

Рабочий лист №1

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

Шифр ПИ-10  
(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																95
№ задания	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Балл																93

Математика

(название олимпиады, заполняется участником)

Применение информационных технологий

(профиль олимпиады, заполняется участником)

1) Для проектирования системы для выявления тематических возможностей будущей использовать микросервисную архитектуру, чтобы разбить систему на независимые компоненты, которые будут взаимодействовать друг с другом через API или сообщения.

Основные компоненты:

- 1) Компонент сбора данных - получение данных из внешних источников (звонки, звонков). Для интеграции с внешними системами можно использовать REST API.
- 2) Компонент хранения данных - хранение записей о тематических разговорах, расшифровка звонков. Можно использовать СУБД (MySQL, PostgreSQL).
- 3) Компонент обработки аудио - преобразование записей разговора в текст (расшифровка). Будем использовать Whisper, OpenAI или можем создать собственную модель на базе Pytorch.
- 4) Компонент анализа данных - анализ расшифровки звонков для выявления тематических возможностей. Используем NLP и машинное обучение.
- 5) Компонент уведомлений - получение уведомлений при обнаружении потенциальных звонков. Для получения уведомлений через e-mail или SMS.

6) Импонент, исторична и шифровка - отсылание работы системы (сбор итогов, метрики). Планы: Графа.

Принципы взаимодействия импонентов: использование очереди (Касса) для передачи данных между импонентами; каждый импонент имеет масштабируемость независимо; использование результирующего копирования; шифрование данных при передаче, хранении; ~~защита~~

Импонент сбора данных получает звонок от оператора связи, аудио передается в систему для расшифровки. Импонент анализирует данные, проверяет текст на наличие подозрительных фраз, если звонок признан подозрительным, система отправляет уведомление оператору или клиенту. Все действия шифруются и собираются метрики.

Импоненты аудио программирования: Python / Java / Go

2) Для определения вероятности совершения мошенничества по текстовой расшифровке телефонного разговора будем использовать алгоритм обработки на базе естественного языка и машинного обучения.

1) Загрузить модель машинного обучения

~~скачать модель~~

- загрузить модель

- загрузить список ключевых фраз

2) Получить расшифровку телефонного звонка (text)

3) ~~Обработка~~ текста для обработки

Подготовка

text = удалить пробелы и привести к нижнему регистру

4) Анализ текста по ключевым фразам

finded - phrase = найти - ключевые - фразы (text)

5) Векторизовать текст

text - vector = векторизовать (text)

# разделим на  
классы 0 - не мошенник  
1 - мошенник

6) Машинное обучение текста

pred = модель - машинное обучение. предсказать (text - vector)

7) Анализ текста и фраз, факторов

Если длина разговора < минимальная длина

увеличить pred[i] (короткие разговоры не мошенники)

Если обнаружены слова, связанные (подозрительны)  
увеличить pred[i]

8) Числовая вероятность

Если pred[i] < пороговое - значение

вернуть "Высокая вероятность мошенничества"

Иначе

вернуть "Низкая вероятность мошенничества"

Использование

1) Данные загружаются данные (фразы и модели)

2) Получение текста (текстовая расшифровка)

Дополнительный рабочий лист  
(без рабочего листа №1 недействителен)

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

Шифр ПИ-10  
(заполняется участником)

- 3) Предобработка текста. Текст очищается от лишних символов и приводится к единому формату.
- 4) Анализ полнотекстовой фразы.
- 5) Векторизация и кодирование. Оценка вероятности того, является ли фразы мошеннической.
- 6) Учет различных фразировок.
- 7) Определение итоговой вероятности.

3) import re

```
def extract_info(text):  
    passport = r'\b\d{4}\s\d{6}\b' # 4 цифры серия  
    snils = r'\b\d{3}-\d{3}-\d{3}\s\d{8}\b' # номер  
    credit_card = r'\b\d{4}\s\d{4}\s\d{4}\s\d{4}\s\d{4}\b'  
    eve_evv = r'\b\d{3}\b'
```

```
    passports = re.findall(passport, text)  
    snilses = re.findall(snils, text)  
    cards = re.findall(credit_card, text)  
    eves = re.findall(eve_evv, text)
```

```
    # Проверяем, что eve ког идет после номера карты  
    filter_eve = []
```

```
    for eve in eves:  
        if any(eve in card[-3:] for card in cards):  
            filter_eve.append(eve)
```

```
    return passports, snilses, cards, filter_eve
```

```
text = "Имя: мой паспорт 1234 567890, СНИЛС  
123-456-789 ## 01, номер карты: 1234 5678  
9012 3456, eve ког: 123."
```

```
extract_info(text)
```

```
passport, snils, card_number, eve = extract_info(text)
```

```
print("Паспорт:", passport, "СНИЛС:", snils, "номер карты:",  
card_number, "ког безопасности:", eve)
```