АДАПТИВНАЯ ПЛАТФОРМА ДЛЯ АНАЛИЗА ТЕКСТОВ НА БАЗЕ ОТКРЫТЫХ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ

Зайда А.В.[1]
Научный руководитель: Савельев А.О.[2]
[1] *НИ ТПУ, студент гр. 8К12 ОИТ, ИШИТР, e-mail: avz68@tpu.ru*
[2] *НИ ТПУ, к.т.н., доцент ОИТ, ИШИТР, e-mail: sava@tpu.ru*

Аннотация

Работа описывает создание платформы, предоставляющей графический интерфейс для применения произвольных открытых моделей обработки естественного языка. С использованием инструментов ресурса Hugging Face удалось автоматизировать применение 101 из 150 апробированных моделей этой площадки. Для применения остальных моделей был реализован расширенный интерфейс с опциональными параметрами.

Ключевые слова: Обработка естественного языка, машинное обучение, микросервисная архитектура, Hugging Face, Django.

Введение

В современном мире возрастает применимость моделей машинного обучения. Различные инструменты обработки данных, основанные на машинном обучении, используются во всё большем количестве сфер человеческой деятельности, о чём свидетельствуют научные публикации. Например, в области обработки естественного языка в 2022 году было опубликовано 599 публикаций по данным ScienceDirect. Несмотря на то, что в 2023 году это количество было равно 674, к 2024 году число публикаций по этой дисциплине увеличилось до 1023 [4]. В силу востребованности открытых моделей, их частыми пользователями становятся специалисты без необходимой подготовки в области информационных технологий. Одним примером такого варианта использования является потребность социально-политических исследователей в обработке большого количества данных на естественном языке. В свою очередь, количество доступных моделей тоже растёт стремительно: в ноябре 2023 года на платформе Hugging Face было загружено приблизительно 400000 открытых моделей [1], а в августе 2024 это число превысило 1000000 [2]. По состоянию на 11.03.25 количество моделей на Hugging Face уже приблизилось к 1500000 [3]. При этом применение таких доступных моделей за пределами программных интерфейсов по-прежнему затруднено. У пользователей инструментов машинного обучения возникает потребность в более понятном графическом интерфейсе для использования разнообразных открытых моделей.

Таким образом, целью работы является прототипирование платформы, предоставляющей пользователям графический интерфейс для применения произвольных открытых моделей обработки естественного языка.

Для достижения поставленной цели были выдвинуты следующие задачи:

1. Провести анализ требований к платформе;
2. Спроектировать платформу с учётом выявленных требований;
3. Программно реализовать платформу;
4. Оценить адаптивность платформы в работе с произвольными моделями.

Анализ требований

Варианты использования платформы включают несколько этапов работ: управление данными и выполнение обработки, как представлено в виде UML диаграммы на рисунке 1.

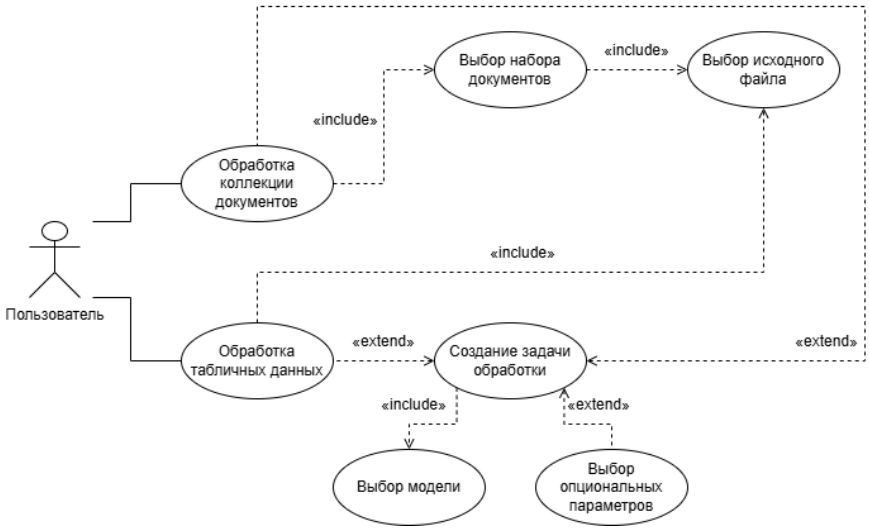

Рис. 1. UML диаграмма вариантов использования платформы обработки текстов на естественном языке

Можно выделить два существенно различных варианта обработки текстов на естественном языке: обработка табличных данных, характерных для анализа контента социальных медиа, и обработка полнотекстовых документов, например, текстов научных публикаций. Редко крупные документы необходимо обрабатывать по отдельности – часто происходит анализ именно коллекций логически связанных файлов. Потому перед обработкой возникает дополнительный шаг: выделение набора документов, которые будут анализироваться вместе. Когда исходные данные определены, запускается задача обработки открытой моделью. Модель обязательно необходимо идентифицировать, зачастую названием. Более того, многие модели принимают специфичные дополнительные параметры, которые изменяют их вывод в соответствии с потребностями пользователя.

