DOI 10.37882/2223-2966.2023.8-2.33

АРХИТЕКТУРА ГЕЙМИФИЦИРОВАНОЙ ПРОГРАММНОЙ ПЛАТФОРМЫ ДЛЯ ПОДГОТОВКИ ДАННЫХ ДЛЯ ОБУЧЕНИЯ AI МОДЕЛЕЙ

ARCHITECTURE OF A GAMIFIED SOFTWARE PLATFORM FOR PREPARING DATA FOR TRAINING AI MODELS

P. Tumanyan
S. Saradgishvili

Summary. This article describes the requirements and architecture for creating a multiplayer gamified platform that solves the problem of processing unstructured data to create training datasets for neural network models. The platform architectures are presented and the system's algorithm of operations is described. The gamification engine for performing data preparation tasks plays an important role in the platform. The article presents the structure of the gamification application mechanics and quality control tools, which ensure improving the efficiency of the data preparation process.

Keywords: gamified mechanics, gamification, data markup, training datasets.

Туманян Полина Игоревна
Аспирант, Санкт-Петербургский политехнический университет Петра Великого
polly.pol@mail.ru
Сараджишвили Сергей Эрикович
Доцент, кандидат технических наук, Санкт-Петербургский политехнический университет Петра Великого
ssaradg@yandex.ru

Аннотация. Статья посвящена описанию требований и архитектуры для создания многопользовательской геймифицированной платформы, решающей задачу обработки неструктурированных данных для создания обучающих наборов данных для обучения моделей нейронных сетей. Приведены архитектурные схемы платформы, описан алгоритм работы системы. Важную роль в платформе имеет движок игровых механик для выполнения задач по подготовке данных. Представлена структура приложения геймифицированные механики и инструменты контроля качества, обеспечивающие повышение эффективности процесса подготовки данных.

Ключевые слова: игровые механики, геймификация, разметка данных, обучающие наборы данных.

Введение

Сегодняшний мир все больше зависит от искусственного интеллекта и машинного обучения. Ключевым фактором для успешного обучения нейронных сетей является наличие больших и качественных наборов данных [1]. В то же время, создание таких наборов данных является сложной и трудоемкой задачей, а для исполнителей (аннотаторов) — это монотонный и неувлекательный процесс. Это может вызывать ошибки и снижение качества полученных данных. В связи с этим предлагается разработка геймифицированной платформы, которая преобразует процесс создания наборов данных в интерактивную, мотивирующую и результативную деятельность.

Внедрение игровых механик в процесс помогает увеличить скорость выполнения задач, поддерживает высокую степень вовлеченности участников, что в свою очередь приводит к повышению эффективности обработки данных. Дополнительно, это может снизить затраты на обучение и найм новых специалистов, уменьшая отток кадров и улучшая качество итоговых данных.

В настоящей работе предлагается архитектурное решение для платформы, решающей задачи классификации, аннотации и верификации данных с использованием игровых механик.

Концепция разрабатываемой системы и требования

Для разработки архитектуры проекта, необходимо определить требования и ограничения, которые накладываются на систему. В рамках данной задачи, требуется разработать архитектуру для платформы, которая позволит внешним заказчикам размещать задачи на подготовку данных (классификация, аннотация, разметка изображений) для обучения нейронных сетей. Платформа должна обеспечивать эффективный процесс подготовки данных с использованием геймификации, а также бесшовную передачу данных от и к заказчику через API.

Основные функциональные требования:

1. Платформа должна предоставлять возможность заказчикам размещать задачи на подготовку данных. Задачи могут включать классификацию, аннотацию и верификацию изображений.
2. Платформа должна автоматически разбивать задачи на подзадачи, которые могут быть выполнены исполнителями.
3. Платформа должна автоматически распределять подзадачи между исполнителями.

4. Платформа должна включать инструменты геймификации, которые будут использоваться для повышения эффективности процесса подготовки данных и улучшения качества данных.
5. После выполнения задачи исполнителем, платформа должна проводить верификацию данных. Этот процесс также должен быть геймифицирован.
6. После верификации данных, платформа должна передавать обработанные изображения заказчику.
7. Платформа должна обеспечивать бесшовную передачу данных от заказчика и к заказчику через API.

Нетехнические требования:

1. Платформа должна соответствовать стандартам безопасности для защиты данных заказчиков и исполнителей.
2. Платформа должна быть масштабируемой, чтобы справляться с увеличением объема задач и пользователей.
3. Производительность. Платформа должна обеспечивать быструю обработку задач и минимальные задержки в передаче данных.

