

Рабочий лист №1

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

Шифр ПИ-37
 (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																97
№ задания	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Балл																93

Магистратура

(название олимпиады, заполняется участником)

Применение информационных технологий

(профиль олимпиады, заполняется участником)

Задача 1. Разрабатываемая программная система представляет из себя сервис для внешнего мониторинга показателей, основанный на клиент-серверной архитектуре и предполагающий пользование как обычными людьми, так и умными устройствами, например смартфонами. Система состоит из нескольких компонентов, обеспечивающих ее работу. Рассмотрим архитектуру разрабатываемого сервиса с его основными компонентами, используемыми технологиями и описанием устройств и взаимодействий между собой.

Первым и, пожалуй, одним из основных компонентов системы является сервер для анализа. Основными данными будут являться внешние API (application programming interface) различных компаний операторов, базы данных телекоммуникационных звонков, в частности уже агрегированные и агрегируемые или агрегируемые. Предполагается реализовать модуль для интеграции с периферийными устройствами через REST API, веб-сервис для получения информации о звонках и, возможно, их расшифровки, если сторонние ресурсы будут предоставлять такие данные, однако подобный модуль будет в системе, но обо всем по порядку.

Вторым важным компонентом системы является хранение данных. Так как для качественного анализа, внешнего статистического и внутреннего мониторинга и прогнозирования, являются ли звонки монотонными, необходима работа с большими объемами данных, то всю эту информацию нужно централизованно хранить и обрабатывать. Для пользовательских данных и информации о звонках отлично подойдет структурированное хранение с помощью реляционной базы данных, которое позволяет оптимизировать данные и выполнять информацию с помощью запросов на языке SQL. Другим решением является система управления базами данных PostgreSQL, которая отлично подойдет под указанные задачи. Для хранения неструктурированных данных, например, расшифровки звон-

нов, на мой взгляд, лучшим выбором будет использование неструктурированной базы данных (NoSQL), например, MongoDB, которая осуществляет хранение данных в формате JSON, что особенно удобно при использовании веб-механизмов. Но на этом применение неструктурированной базы данных не заканчивается. Там же, в разрабатываемой системе предполагается наличие пользовательского веб-интерфейса, но хорошим решением будет использование базы данных "инст-значение" Redis, осуществляющей хранение данных в оперативной памяти, что позволяет мгновенно получать и иметь доступ. Она будет использоваться для имитирования и обработки сессии.

Следующим немаловажным компонентом системы является модуль для обработки данных. Для обеспечения ETL-процесса (Extract, Transform, Load), т.е. извлечения, трансформации и загрузки данных предполагается работа с потоковыми данными, помочь с чем может Kafka. Дополнительно, и, наверное, наиболее важным модулем системы будет модуль для анализа и обработки расшифрованных записей. Для его реализации предполагается использование алгоритмов из подкласса искусственного интеллекта - обработки естественного языка. Основным языком программирования для этого модуля, конечно, будет являться Python с его обширным набором библиотек, в частности scapy и NLTK.

Еще одним важным компонентом системы является модуль анализа и внешнего мониторинга. Для решения этой задачи предполагается снова прибегнуть к искусственному интеллекту и использовать алгоритм машинного обучения для построения моделей, выявляющих подозрительные поведение на основе имеющихся данных. Здесь могут пригодиться различные методы машинного обучения, например, обучение с учителем, в частности задачи классификации и регрессии и использование для их решения алгоритмов: логистическое решение, деревья решений, нейронные сети; или обучение без учителя - задачи кластеризации и т.д. Реализация данного модуля нам не будет приходиться на язык Python (используя различные библиотеки библиотек и фреймворков, такие как NumPy, Tensorflow, Keras, scikit-learn и т.д.).

Следующим важным компонентом, уже упомянутым ранее, является пользовательский интерфейс. Предполагается создание интерфейса для анализа и операторов системы поддержки, позволяющего просматривать данные о записях, а также для базовых пользователей, позволяющего управлять функцией анализа и получать уведомления о подозрительных действиях. Для реализации предполагается использование фреймворков JavaScript, таких как React.js или Angular, предоставляющих компонентный подход и позволяющий переиспользовать код. Реализация серверной части веб-приложения может быть произведена на том же языке JS, благодаря платформе Node.js.

Немаловажным является не только использование системы, но и мониторинг ее состояния. Для этого предполагается применение системы мониторинга Prometheus, которая будет считывать основные метрики системы, в сборе с сервисом Grafana, который будет предоставлять визуальное представление состояния системы.

Таким образом, данная архитектура позволяет системе эффективно выявлять потенциальные угрозы благодаря наличию различных компонентов. Компонент сбора данных периодически опрашивает внешние источники и загружает информацию в хранилище данных, куда не попадают и загруженные пользователями записи и расшифровки. Далее ETL-процесс извлекает данные из хранилища, обрабатывает их и загружает обратно в базу данных. Модуль анализа получает данные о записях из хранилища, применяет модели машинного обучения и генерирует отчеты о подозрительных действиях. Веб-интерфейс запрашивает данные из API, отображает

Шифр ПК-37
(заполняется участником)

Лист № 2
(заполняется участником)

Преем

- Обработка данных

- Очистка текста:

- удалить лишние символы (пробелы и tab, \n)
- привести текст к нижнему регистру (lowercase)
- удалить знаки препинания (.,?! и т.д.)
- Tokenization, т.е. разбиение текста на отдельные слова - токены
- привести слова к базовой форме (используя spaCy)
- удалить распространённые стоп-слова, не несущие смысла (частицы и, а, не, и т.д.)

- Извлечение признаков

- Частотный анализ

- Создать словарь уникальных слов и фраз, связанных с тематическим вектором ("перевод денег", "работорговля" и т.д.)
- посчитать частоту встречаемости каждого уникального слова (фразы)

- Анализ структуры и триграмм

- Создать n-граммы из токенизированного текста.
- проверить наличие n-грамм, связанных с тематическим вектором

- Анализ тематичности

- исп. библиотеку для анализа тематичности TextBlob
- извлечь тему тематичности

- Извлечение сущностей (NER)

- исп. библиотеку NER, например spaCy для извлечения имен, мест, дат и т.д.

- проверить, есть ли среди извлеченных сущностей подозрительные фразы (фреймворк оптимизации)

- Проверка на наличие URL-адресов и телеграмм

- Формирование вектор признаков

- объединить все извлеченные признаки в один вектор, нормализуя

- Предназначенный вектор тематичности

- Исп. обученную модель машинного обучения (например, логистический регрессор). Модель должна быть предварительно обучена на базовых данных, содержащих расшифрованные звонки и соответствующий им метки (тематический регрессор или нет)
- Ввести вектор признаков в обученную модель
- Получить предсказание вектора тематичности

- Возврат результатов

- получение вектора тематичности r