На основе описанных вариантов использования были выявлены конкретные требования к платформе. Разрабатываемое решение направлено на анализ данных, предоставленных пользователем. Притом сбор данных пользователем выносится за границы платформы, что требует установить формат данных, соблюдение которого должен обеспечить пользователь. Эти требования учитывают 2 укрупнённых варианта использования: обработку табличных данных с множеством коротких, однотипных записей и обработку длинных документов.

Требования к данным:

Д.1. Данные должны быть представлены файлами в форматах CSV и PDF.

Д.2. Файл в формате PDF, переданный в качестве входных данных, должен содержать текстовых слой.

Переданные данные сохраняются в системе. Для удобства работы с загруженными файлами, необходимо предусмотреть возможность индексировать и искать файлы по пользовательским метаданным: названиям, авторам, тэгам и парам «ключ – значение». На этапе обработки сохранённых файлов, нужно предусмотреть возможность обрабатывать несколько из них совместно, чтобы обеспечить вариант использования с обработкой коллекции документов. В контексте платформы такие наборы файлов были названы датасетами.

Требования к хранению данных:

Х.1. Система должна хранить файлы вместе с метаданными, заданными пользователем при загрузке.

Х.2. Система должна поддерживать удаление, выгрузку и изменение метаданных загруженных файлов.

X.3. Система должна поддерживать объединение файлов в коллекции, обрабатываемые совместно – датасеты.

X.4. Система должна хранить датасеты вместе с метаданными, заданными пользователем при создании.

X.5. Система должна поддерживать удаление, изменение метаданных и изменение состава датасетов.

Для обработки данных произвольными моделями необходимо обозначить универсальные способы для их выбора и использования в системе. Более того, использование инструментов машинного обучения сопровождается значительными промежутками ожидания при вычислении результата. Потому важно организовать возможность создания очереди задач обработки данных моделями, чтобы пользователь не терял время на ручном переключении между задачами с долгим временем обработки.

Требования к обработке данных:

O.1. Система при создании задачи обработки должна принимать от пользователя датасет, подлежащий обработке, полное название открытой модели на Hugging Face и опциональные параметры конкретной модели.

O.2. Система должна ставить задачи обработки в очередь на исполнение.

O.3. Система должна поддерживать параллельное выполнение задач обработки.

O.4. Система должна отображать пользователю статус каждой задачи обработки.

O.5. Система должна отображать пользователю подробную информацию об ошибке в задаче обработки, если от пользователя требуются дополнительные данные о модели.

Проектирование платформы

Исходя из выдвинутых требований, было решено спроектировать платформу на основе микросервисной архитектуры, что придаст ей гибкость в условиях низкой связности компонентов. Всего было выделено 6 узлов программного решения: обратный прокси сервер, сервер наборов данных, сервер реляционной базы данных, сервер документно-ориентированной базы данных, брокер сообщений и сервер выполнения задач обработки. Обратный прокси сервер призван обеспечить доступность платформы по веб-протоколам, вынося логику маршрутизации запросов из прикладного уровня прочих компонентов. Сервер наборов данных обслуживает запросы пользователей, связанные с управлением исходными файлами и датасетами. Физически файлы и их метаданные сохраняются на сервере реляционной базы данных, с которым взаимодействует сервер наборов данных. Это позволяет индексировать данные при помощи инструментов реляционных СУБД, облегчая последующий поиск хранимой информации согласно требованиям X.1 и X.4. Запросы пользователя на обработку данных также проходят через сервер наборов данных, который записывает задачу обработки в реляционную базу данных и делегирует её выполнение, отправляя запись в брокер сообщений. Брокер сообщений поддерживает очередь сообщений, из которой сервер выполнения задач обработки последовательно извлекает задания, что удовлетворяет требованию O.2. Поскольку брокер сообщений может поддерживать множество потребителей сообщений, становиться возможным горизонтально масштабировать сервер выполнения задач обработки, позволяя параллельно выполнять независимые задачи, в соответствии с требованием O.3. В свою очередь, сервер выполнения задач обработки запускает запрашиваемые модели на выбранных текстах и записывает выходные данные в документно-ориентированную базу данных. Выбор документно-ориентированной модели связан с несогласованностью форматов сериализации результатов работы различных моделей, что не позволяет построить для таких записей единую схему. После записи результата в документно-ориентированную базу данных, пользователь может запросить результат, отправив запрос на сервер наборов данных. Описанная высокоуровневая архитектура узлов платформы представлена на UML диаграмме развёртывания на рисунке 2.