Технологические требования:

1. Языки программирования. Выбор конкретных языков программирования будет зависеть от команды разработчиков, но они должны быть современными и поддерживать разработку высокопроизводительных веб-приложений.
2. База данных. База данных должна быть масштабируемой и обеспечивать быстрый доступ к данным.
3. API. API должен быть разработан с учетом лучших практик REST или GraphQL и обеспечивать безопасную и эффективную передачу данных.

Архитектура системы

Для решения задачи разработки платформы предлагается архитектура, изображенная на рис. 1.

Ее основные компоненты и выбранный технологический стек:

1. Фронтенд. Был выбран популярный JavaScript-фреймворк React.js для создания интерактивных пользовательских интерфейсов [2]. Он поддерживает компонентный подход, что упрощает разработку и поддержку кода. React.js был выбран из-за его эффективности, гибкости и широкой поддержки сообщества. React позволяет создавать масштабируемые и производительные веб-приложения с использованием компонентного подхода. Это упрощает разработку и поддержку кода.
2. Бэкенд. Node.js и Express.js были выбраны из-за их производительности и гибкости [3]. Node.js — это среда выполнения JavaScript на стороне сервера, которая позволяет разрабатывать масштабируемые веб-приложения. Она поддерживает асинхронное и событийное программирование, что идеально подходит для обработки большого количества запросов. Express.js минималистичный и гибкий веб-фреймворк для Node.js, который обеспечивает набор функций для веб— и мобильных приложений.
3. База данных. MongoDB — это NoSQL база данных, которая обеспечивает высокую производительность, высокую доступность и легкую масштабируемость [4]. Она работает на концепции коллекций и документов и поддерживает различные типы данных.
4. API. GraphQL — это язык запросов для API, который обеспечивает эффективное и мощное взаимодействие с данными. Он позволяет клиентам определять структуру ответов, что упрощает обработку данных на стороне клиента. GraphQL был выбран из-за его эффективности и мощности [5].
5. Геймификационный движок может быть разработан с нуля для обеспечения максимальной гибкости и соответствия требованиям платформы.

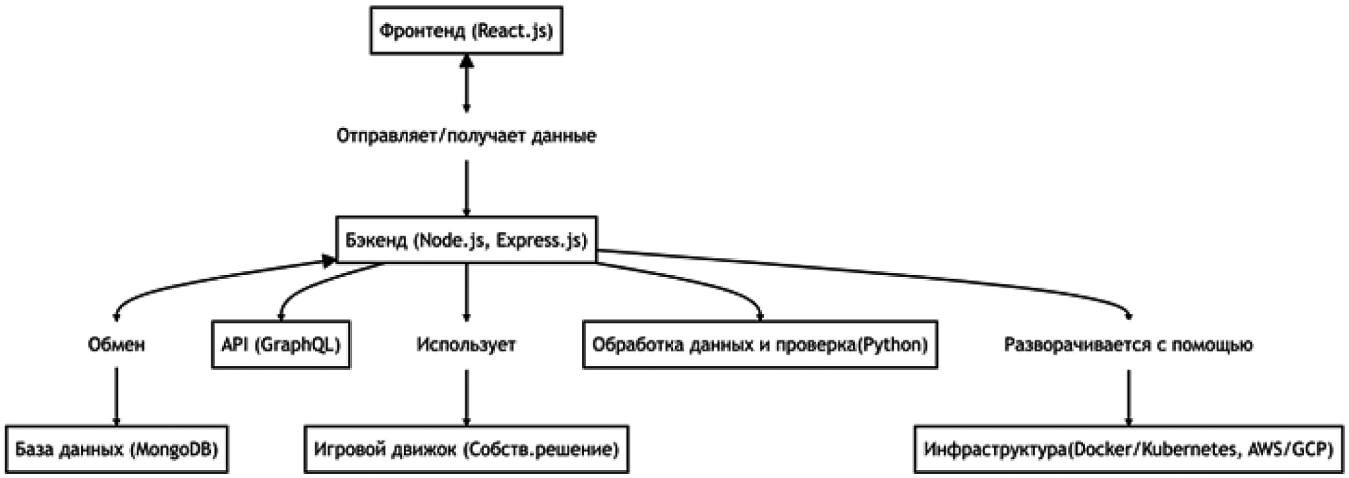

Рис. 1. Архитектура платформы

6. Обработка и верификация данных. Для этой цели выбран язык Python, он является одним из наиболее популярных языков для обработки данных и машинного обучения. Он поддерживает множество библиотек для обработки изображений и данных, таких как NumPy, Pandas и OpenCV.
7. Облачная инфраструктура. Docker/Kubernetes: Docker и Kubernetes используются для контейнеризации и оркестровки приложений, что обеспечивает их надежность, масштабируемость и портативность. AWS и GCP — это облачные провайдеры, которые предоставляют надежную и масштабируемую инфраструктуру для развертывания приложений [6].

