

Рабочий лист №1

Дата "1" февраля 2025 г.
(заполняется оргкомитетом)

Шифр ПЦ-45
(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																ПЗ

Магистры Уи

(название олимпиады, заполняется участником)

Трикарная информатика

(профиль олимпиады, заполняется участником)

Задача 1

Предлагается спроектировать архитектуру пропр. системы для выявления мошенничества. Упор делается на краевые данные о контакте/тел. разговоре с предполож. мошенником. Допустим, это будет база данных. Предполагается работа с внешними источниками данных. То есть нам нужно какое-то соединение по сети с этими источниками, API. Нужна также система обработки тел. звонков, возможно, подключение внешних библиотек или создание своего распознавателя, нейросети, для транскрибации. Также следует прокрасить анализ ключевых слов, их количество.

В кону предложить архитектуру разрабатываемой системы

система для языка программирования Java. Нашим любимой технологией на языке Java предоставляет огромный спектр возможностей командной разработки ПО любой сложности. (Аналоги проектируют система масштабируемости, а я, как будущий разработчик Java, сделаю это за 2 часа!)

Любая система, разрабатываемая для использования, подразумевает какое-то количество нагрузки на систему. Для того, чтобы наш сервер не "упал", нужно будет обеспечить стабильное представление между запросами, ответами. С этим прекрасно справится брокер сообщений Apache Kafka. Эта технология отправляет все запросы в очередь запросов и по мере ответов от сервера возвращает ответ клиенту. Я реализовала собственный брокер сообщений на основе технологии Kafka, так что глубоко погружаюсь в тему масштабируемости, распределения запросов.

Также я предпочитаю использовать в системе фреймворк Spring Boot для создания REST API или Web-socket серверов. Эта система Spring позволяет разработчику сконцентрироваться на обработке информации, внешним видом приложения, не заботясь о настройках системы, портов, зависимостей. Все это Spring сам поднимает, настраивает пути и т.п. На данный момент

Дополнительный рабочий лист
(без рабочего листа №1 недействителен)

Дата "1" февраля 2025 г.
(заполняется участником)

Шифр ИИ-45
(заполняется участником)

Spring занимает 1 место в технологиях для разработки клиент-серверных приложений.

Spring Integration использ. для обработки потоков, а Kafka Connect для интеграции с внешними данными.

Для хранения данных в Java самой популярной фреймворк это Hibernate для работа с реляционными БД. Hibernate позволяет создавать в коде структуру таблицы, которое Spring потом интегрирует на сервер БД. Hibernate оперирует аннотациями (@), кросс-вызываемыми. Hibernate ~~создает~~ — ORM — object relationship mapping. (как раз то, что и создает структуру в коде). А сами разработчики своего ORM на Java, так что у меня есть полное понимание, как работает Hibernate внутри! Также в Spring есть Spring Data JPA, но сейчас я не работаю. JPA — ~~это~~ правила, которые используются в Hibernate, однако Hibernate пошлось раньше JPA, JPA взяли за основу разработки Hibernate.

Для самой БД удобнее всего использовать

PostgreSQL или MongoDB. Выберем первое. Я работала и с тем и с другим, у PostgreSQL есть удобный интерфейс pgAdmin. Там можно в рамках времени сделать загрузку, импорт данных. Я знаю, что есть БД, которые хранят аудиозаписи. У нас будет храниться транскрибирование, чтобы экономить ~~мем.~~ память.

Для транскрибирования есть хуга сервисов, есть даже Java Sound API. Или можно использовать разработки Индекс для распознавания русской речи.

Для анализа данных будем использовать технологию machine Learning для классификации слов, текста. Можно реализовать свой классификатор (уже по тексту). Обучить свою нейросеть. Или же интегрироваться с внешними сервисами.

Когда система поймет, что звонит мошенник (в телем разговора) предполагается уведомление клиента об этом через почту (Java Mail API), или через отправку сообщения.

Чтобы создать отчет, для просмотра, будем использовать Spring Web, HTML, CSS, JS, Thymeleaf для визуализации данных ~~на~~ в браузере. Они (технологии) используются для верстки стр. и связи с сервером и клиентом)

Дополнительный рабочий лист
(без рабочего листа №1 недействителен)

Дата " 1 " февраля 20 25 г.
(заполняется участником)

Шифр 11-45
(заполняется участником)

Все компоненты взаимодействуют друг с другом асинхронно через Kafka и синхронно через REST API события (напр, получение нового звонка) инициируют запуск нужных компонентов. Каждый компонент может масштабироваться в соответствии с системой и количеством клиентов.