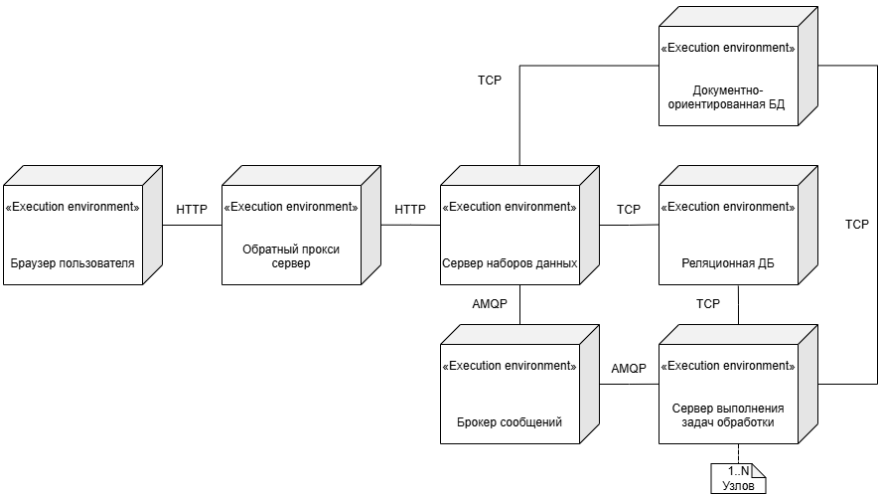

Рис. 2. UML диаграмма развёртывания платформы обработки текстов на естественном языке

Для хранения данных согласно требованиям X.1–X.5 была разработана модель файлов, их наборов и метаданных. Данная модель позволяет создать согласованное представление данных как на сервере наборов данных, так и на сервере реляционной базы данных путём объектно-реляционного отображения. В её основе лежит абстрактный класс Datafile, который представляет собой любой бинарных файл, хранимый в системе. С каждым файлом в системе ассоциированы метаданные, в соответствии с требованиями X.1 и X.4. Метаданные представлены абстрактным классом Metadata, который определяет название и автора файла, а также ассоциированные тэги и пары «ключ – значение». От класса Datafile наследуются два типа: SourceFile и DatasetFile. Эти конкретные имплементации класса Datafile ассоциированы с типами SourceMetadata и DatasetMetadata соответственно, которые различаются лишь набором тэгов. Теги, в свою очередь, описаны абстрактным классом Tags и конкретизированы для файлов исходников и датасетов типами SourceTags и DatasetTags соответственно. Дальнейшее отличие класса DatasetFile заключается в том, что он представляет наборы данных путём агрегирования файлов исходников в отдельный файл. Поскольку пользователь может удалять файлы из набора данных и включать в него новые, класс DatasetFile дополнительно хранит текущий список файлов исходников типа SourceFile, на основе которого может быть собран файл датасета. Описанная модель данных представлена на рисунке 3 в виде UML диаграммы классов.

