МЕЖДУНАРОДНАЯ МОЛОДЕЖНАЯ НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ С ЭЛЕМЕНТАМИ НАУЧНОЙ ШКОЛЫ "ПРИКЛАДНАЯ МАТЕМАТИКА И ФУНДАМЕНТАЛЬНАЯ ИНФОРМАТИКА"

СОЗДАНИЕ МНОГОКАНАЛЬНОГО ЧАТ-БОТА С МОДЕЛЬЮ КЛАССИФИКАЦИИ НАМЕРЕНИЙ ПОЛЬЗОВАТЕЛЕЙ

Нелин Максим Андреевич,
Лонский Денис Олегович,
Крумина Ксения Васильевна

Омск 2023

Проблематика

Необходимо создать приложение чат-бот, которое будет отвечать на популярные вопросы абитуриентов заготовленными ответами. Сердцем приложения будет выступать модель intent recognition, а сам чат-бот должен быть доступен на нескольких платформах.

Для решения необходимо выполнить следующие задачи:

1. ***поиск и/или разметка данных для создания модели***
2. ***создание пайплайна для первичной обработки данных***
3. ***создание ансамбля нейронных сети прямого распространения***
4. ***введение метрик качества и быстродействия модели***
5. ***реализация многоканального чат-бота с полученной моделью для классификации намерений.***

Что по данным?

Имеем 12 намерений типа smalltalk
и 46 прикладных намерения

```
"budget_places": {
  "desc": "budget_places",
  "type": "product",
  "responses": [
    "На 2022 год количество бюджетных мест составляет: ",
    "49 для направления \"Фундаментальная информатика и информационные технологии\", ",
    "22 для направления \"Математическое обеспечение и администрирование информационных систем\"."
  ]
},
```

```
"what_are_you_doing": {
  "desc": "what_are_you_doing",
  "type": "smalltalk",
  "responses": [
    "Всё своё время я нахожусь тут и отвечаю на вопросы пользователей.",
    "У меня только одно дело – помогать пользователям."
  ]
```

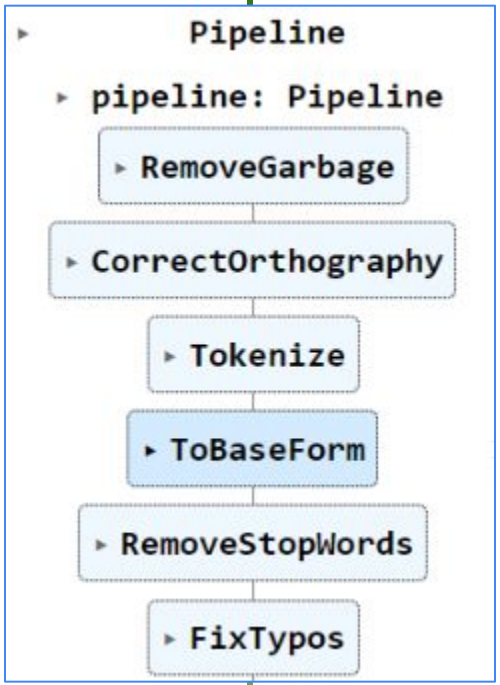
```
"documents": {
  "desc": "documents",
  "type": "product ",
  "responses": [
    "Подать документы можно:",
    "Лично в приемной комиссии по адресу: пр-т Мира, 11,  корпус 8, кабинет 115;",
    "Онлайн через личный кабинет на сайте ОмГТУ во вкладке «Абитуриенту»: https://www.omgtu.ru/entrant/zayavl.php",
    "С помощью сервиса «Поступление в вуз онлайн»: https://www.gosuslugi.ru/vuzonline"
  ]
```

```
{
  "intent": "budget_places",
  "desc": "budget_places",
  "patterns": [
    "Сколько бюджетных мест на направлениях?",
    "Какое количество мест?",
    "Какое количество бюджетных мест на специальности?",
    "Сколько бюджетных мест предоставляется на направление?",
    "Как распределяются бюджетные места на факультете?",
    "Какова доля бюджетных мест на специальности?",
    "Сколько бюджетных мест доступно для поступления на данный момент?",
    "Какое количество бюджетных мест на каждую специальность в этом году?",
    "Какие проценты бюджетных мест в этом году?",
    "Сколько бюджетных мест на специальность я могу получить?",
    "Какие критерии распределения бюджетных мест на направлениях?",
    "Сколько бюджетных мест на магистратуру?",
    "Сколько предоставляется мест?"
  ]
},
```