Задача 2:

У нас есть текстовая расшифровка text, список ключевых слов keywords, напр. "скажите кор", "скажите парю" и т.п. Но нам будем определять тип звонка. Нам нужен порог вероятности для того, считать ли звонок мошенническим или нет. Тревожение порога говорит о том, что звонок подозрительный. (threshold)

Ваш алгоритм:

- 1) Убираем из text только слова: убираем все знаки препинания. ~~Посчитаем~~ ~~какую-то~~
- 2) Посчитаем, сколько вхождений каждого ключевого слова есть в тексте
- 3) Посчитаем вероятность отклонения количества найденных ключевых слов к общему количеству ключевых слов в тексте

4) Если вероятность превышает нам threshold, будем говорить, переформулировка или нет. Вернем вероятность и информацию.

исходно:

еще и в функцию решить перевод

```
clean_text = deleteMarks(text)
```

создадим словарь типа {слово: ..., кол-во встреч: ...}

```
word_counts = Counter(words) # из Python
```

probability = $\frac{\text{кол-во встречных слов среди тех из которых}}{\text{кол-во всех встречных слов любого типа}}$

посчитаем до этой строки, и не знай

```
if probability > threshold:
```

```
    is_moment = True
```

```
else:
```

```
    is_moment = False
```

```
return probability, is_moment
```

```
print("с вероятностью", probability, "звонок от моментика")
```

~~probability = {word: ...}~~ Это можно сделать на Python с помощью регулярных выражений, легкую работу со словарями и подсчетами, списковыми включениями.

Можно улучшить алгоритм путем контекста (поиск словосочетаний), добавлением всех слов, обучение модели классификации на данных с помощью машинного обучения.

Дополнительный рабочий лист
(без рабочего листа №1 недействителен)

Дата " 1 " февраля 2025 г.
(заполняется участником)

Шифр ИИ-45
(заполняется участником)

Задача 3:

Для увеличения информации из текстовой расшифровки ~~нужно~~ лучше использовать Python, хотя ранее и проектировалась система на Java, т.к. можно встроить Python распознавание, хотя на Java тоже есть регулярные выражения. Сейчас удобнее сделать это на Python.

предполагается след структура данных:

1) когда бот или сотрудник (то же самое верно, потому что тогда он бот сам записывал себе в чужую или информацию о клиенте), будет задан соотв. вопрос о (напр) серии и номер паспорта.

Человек говорит: 1234 567890

- паспорт →

- индекс, напр. 123-456-789 01

- номер банковской карты (16 или 15 цифр):

1234 5678 9012 3456

- CVV/CVV-код: 123 или 1234

На Python через регулярные выражения найдём эти слова. Допустим, наши нейросети прекрасно расшифровали данные с аудиозаписи, представив цифры не "одни", а "1" и т.д., тогда как "1"

Или же "тире" как "-"; подчеркните под
каждой буквой кор регулярного выражения.

```
import re # библиотека пер. бор. на Python.
```

```
def get_personal_info_from_transkrib(text):
```

заданная регуляризатор для поиска данных в тексте.

passport_pattern = r'\b\d{23}\s?\d{23}\s?\d{63}\b'

грануло-
инва

2 члора

24.09.20

визмор

Ошш +
вх оне решии

snils-pattern = r\b\d{3}-\d{3}-\d{3}\s?\d{2}\b

card_number_pattern = "\b\d{4}\s?\d{4}\s?\d{4}\s?"

1d431b'

используются одни и те же символы в рег. вар.

evc-evw-pattern = n\ b\ d { 3, 4 } b

3 мм 4. Вокруг.

ищем все совпадения в тексте

passport_matches = re.findall(passport_pattern, text)

`snippets_matches = re.findall(snippets_pattern, text)`

card_number_matches = re.findall(card_number_pattern,

```
evc_cvv_matches = re.findall(evc_cvv_pattern, text)
```

нужно убедиться, что cnc/cni получено

~~в какой~~ # после номера карты, и не будет

случае с другой посредственностью цифр.

cvc_cvv_matches = [code for code in cvc_cvv_matches
if len(code) == 5]

if len(word) in [3, 4]

что можно сделать для оптимизации
этого свс идет после слова "бета/человека" "CVC"

или "CVV", тогда его реп. выражение будет:

cvc_pattern = r'(?:cvc|cvv|kop)\s?\.\s?\.\s?\.\s?b(1d{3,4})'

161

не сохр в рез поиска группировки

Дополнительный рабочий лист
(без рабочего листа №1 недействителен)

Дата "1" февраля 2025 г.
(заполняется участником)

Шифр ПУ-45
(заполняется участником)

```
return { , passport': passport_matches,  
        , snils': snils_matches,  
        , card-number': card-number_matches,  
        , evc-evv': evc-evv_matches
```

Выводить в main, напр, `print (person_in fo I, passport)`

Для предложенной структура данных код
будет работать и возвращать крат. информацию.