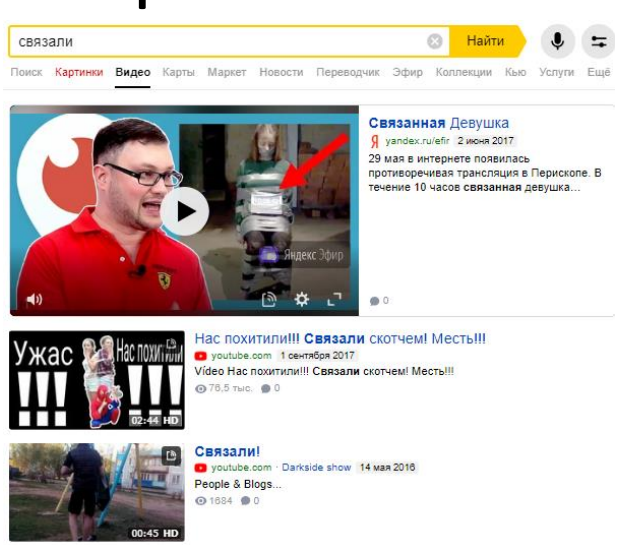
- Яндекс.Толока – краудсорсинговая платформа для разметки данных.

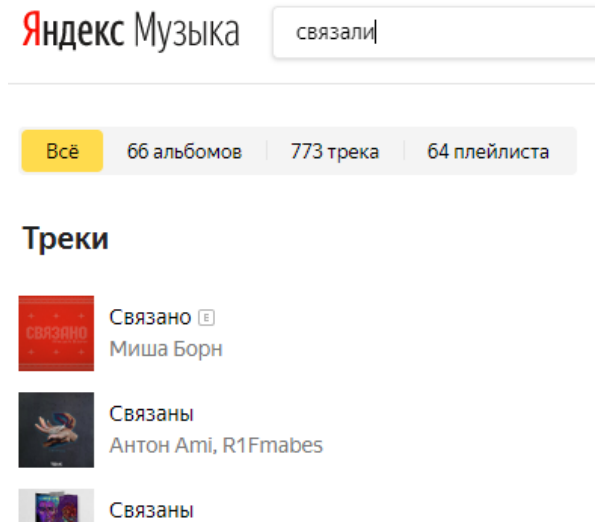


Грамматики

- Как создать NLU, не имея обучающих данных?
- Придумаем собственную грамматику для запросов маршрута:
  - <Number> = {0, 1, 2, 3, …}
  - <HouseNumber> = {<Number>, <Number> дробь <Number>, …}
  - <Street> = {Абельмановская, Абрамцевская, Авангардная, Авиаконструктора Микояна, …}
  - <Address> = {<Street> <HouseNumber>, улица <Street> дом <HouseNumber>, …}
  - <Destination> = {до <Adddress>, в <Address>, к <Address>, …}
  - ...
  - <Route> = {как доехать <Source> <Destination>, маршрут <Destination>, …}
- Такое можно использовать и для разбора, и для генерации фраз

Ранжирование сценариев

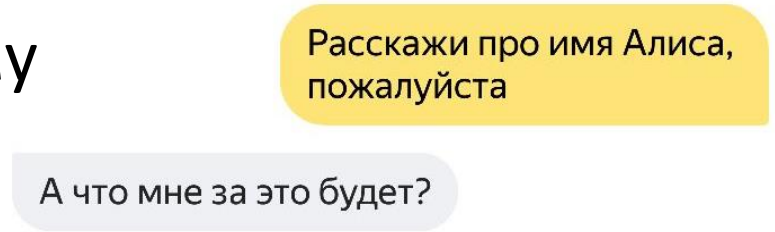




- «Алиса, включи Связали»
  - Это фильм?
  - Это песня?
  - Это внешний навык?
  - «Свет в зале»?
  - Если ничего из этого – хорошо ли ответит веб-поиск? Болталка?
- Чего хотим: выбирать сценарий так, чтобы дать наилучший ответ
  - опрашиваем одновременно n сценариев
  - по набору контекст-запрос-сценарий-ответ предсказываем оценку
  - выбираем сценарий с наивысшей предсказанной оценкой

Болталка

- Задача: поддержать разговор на любую тему
- Генеративный подход
  - Запрос + контекст --> нейронка --> ответ
  - Такое сложно обучать (но все пытаются, включая Гугл)
  - Сложно контролировать качество
  - Ответы (теоретически) могут быть очень разнообразными
- «Поисковый подход»
  - Запрос + контекст + готовый ответ --> нейронка --> оценка уместности
  - Базу готовых ответов можно пофильтровать как угодно
  - Проще добиться осмысленности и разнообразия
- Переранжирование гипотез генеративной и поисковой болталки



