### **Техническое задание на разработку алгоритма обработки данных**

**Цель проекта:** Создание гибкой, оптимизированной и масштабируемой программы для обработки сырых данных (полученных с парсинга RSS-фидов, Telegram-каналов) с использованием методов NLP, кластеризации и векторных эмбеддингов.

Ожидаемый результат: структурированная база данных с категориями, тегами, ключевыми словами и метаданными для аналитики.

#### **1. Требования к функционалу**

**1.1. Входные данные:**

* Сырые данные в формате JSON/CSV, собранные парсером (тексты, метаданные: источник, дата, автор и т.д.)
* Поддержка новых типов источников в будущем (архитектура должна быть расширяемой)

**1.2. Этапы обработки:**

* **Предобработка текста:**
	+ Очистка от HTML-тегов, эмодзи, стоп-слов, спецсимволов
	+ Токенизация с учетом ограничений по количеству токенов (настраиваемый параметр)
	+ Лемматизация/стемминг (выбор языка: русский/английский/китайский)
* **NLP-анализ:**
	+ Извлечение сущностей (имена, даты, локации, организации, и все возможное)
	+ Автоматическая генерация тегов и ключевых слов (определить лучшие алгоритмы, например: TF-IDF, YAKE, KeyBERT)
	+ Определение тональности текста
* **Кластеризация:**
	+ Группировка текстов по темам (определить лучшие алгоритмы, например: K-means, DBSCAN, HDBSCAN)
	+ Визуализация кластеров (опционально, например, t-SNE)
* **Векторизация:**
	+ Генерация эмбеддингов (определить лучшие модели, например: Word2Vec, BERT, FastText)
	+ Сохранение векторов для дальнейшего анализа
* **Структурирование данных:**
	+ Формирование единой схемы для чистой базы данных:
		- Источник (тип, URL, дата публикации)
		- Категория/кластер
		- Теги, ключевые слова, сущности
		- Тональность (если включено)
		- Векторное представление

**1.3. Выходные данные:**

* Чистая база в формате SQL/NoSQL (например, PostgreSQL, MongoDB) с индексами для быстрого поиска
* Логирование ошибок и статистика обработки (количество обработанных записей, время выполнения)

#### **2. Технические требования**

* **Язык программирования:** Python
* **Масштабируемость:**
	+ Поддержка обработки больших объемов данных (пакетная обработка, асинхронность)
	+ Возможность запуска в Docker/Kubernetes.
* **Оптимизация:**
	+ Использование кэширования для повторяющихся операций
	+ Минимизация потребления RAM/CPU
* **Гибкость:**
	+ Конфигурационные файлы для настройки параметров (например, выбор модели NLP, пороги для кластеризации)
	+ Модульная архитектура для добавления новых источников и методов обработки

#### **3. Этапы разработки**

1. **Анализ и проектирование:**
	1. Изучение NLP-моделей и алгоритмов кластеризации
	2. Проектирование схемы выходной базы данных
	3. Определение стека исполнителя
	4. Корректировка ТЗ исходя из комментариев исполнителя
2. **Тест-прототип:**
	1. Реализация базовой обработки (очистка, токенизация)
	2. Тестирование на небольшом датасете (100–500 записей)
3. **Расширение функционала:**
	1. Интеграция NLP и кластеризации
	2. Настройка векторизации и экспорта в БД
4. **Оптимизация и тестирование:**
	1. Стресс-тесты на больших данных (10k+ записей)
	2. Исправление узких мест в производительности
5. **Документация:**
	1. Руководство по запуску и настройке
	2. Примеры конфигурационных файлов

#### **4. Критерии приемки**

* Программа обрабатывает данные без потерь и ошибок (проверка на нашем датасете)
* Выходная база содержит все требуемые поля (источник, тип, категория, теги, кластеры, векторы и прочие).
* Время обработки: не более 2 секунд на 1000 текстов

#### **5. Бюджет и сроки**

* **Срок выполнения:** согласовывается с исполнителем
* **Бюджет:** по договоренности

#### **6. Условия сотрудничества**

* Отчеты о прогрессе по запросу заказчика
* Передача исходного кода, документации и лицензий на использованные библиотеки