Этот технологический стек был выбран в сравнении с другими технологиями из-за его производительности, масштабируемости, гибкости и поддержки сообщества. Кроме того, все эти технологии хорошо интегрируются друг с другом, что обеспечивает гладкую и эффективную разработку.

Описание взаимодействия модулей на примере жизненного цикла выполнения задачи на платформе

1. Пользователь входит в систему через интерфейс, разработанный на React.js (Frontend). При входе в систему данные пользователя проверяются и аутентифицируются.
2. Пользователь выбирает задачу для выполнения, запрос передается через GraphQL (API) на сервер.
3. Сервер, работающий на Node.js и Express.js, обращается к базе данных MongoDB для получения информации о задаче.
4. Информация о задаче возвращается на сервер, обрабатывается и возвращается обратно на клиент через GraphQL (API).
5. Пользователь видит информацию о задаче и начинает работать над задачей.
6. После завершения задачи результаты отправляются на сервер через GraphQL (API).
7. Сервер обрабатывает результаты задачи с помощью Python (Data Processing) и сохраняет результаты в базе данных MongoDB.
8. Обратная связь о результате задачи возвращается пользователю через GraphQL (API) и отображается на клиенте с помощью React.js (Frontend).
9. На основе результатов задачи обновляются награды пользователя в геймификационном движке.

Вся инфраструктура развертывается и управляется с помощью Docker/Kubernetes и облачных провайдеров, таких как AWS или GCP, что обеспечивает надежность, масштабируемость и портативность приложения.

Механики геймификации и структура приложения

Особенность разрабатываемой платформы подготовки данных являются встроенные в процесс работы с данными игровые механики. Далее подробно описан геймификационный движок, механики и их влияние на повышение эффективности процесса:

1. Система уровней и опыта. Аннотаторы могут зарабатывать опыт за выполнение задач и достижение целей. При достижении определенного количества опыта исполнитель повышает свой уровень. Уровни могут открывать новые возможности или задачи более высокого уровня. Эта механика мотивирует исполнителей улучшать свои навыки и усердно работать, чтобы получить больше опыта и повысить свой уровень. Это может привести к увеличению количества выполненных задач и улучшению качества данных.
2. Аннотаторы могут получать бейджи и достижения за выполнение определенных задач или достижение важных вех. Это может быть, например, выполнение определенного количества задач, работа над сложными проектами или высокая точность в задачах классификации. Данная механика может увеличить удовлетворение исполнителей от работы и мотивировать их на дальнейшие успехи, снизит отток исполнителей с платформы.
3. Лидерборды могут быть использованы для стимулирования здоровой конкуренции между исполнителями, что может увеличить их производительность и качество работы. Они могут отображать топ-исполнителей по различным метрикам, таким как количество выполненных задач, точность классификации или общий опыт.
4. Система наград. Исполнители могут получать награды за достижение определенных целей или выполнение задач. Награды могут быть виртуальными (например, бонусы или улучшения в приложении) или реальными (например, бонусы к зарплате или подарочные карты). Награды за достижение определенных целей или выполнение задач могут служить дополнительной нематериальной мотивацией для исполнителей.
5. Миссии и квесты могут быть использованы для обучения новых исполнителей или стимулирования выполнения определенных типов задач. Они могут включать в себя серию задач, которые нужно выполнить для получения награды.
6. Обратная связь и прогресс. Исполнители должны получать обратную связь о своем прогрессе и производительности. Это может включать в себя статистику о выполненных задачах, точности классификации, текущем уровне и опыте, а также предстоящих целях или задачах. Постоянная обратная связь и видимость прогресса могут помочь исполнителям оценить свою работу и улучшить свои навыки.

В жизненном цикле исполнителя на платформе геймифицированные механики используются на следующих этапах:

1. Исполнитель регистрируется на платформе и проходит начальное интерактивное обучение.
2. Исполнитель начинает выполнять задачи, за которые он получает опыт и возможно награды.
3. По мере выполнения задач исполнитель получает бейджи и достижения, которые отображаются в его профиле.
4. Исполнитель может видеть свой прогресс и сравнивать его с другими исполнителями на лидербордах.
5. Исполнитель может получать дополнительные задачи или миссии, которые помогают ему улучшить свои навыки и получить больше наград.
6. Исполнитель продолжает работать, улучшая свои навыки и получая награды.