Недавние достижения и нерешённые задачи

Генерация осмысленного текста

- Легко генерировать однообразный осмысленный текст: шаблоны
- Несложно генерировать «какой-то» разнообразный текст: можно предсказывать языковой моделью слово за словом
- Не очень сложно генерировать текст с конкретным смыслом: это уже давно делают переводчики
- Сложно: сгенерировать текст
  - с передачей определенного смысла (или на заданную тему)
  - с долей импровизации
  - без вранья
  - в заданном стиле

 интересные чатботы

 улучшение перевода

 суммаризация информации

 голосовые интерфейсы ко всему

Transfer learning

- Для хорошего машинного обучения нужно много данных
  - Размечать данные для своей (узкой) задачи – недёшево
  - Идея: обучить модель решать иную задачу, для которой не нужны дорогие данные. Потом адаптировать её для своей задачи
- Бесплатные данные: тексты
  - Предсказывать следующее слово в тексте или пропущенное слово
  - Можно обобщить для генерации текстов или понимания их смысла
  - Например: word2vec или BERT
- За что боремся:
  - Одной и той же предобученной моделью решать всё больше задач
  - Дообучать модели на всё меньших объемах данных (в идеале zero-shot)
  - Трансфер знаний из одних языков в другие

GPT-3 (OpenAI)

- Как и предшественники, это *авторегрессионная языковая модель*, обученная предсказывать следующее слово на 1 ТБ текстов
- По факту, может правдоподобно продолжать длинные тексты
- Если собрать текст из пар $[x_1 y_1]\ [x_2 y_2]\ \ldots [x_{k-1} y_{k-1}][x_k]$, оказывается, модель достаточно часто угадывает правильные y_k
- Это значит: возможно, модель даже не нужно *дообучать* на новых задачах
- Проблема: медленно, дорого, не очень практично

Meena (Google)

- Огромная GPT-подобная болталка
- Обучали на диалогах из соцсетей (наверное)
- Показали, что чем лучше модель угадывает следующие слова, тем более качественный она собеседник
- Возможно, уже применяют это в Google Assistant

BST* (Facebook Research)

- Ещё одна большая болталка (точнее, микс из трёх)
- Обучали на соцсетях и на краудсорсинговых данных (в т.ч. на обсуждениях Википедии)
- Миксуют генерацию текста и выбор готовых ответов, как Алиса
  - За счёт этого поддерживают ещё более широкий спектр тем
- Кажется, пока что этот бот нигде не запущен

*Blended Skill Talk

Чего пока не могут болталки

- Не врать ¯_(ツ)_/¯
- Использовать данные о собеседнике
  - но уже умеют пользоваться информацией «о себе»
- Удерживать длинный контекст
  - например, предыдущие беседы
- Вести разговор в определённом направлении
  - например, убеждать или объяснять

Чего ещё хочется от «полезного» ассистента

- Обучать новым задачам с минимальным числом примеров
- Вместе с примерами, хочется давать *схему* - описание интентов и слотов в свободной форме
- Уместная проактивность в решении задач
- Обучение на неявном фидбеке от юзеров

- Всё это в той или иной форме уже придумано,
  но пока не дозрело до продакшна

Ваши вопросики?

Можете писать мне на t.me/cointegrated

Спасибо за внимание!

Литература

- *Speech and Language Processing* by Jurafsky and Martin: фундаментальный учебник по NLP
- Набор постов про устройство Алисы:
  - https://habr.com/ru/company/yandex/blog/333912/ - про архитектуры "болталок"
  - https://habr.com/ru/company/yandex/blog/339638/ - очень в общих словах про Алису
  - https://habr.com/ru/company/yandex/blog/349372/ - более детально про кишки Алисы
- Статьи про крутые модели:
  - https://arxiv.org/abs/2005.14165 - статья про GPT-3
  - https://arxiv.org/abs/2004.13637 - статья про чатбот Фейсбука
  - https://arxiv.org/abs/2001.09977 - статья про чатбот Гугла

Создаём свою болталку за полчаса

Мастер-класс для самых упорных

по мотивам habr.com/ru/post/462333

Задача: болталка без нейронок

```
print(pipe.predict(['где ты живешь?']))
```

```
['я сейчас в библиотеке .']
```

```
print(pipe.predict(['ты любишь читать книги?']))
```

```
['не люблю .']
```

```
print(pipe.predict(['а что ты делаешь в библиотеке?']))
```

```
['футболку ищу . . .']
```