Что делать с этими данными?

Предобработать перед этапом выделения признаков

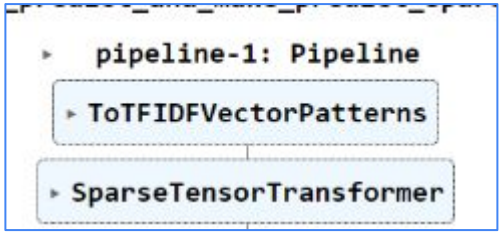
"Добрий типа деннь!"



["добрый", "день"]

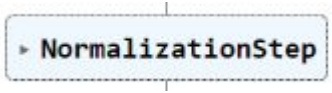
Архитектура модели

Feature extraction для разреженных признаков

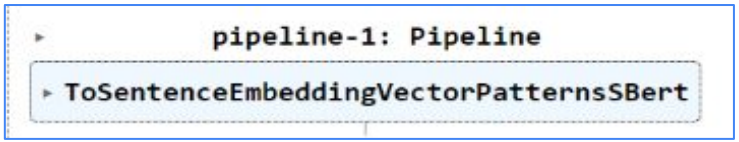


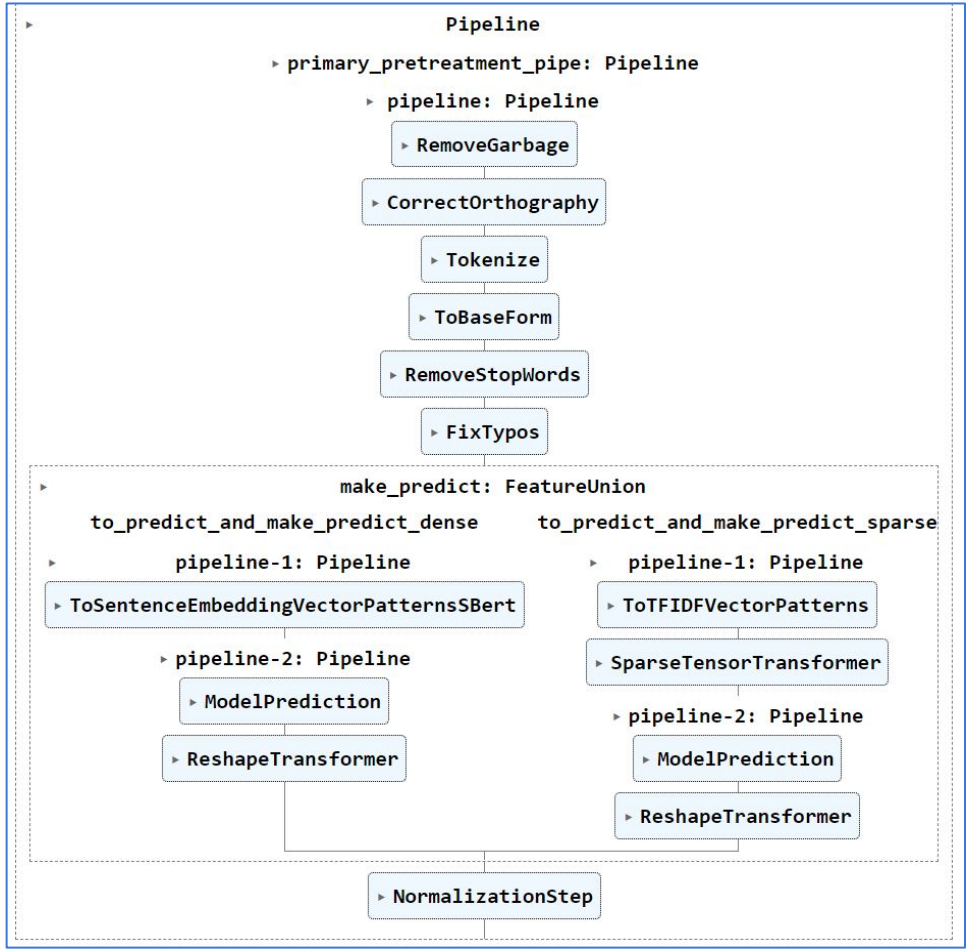
$V_{sum} = V_1 + V_2$

$V_{norm} = V_{sum} / || V_{sum} ||$

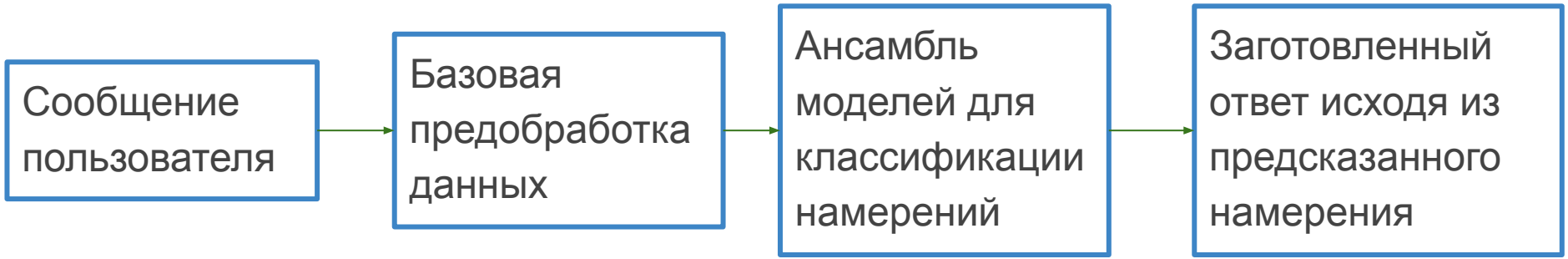


Feature extraction для полных признаков





Концепция модели



А какие метрики?

Можно выделить метрики быстродействия и точности модели

Метрики скорости работы, это время обучения модели и среднее время генерации предсказания для отправленной фразы. Среднее время обучения составляет всего **8** секунды. Среднее время предсказания составляет около **380 сотых** секунды, что является ощутимым временем.

Наиболее интересны для нас метрики micro average f1-score и weighted average f1-score. Разница у них в том, что вторая считается с учетом количества наблюдений, которые относятся к классу, а первая без учета. В нашей задаче все классы одинаково важны, в не зависимости от количества фраз в них, поэтому за ключевую метрику возьмем именно micro average f1-score.