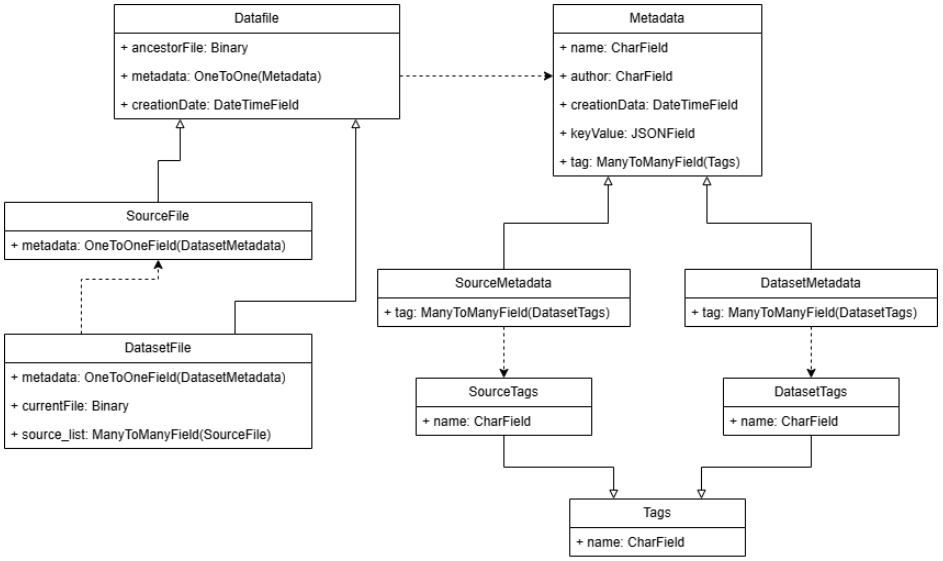

Рис. 3. UML диаграмма классов модели данных

Также, был специфицирован процесс обработки данных, включающий взаимодействие всех сервисов платформы. Этот процесс начинается с отправки пользователем запроса на обработку на сервер наборов данных, согласно требованию О.1. Если все данные запроса оказались корректными, создаётся запись о задаче обработки в реляционной базе данных, с целью отслеживания прогресса её выполнения, как требуется в О.4. После отправки задачи в очередь сообщений, пользователь получает уведомление. Сервер выполнения задач обработки принимает сообщение с задачей, выполняет её, изменяет статус по необходимости и записывает результат работы в документно-ориентированную базу данных. Притом в качестве результата также может быть записано сообщение об ошибке, согласно требованию О.5. Описанный порядок взаимодействий представлен на рисунке 4 в виде UML диаграммы последовательности.

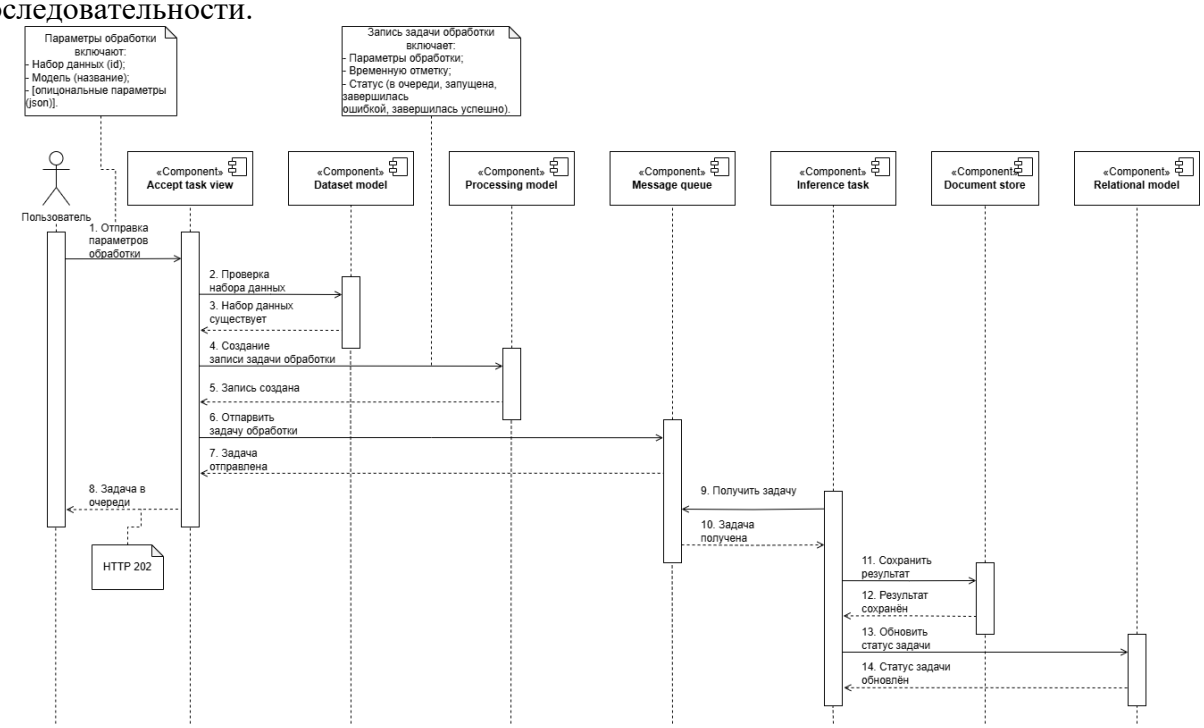

Рис. 4. UML диаграмма последовательности выполнения задачи обработки

Реализация платформы

Для создания сервера наборов данных был использован язык программирования Python с фреймворком Django, который позволяет реализовать пользовательский web-интерфейс. Для передачи задач между сервером наборов данных и сервером выполнения задач обработки была использована библиотека Celery для Python, организующая обмен сообщениями с несколькими производителями и потребителями. Для запуска моделей машинного обучения использовалась высокоуровневая библиотека Transformers от Hugging Face, которая позволяет запускать множество открытых моделей. В качестве реляционной базы данных была использована расширяемая PostgreSQL. В роли сервера документно-ориентированной базы данных была выбрана широко поддерживаемая MongoDB. В качестве брокера сообщений используется RabbitMQ, благодаря простой интеграции с Celery. Также, в качестве обратного прокси сервера был использован высокопроизводительный и гибкий Nginx. Созданные приложения и готовые решения были размещены в Docker контейнерах, чтобы обеспечить лёгкое развёртывание и внедрение платформы.