Идея: брать готовые ответы на вопросы

Загружаем в Python базу диалогов из фильмов

```
import pandas as pd # подгрузили библиотеку для работы с таблицами
good = pd.read_csv('good.tsv', sep='\t') # считали данные с диска
good.sample(3) # показали три случайные строки: контекст и ответ
```

	context_2	context_1	context_0	reply	label
710	NaN	NaN	цветы цветут летом	розовые цветы .	good
54283	NaN	NaN	я всё улажу .	да , и каким образом ?	good
25861	NaN	я знаю .	эшли сказала мне .	этого она мне не говорила .	good

Выбираем случайный из всех ответов на данный вопрос

```
rep = good[good.context_0 == 'как дела ?'].reply # отфильтровали ответы
if rep.shape[0] > 0: # убедились, что нашёлся хоть один ответ
    print(rep.sample(1).iloc[0]) # выбрали случайный и напечатали
```

```
отлично , а у тебя ?
```

Векторизуем тексты для нечеткого поиска

Превращаем тексты в числовые векторы

```
# импортируем sklearn - самую популярную библиотеку с машинным обучением
from sklearn.feature_extraction.text import TfidfVectorizer
# создаём объект, который будет преобразовывать короткие в числовые векторы
vectorizer = TfidfVectorizer()
# "обучаем" его на всех контекстах -> запоминаем частоту каждого слова
vectorizer.fit(good.context_0)
# записываем в матрицу, сколько раз каждое слово встречалось в каждом тексте
matrix_big = vectorizer.transform(good.context_0)
print(matrix_big.shape) # размер матрицы = число фраз * размер словаря
```

```
(60049, 14123)
```

Сократим размер, чтобы влезло в память

Сокращаем размерность с 14123 до всего 300 измерений

```
# импортируем алгоритм, известный как Метод главных компонент
from sklearn.decomposition import TruncatedSVD
# алгоритм будет проецировать данные в 300-мерное пространство
svd = TruncatedSVD(n_components=300)
# коэффициенты этого преобразования выучиваются так,
# чтобы сохранить максимум информации об исходной матрице
svd.fit(matrix_big)
matrix_small = svd.transform(matrix_big)
# в результате строк (наблюдений) столько же, столбцов меньше
print(matrix_small.shape)
# при этом сохранилось больше 40% исходной информации
print(svd.explained_variance_ratio_.sum())
```

```
(60049, 300)
0.4391518818760229
```

Выбираем вопрос из ближайших соседей

(1) Пишем класс для случайного выбора среди подходящих ответов

```
import numpy as np
from sklearn.neighbors import BallTree
from sklearn.base import BaseEstimator

def softmax(x):
    """ Функция для создания вероятностного распрделения """
    proba = np.exp(-x)
    return proba / sum(proba)

class NeighborSampler(BaseEstimator):
    """ Класс для случайного выбора одного из ближайших соседей """
    def __init__(self, k=5, temperature=1.0):
        self.k = k
        self.temperature = temperature
    def fit(self, X, y):
        self.tree_ = BallTree(X)
        self.y_ = np.array(y)
    def predict(self, X, random_state=None):
        distances, indices = self.tree_.query(X, return_distance=True, k=self.k)
        result = []
        for distance, index in zip(distances, indices):
            result.append(np.random.choice(index, p=softmax(distance * self.temperature)))
        return self.y_[result]
```

Собираем всё в один конвеер

(2) Соединяем векторизацию, сокращение размерности, и поиск соседей

```
from sklearn.pipeline import make_pipeline
ns = NeighborSampler()
ns.fit(matrix_small, good.reply)
pipe = make_pipeline(vectorizer, svd, ns)
```

```
print(pipe.predict(['где ты живешь?']))
```

```
['я сейчас в библиотеке .']
```

```
print(pipe.predict(['ты любишь читать книги?']))
```

```
['не люблю .']
```

```
print(pipe.predict(['а что ты делаешь в библиотеке?']))
```

```
['футболку ищу . . .']
```

Запускаем как бота в Телеграме

Запуск бота в Telegram

```
# эту ячейку нужно один раз запустить раскомменченной,
# чтобы установить библиотеку к вам. Потом снова закомментить.
# !pip install pytelegrambotapi
```

```
TOKEN = # токен нужно получить от @botfather, создав в нём собственного бота
```

```
import telebot

bot = telebot.TeleBot(TOKEN)

@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
    bot.reply_to(message, "Howdy, how are you doing?")

@bot.message_handler(func=lambda message: True)
def echo_all(message):
    bot.reply_to(message, pipe.predict([message.text.lower()])[0])

bot.polling()
```

Запускаем как навык в Алисе