micro average f1-score равен **0,95**

	precision	recall	f1-score	support
NONE	0.00	0.00	0.00	0
admission	1.00	1.00	1.00	9
budget_places	1.00	1.00	1.00	3
chair	1.00	1.00	1.00	4
commands	1.00	0.90	0.95	10
contacts_of_university	1.00	1.00	1.00	13
deadlines	1.00	1.00	1.00	10
disciplines	1.00	0.83	0.91	6
documents	1.00	1.00	1.00	13
exams	1.00	1.00	1.00	6
filthy_language	0.95	1.00	0.98	40
flagship	1.00	1.00	1.00	2
full-time	1.00	1.00	1.00	4
goodbye	1.00	0.98	0.99	60
greetings	1.00	1.00	1.00	33
have_a_question	1.00	1.00	1.00	6
hostel	1.00	1.00	1.00	9
how_are_you	1.00	0.94	0.97	16
military	1.00	1.00	1.00	9
name	1.00	1.00	1.00	5
paid_places	1.00	1.00	1.00	6
partners	1.00	0.78	0.88	9
pass_score	1.00	1.00	1.00	5
practice	1.00	1.00	1.00	10
professions	1.00	0.88	0.93	8
reviews	1.00	1.00	1.00	7
special_rights	1.00	1.00	1.00	7
teachers	1.00	0.71	0.83	14
thanks	0.90	1.00	0.95	9
university	1.00	1.00	1.00	5
university_location	1.00	1.00	1.00	9
what_are_you_doing	1.00	1.00	1.00	5
what_you_can_do	1.00	0.83	0.91	6
who_can_help	1.00	1.00	1.00	6
accuracy			0.97	364
macro avg	0.97	0.94	0.95	364
weighted avg	0.99	0.97	0.98	364

Какую модель используем?

Используем ансамбль нейронных сетей прямого распространения

Модель для полных признаков

```
model = Sequential()

model.add(Input(shape=(INPUT_SHAPE,), sparse=True))
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Dense(OUTPUT_SHAPE, activation='softmax'))

early_stop = EarlyStopping(monitor='loss',
                           min_delta=0.05,
                           patience=10,
                           restore_best_weights=True)
optimizer  = SGD(learning_rate=0.01, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical_crossentropy',
                                  optimizer=optimizer,
                                 metrics=['accuracy'])

history_training = model.fit(dense_tensor_train,
                           train_y, epochs=100,
                           batch_size=25,
                           callbacks=[early_stop])
```

Модель для разреженных признаков

```
model = Sequential()

model.add(Input(shape=(INPUT_SHAPE,), sparse=True))
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))
model.add(Dense(OUTPUT_SHAPE, activation='softmax'))

early_stop = EarlyStopping(monitor='loss', min_delta=0.05, patience=10,
restore_best_weights=True)
optimizer  = SGD(learning_rate=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=optimizer ,
metrics=['accuracy'])

history_training = model.fit(sparse_tensor_train, train_y, epochs=100,
batch_size=25, callbacks=[early_stop])
```

Схематическое изображение


```python
class IBot(ABC):
    __chat_bot = None

    def __init__(self, api_key: str, platform) -> None:
        """
        Initialize IBot class with `logger` and authorize using `api_key`.
        :param api_key: api key для получения доступу для получения и отправки сообщений.
        :param logger: custom class responsible for logging.
        """
        self.__chat_bot = ChatBot(platform)
        self.__authorization(api_key)

    @abstractmethod
    def start(self):
        pass

    @abstractmethod
    def __authorization(self, api_key: str) -> None:
        pass

    @abstractmethod
    def __handle_start(self):
        pass
```

```python
class BotVk(IBot):
    _IBot__chat_bot = None
    _IBot__logger = None
    __vk_api_access = None
    __long_pool = None

    def start(self):
        try:
            for event in self.__long_pool.listen():
                if event.type == VkEventType.MESSAGE_NEW and event.to_me and event.text
                    message_text = event.text
                    receiver_user_id = event.user_id
                    if message_text in ("/start", "!start"):
                        self._IBot__handle_start(receiver_user_id)
                    elif message_text in ("/help", "!help"):
                        self._IBot__handle_help(receiver_user_id)
                    else:
                        self._IBot__handle_text(message_text, receiver_user_id)
        except requests.exceptions.ReadTimeout:
            print("\n Переподключение к серверам ВК \n")
            time.sleep(20)
```

```python
class BotTelegram(IBot):
    _IBot__chat_bot = None
    _IBot__logger = None
    __bot = None
    __dp = None

    def __init__(self, api_key: str, platform):
        super().__init__(api_key, platform)

    def start(self):
        loop = asyncio.new_event_loop()  # создаем новый цикл событ
        asyncio.set_event_loop(loop)  # устанавливаем его в качеств
        loop.create_task(executor.start_polling(self.__dp))  # запу
        loop.run_forever()  # запускаем бесконечный цикл событий

    def _IBot__authorization(self, api_key: str):
        self.__bot = Bot(api_key)
        self.__dp = Dispatcher(self.__bot)
        self.__register_handlers()
```