Также исполнитель может обменять накопленный опыт на материальные награды, например, фирменные вещи.

Контроль качества исполнителей

Контроль качества подготовки данных на платформе будет осуществляться с использованием следующих механик:

1. Gold Set (Золотой набор). Это набор данных, который был тщательно размечен и представляет собой эталон правильной разметки. Этот набор данных используется для оценки качества работы аннотаторов. Изображения из данного набора периодически появляются среди настоящих, если исполнители делают ошибки в разметке этого набора данных, это указывает на проблемы в их работе [7].
2. Использование слепых этапах, на которых несколько исполнителей независимо размечают одни и те же данные. Затем их результаты сравниваются. Если аннотации совпадают, это указывает на высокое качество работы. Если аннотации не совпадают, данные отправляются на дополнительную проверку.
3. Время, которое исполнитель тратит на выполнение задачи, также может быть индикатором качества. Если аннотатор выполняет задачу слишком быстро, это может указывать на то, что он не уделяет достаточно внимания деталям.
4. Исполнители получают обратную связь о своей работе и имеют возможность улучшить свои навыки. Это может включать в себя обучение на основе золотого набора данных, а также обратную связь от контролеров качества.
5. Механики геймификации, такие как система уровней, бейджи и достижения, могут стимулировать исполнителей улучшать качество своей работы. Аннотаторы, которые стремятся получить более высокий уровень или бейдж, могут уделить больше внимания качеству своей работы.

В контексте жизненного цикла исполнителя задачи, аннотатор начинает с выполнения задач и получения обратной связи. По мере того, как он улучшает свои навыки, он начинает работать над более сложными задачами и получает больше опыта и наград. Весь этот процесс подкрепляется механиками геймификации, которые делают процесс более интересным и мотивирующим.

В заключение, разработка архитектуры и геймифицированных механик для платформы подготовки данных для обучения AI моделей является сложной, но важной задачей. Правильно спроектированная архитектура обеспечивает эффективность, масштабируемость и надежность платформы, в то время как геймификация помогает повысить мотивацию и производительность исполнителей.

Механики геймификации, такие как система уровней, бейджи, лидерборды и система наград, используются для стимулирования аннотаторов и повышения качества подготовки данных. Эти механики, вместе с обратной связью и обучением, помогают исполнителям улучшать свои навыки и продуктивность.

Контроль качества данных обеспечивается с помощью золотых наборов, слепых этапов, отслеживания времени выполнения задачи и обратной связи. Эти механизмы помогают обеспечить высокое качество подготовки данных, что в свою очередь ведет к более точным и эффективным AI моделям.

В целом, разработка архитектуры и геймифицированных механик для такой платформы требует глубокого понимания как технологий, так и психологии пользователя. Но при правильном подходе она может привести к созданию мощной и эффективной платформы для подготовки данных для обучения AI моделей.

ЛИТЕРАТУРА

1. Linjordet T., Balog K. Impact of Training Dataset Size on Neural Answer Selection Models [Электронный ресурс]. URL: https://ar5iv.org/pdf/impact-of-training-dataset-size-on-neural-answer-selection-models.pdf (Дата обращения: 27.05.2023).
2. Azad, A.K., & Rana, A.E. (2020). A Comparative Study of React.js and Angular.js Frameworks in Web Development. International Journal of Computer Applications, 175(4), 7–11.
3. Fertalj, K., & Hoic-Bozic, N. (2018). The Impact of Node.js on Web Application Development. Journal of Universal Computer Science, 24(8), 1091–1106.
4. Chodorow, K. (2013). MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. O'Reilly Media, Inc.
5. Biehl, M. (2018). Learning GraphQL: Declarative Data Fetching for Modern Web Apps. O'Reilly Media, Inc.
6. Burns, B., & Oppenheimer, D. (2016). Design patterns for container-based distributed systems. In 8th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 16).
7. Способы обеспечения качества данных для машинного обучения [Электронный ресурс]. URL: https://vc.ru/ml/353279-sposoby-obespecheniya-kachestva-dannyh-dlya-mashinnogo-obucheniya (Дата обращения: 02.02.2023).

Журнал «Современная наука: актуальные проблемы теории и практики»