

Рабочий лист №1

Дата "01" февраля 2025 г.
(заполняется оргкомитетом)

Шифр ПИ-19
(заполняется оргкомитетом)

Оценка работы

(таблица заполняется по итогам проверки работы членами жюри олимпиады)

№ задания	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Итого (итоговый балл, подпись председателя жюри)
Балл																88
№ задания	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Балл																93

Микросервисы

(название олимпиады, заполняется участником)

Прикладная информатика

(профиль олимпиады, заполняется участником)

① Используем микросервисную архитектуру, что позволяет реализовать функциональность на независимых компонентах, которые можно масштабировать
основные компоненты:

1. Исходные данные (микросервисы оператора, системы жалоб звонков, бд с информацией о подразделениях компаний)

2. Компоненты системы:

Data Ingestion Service (сбор данных)

Data Storage (хранение данных)

Audio Processing Service (обработка аудио)

Notification Service (уведомления)

API Gateway (Единая точка входа для внешних систем)

Monitoring and Logging (мониторинг и логирование)

3. Используем технологии:

ЯП: Python + фреймворк Django

БД: PostgreSQL (для структурированных данных)

MongoDB (для неструктурированных данных)

Хранение аудио ф. Amazon S3

сдачу: 3 листа

Очередь сообщений: Apache kafka

Аннотация данных: Apache kafka

Расшифровка аудио: Google speech to text

Контейнеризация: Docker

~~Данные~~ Мониторинг и логирование (Prometheus)

Data Ingestion Service. Задача: Получение данных из внешних источников (телефонная оператор, системы защиты железной дороги)

Функционал: Подготовили API телефонной оператор, загрузке аудиопотока и мета-данных.

Audio Processing Service. Задача: обработка аудиопотока (конвертирование аудио в формат пригодный для расшифровки)

Transcription Service. Задача: расшифровка аудио в текст.

Функционал: использование модели speech-to-text для преобразования аудио в текст, сохранение расшифровки в БД

② Функция определения

шаг 1. Предобработка текста
очищенный текст = предобработать текст (Текст)

шаг 2. Поиск ключевых фраз

ключевая фраза = ['фраза']

найденная фраза = найти ключевые фразы (очищенный текст, ключевые фразы)

3. Оценка вероятности появления слов на основе ключевых фраз

вероятность по фразам = Оценка вероятности по фразам (найденные фразы)

классификация текста с использованием модели МО

вероятность по модели = классифицировать текст (очищенный текст)

5. Комбинирование результатов

общая вероятность = комбинировать = комбинировать вероятности (вероятности по фразам, вероятности по модели)

Дополнительный рабочий лист
(без рабочего листа №1 недействителен)

Дата "01" февраля 2028 г.
(заполняется участником)

Шифр 121 - 19
(заполняется участником)

Возврат результатов
Вернуть общую вероятность

Функции ~~for~~ преобразования текста (текст):

приведение текста к нижнему регистру
текст = текст.lower()

Удаление пунктуации и ~~и~~ специальных символов
текст = удалить_пунктуацию (текст)

Удаление стоп-слов (например, "и", "на" и т.д.)
текст = удалить_стоп_слова (текст)

приведение слов к нормальной форме
текст = привести_к_нормальной_форме (текст)
return текст

Функции найти ключевые фразы (текст, ключевые фразы)

найденое_фраза = []

для каждой фразы в ключевых фразах:

Если фраза содержится в тексте:

Добавить фразу в найденные фразы

return найденные_фразы

Функции Оценка вероятностей по фразам (найденные фразы):

Если длина найденных фраз > 0

Вернуть 0.8 # высокая вер-т нахождения слов

Else:
Вернуть 0.1 # низкая вер-т нахождения слов

Функции классифицировать текст (текст)

загрузка предобученной модели МО

модель = загрузить_модель ('train_classification_model.pkl')

векторизация текста (преобразовать текст в числовой формат)

вектор = векторизировать_текст (текст)

предлагаемые вероятности возможных классов

вероятности = модель.предсказать (вектор)

return вероятности

Функция комбинировать вероятности (вероятности по фразе, вероятности по модели):

Простое усреднение вероятности

общие_вероятности = (вероятности по фразе + вероятности по модели) / 2

return общие_вероятности

Описание: 1. Преобразование текста (текст приводится к нижнему регистру, удаляются пунктуация и стоп-слова, выполняется приведение слов к начальной форме)

2. Поиск ключевых фраз (алгоритм ищет в тексте ключевые фразы, характерные для эмоциональных звуков)

3. Оценка вероятности по ключевым фразам (если найдены ключевые фразы, вероятность возможных классов повышается)

4. Классификация текста с использованием МО

5. Комбинирование результатов (вероятности, полученные из обеих ключевых фраз и модели МО объединяются)

6. Возврат результата. (Алгоритм возвращает общую вероятность возможных классов)



Дополнительный рабочий лист
(без рабочего листа №1 недействителен)

Дата "01" / февраль / 2025 г.
(заполняется участником)

Шифр ПМ-19
(заполняется участником)

```
import re

def extract_personal_info(text):

    passport_pattern = r'\d{4}[5?]\d{6}\d' # паспортные данные
    snils_pattern = r'\d{3} - \d{3} \d{2} \d{2} \d{2}' # снилс
    phone_pattern = r'\+7\d{3}(\d{3}|\d{4})? \d{3} - \d{2} \d{2} \d{2}'

    # можно еще добавить

    passports = re.findall(passport_pattern, text)
    snils = re.findall(snils_pattern, text)
    phones = re.findall(phone_pattern, text)
    # так как мы собираем данные из разных источников, номер банк. карты и т.д.
    return {

        "паспортные данные": passports,
        "снилс": snils,
        "номер телефона": phones

    }

# пример использования
text = "Текст с данными"
result = extract_personal_info(text)

for key, value in result.items():
    print(f"{key}: {value}")

Объяснение: # passport pattern и остальные из других сервисов и
номер паспорта, снилс, номер телефона, + и код кода добавить
номер банковской карты и другую банковскую информацию
Функция extract_personal_info принимает текст и
```

использует регулярные выражения для поиска личной информации.

Возвращает словарь с найденными данными.

① Дополнения.

Принципы взаимодействия компонентов

1. Асинхронная коммуникация (компоненты обмениваются данными через очередь сообщений, что обеспечивает масштабируемость и
2. REST API (внешние системы взаимодействуют с API Gateway, который перенаправляет запросы к соответствующим микросервисам)
3. Event-Driven архитектура (события инициируют выполнение задач в других компонентах)
4. Масштабируемость (каждый микросервис может масштабироваться независимо)
5. Безопасность (использование HTTPS для аутентификации и авторизации, шифрование данных при передаче и хранении).