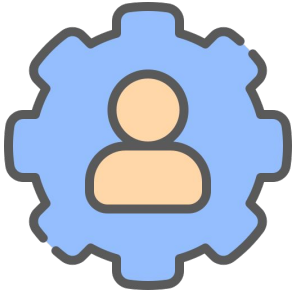
```python
class BotDS(IBot):
    _IBot__chat_bot = None
    _IBot__logger = None

    def __init__(self, api_key: str, platform):
        super().__init__(api_key, platform)
        intent = discord.Intents.default()
        intent.members = True
        intent.message_content = True
        self.client = discord.Client(intents=intent)
        self.client.event(self.on_message)
        self.api_key = api_key

    def start(self):
        self.client.run(self.api_key)
```

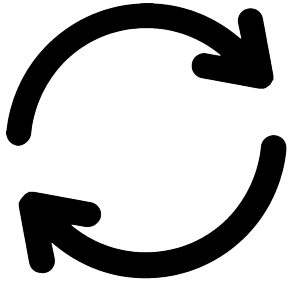
Какие преимущества многоканальной коммуникации?

Упрощение масштабирования

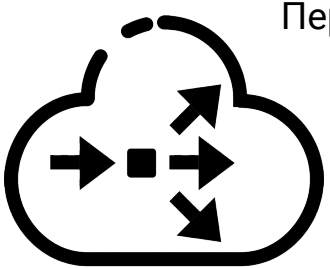


Персонализация

Сбор и анализ данных



Унификация обновлений



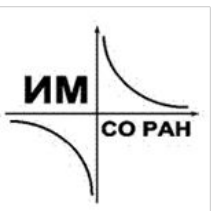
Экономия ресурсов и распределение нагрузки

Многоканальная коммуникация

Единое управление

Список источников

1. Хобсон Л., Ханнес Х., Коул Х. Обработка естественного языка в действии – Санкт-Петербург: Питер, 2020 – 57 с.
2. Бенгфорт Б., Билбро Р., Охеда Т. Прикладной анализ текстовых данных на Python – Санкт-Петербург: Питер, 2019 – 368 с.
3. NLP. Основы. Техники. Саморазвитие. Часть 1 [Электронный ресурс]: URL: https://habr.com/ru/company/abbyy/blog/437008/ (дата обращения: 15.11.2021).
4. Как решить 90% задач NLP: пошаговое руководство по обработке естественного языка [Электронный ресурс]: URL: https://habr.com/ru/company/oleg-bunin/blog/352614/ (дата обращения: 15.11.2021).
5. Основы Natural Language Processing для текста [Электронный ресурс]: URL: https://habr.com/ru/company/Voximplant/blog/446738/ (дата обращения: 15.1.2021).
6. NLP: разбираем на пальцах практические кейсы без заморочек с ML [Электронный ресурс]: URL: https://habr.com/ru/post/557358/ (дата обращения: 15.11.2021).



МЕЖДУНАРОДНАЯ МОЛОДЕЖНАЯ НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ С ЭЛЕМЕНТАМИ НАУЧНОЙ ШКОЛЫ "ПРИКЛАДНАЯ МАТЕМАТИКА И ФУНДАМЕНТАЛЬНАЯ ИНФОРМАТИКА"

СОЗДАНИЕ МНОГОКАНАЛЬНОГО ЧАТ-БОТА С МОДЕЛЬЮ КЛАССИФИКАЦИИ НАМЕРЕНИЙ ПОЛЬЗОВАТЕЛЕЙ

Нелин Максим Андреевич,
Лонский Денис Олегович,
Крумина Ксения Васильевна

Омск 2023