С использованием описанных технологий был разработан прототип платформы, интерфейс которой представлен на рисунке 5.

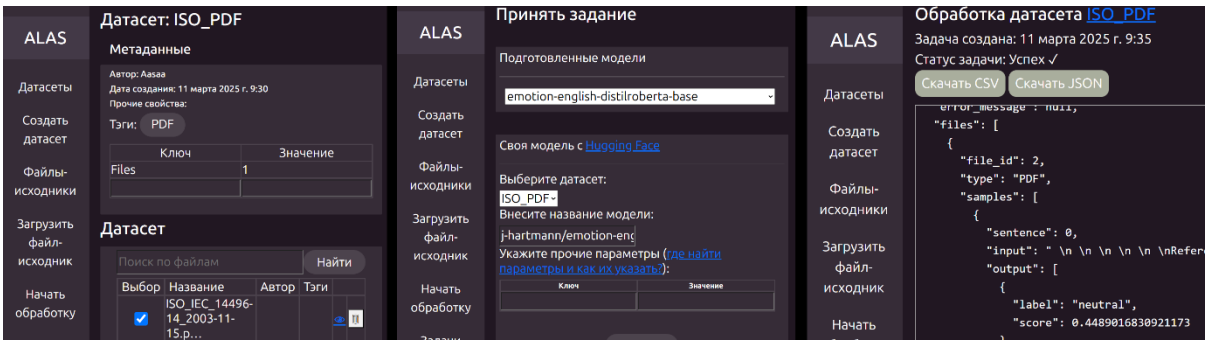

Рис. 5. Веб-интерфейс платформы

Оценка адаптивности платформы

При помощи API платформы Hugging Face были извлечены 150 моделей из 87473 с тэгом «text-classification», отсортированных по убыванию количества загрузок. Данная процедура выбора была нацелена «захватить» наиболее популярные и приближенные к задачам разрабатываемой платформы модели, ограничивая при этом аппаратные затраты на их проверку. В результате апробации, 101 из 150 моделей, идентифицированных лишь названием, были успешно собраны и запущены в рамках разрабатываемой платформы. Наиболее частой ошибкой при использовании моделей была неопределённость задачи, которую необходимо указать для некоторых мультимодальных моделей. В связи с этим, в интерфейс платформы были добавлены инструкции по заполнению опциональных параметров обработки, которые включат в себя указание задачи. Таким образом, действительная адаптивность платформы к произвольным моделям обработки естественного языка была повышена. Неучтёнными остаются ограничения применения квантованных моделей, веса которых закодированы специальным образом, что не позволяет запустить их стандартными подходами.

Заключение

В результате работы был реализован прототип адаптивной платформы с графическим интерфейсом для анализа текстов на базе открытых моделей машинного обучения с ресурса Hugging Face. Созданная платформа способна обрабатывать CSV и PDF файлы, а также их коллекции. При помощи программного обеспечения удалось скрыть детали применения моделей, запрашивая у пользователя лишь идентификатор средства обработки. При апробации платформы на 150 открытых моделях обработки естественного языка, платформа успешно запустила 67% из них. Дальнейшие вызовы, связанные с поддержкой квантованных моделей, были обозначены.

Список использованных источников

1. Aryani A. Estimated time: 12 Years to Review all 400k Models on Hugging Face // Medium. 2023. [Электронный ресурс]. – URL: medium.com/@amiraryani/estimated-time-12-years-to-review-all-400k-models-on-hugging-face-5b9b5b89bc17 (дата обращения: 11.03.2025).

2. HuggingFace Statistics // Originality.ai. [Электронный ресурс]. – URL: originality.ai/blog/huggingface-statistics (дата обращения: 11.03.2025).

3. Models // Huggingface.co. [Электронный ресурс]. – URL: huggingface.co/models (дата обращения: 11.03.2025).

4. Search ScienceDirect // Sciencedirect.com [Электронный ресурс]. – URL: sciencedirect.com/search?date=2022-2025&tak=NLP (дата обращения: 11.03.2025).