Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 21.Б11-мм

Система для автоматизированной проверки больших языковых моделей на основе заданий государственных экзаменов

Выродов Михаил Владимирович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
асс. кафедры СП Е. С. Спирин

Санкт-Петербург
2024

Оглавление

Введение **3**

1. Постановка задачи **5**

2. Обзор **6**

2.1. Нужные понятия из машинного обучения 6

2.2. Обзор аналогов 8

2.3. Выбор набора данных для валидации 11

3. Метод **14**

3.1. Парсер 14

3.2. Система для валидации БЯМ 17

4. Эксперимент **22**

Заключение **24**

Список литературы **25**

Введение

В последние годы большие языковые модели (БЯМ) получил огромное развитие и стали использоваться повсеместно. Для хорошей работы они должны сначала обучиться на огромном массиве текстовых данных. После обучения модели разработчики должны провалидировать её работу. Если результат валидации удовлетворительный, то это значит, что языковая модель готова к использованию. Популярным примером БЯМ является GPT-4 [2].

Однако, несмотря на их впечатляющие достижения, внедрение БЯМ в IT проекты требует тщательного процесса валидации. Валидация языковых моделей (ЯМ) – это процесс проверки их способности к правильному пониманию и генерации текста. ЯМ нуждаются в валидации для того, чтобы оценить их эффективность и способность разумно отвечать не только на вопросы, на которых модель обучалась, но и на вопросы, которые она видит в первый раз.

Ручная валидация БЯМ требует значительных временных ресурсов. Необходимость в анализе больших объемов данных и различных сценариев использования модели делает этот процесс длительным и сложным. Возникает потребность в системе, которая сделает процесс валидации БЯМ более удобным, автоматизируя рутинные задачи и предоставляя быстрые результаты.

Одним из ключевых аспектов успешной валидации является выбор подходящего набора данных. Он должен проверять способность ЯМ отвечать на вопросы по множеству различных тем. Кроме того, задания в наборе данных должны быть различной сложности, так как хорошая БЯМ должна уметь отвечать как на вопросы уровня младшей школы, так и на вопросы университетского уровня.

MMLU [4] и MT-Bench [3] — примеры наборов данных для валидации ЯМ на английском языке. Сейчас они являются одними из стандартов для валидации ЯМ. Задания в MMLU нацелены на выбор одного правильного ответа из четырёх предложенных. Вопросы из MT-Bench, в свою очередь, требуют связный текст в качестве ответа.

Для проверки работы БЯМ сразу в нескольких сценариях использования разработчики часто валидируют БЯМ и на MMLU, и на MT-Bench. Стоит отметить, что эти наборы данных не подходят для валидации БЯМ, нацеленных на русскоязычного пользователя, так как в них содержатся задания только на английском языке.

Хорошим набором данных для валидации БЯМ, обученных на русскоязычных текстах, могут послужить задания из ЕГЭ и ОГЭ. Они затрагивают множество различных тем и содержат как лёгкие вопросы, так и сложные. Также вопросы в этих экзаменах покрывают сразу несколько сценариев использования БЯМ, так как в них есть задания как на односложный, так и на развёрнутый ответ.

1. Постановка задачи

Целью работы является создание системы, предназначенной для валидации БЯМ, которая будет основана на наборе данных, состоящем из заданий государственных экзаменов. Для её выполнения были поставлены следующие задачи:

1. Провести обзор предметной области, который включает в себя обзор аналогов системы, анализ популярных валидационных наборов данных и существующих бенчмарков для валидации БЯМ;

2. Собрать банк вопросов из государственных экзаменов, провести очистку и классифицировать задания по типам вопросов и способам взаимодействия с БЯМ;

3. Создать систему, которая будет валидировать подаваемую на вход БЯМ на собранном наборе данных. Для этого система должна поддерживать наиболее популярные интерфейсы для взаимодействия с БЯМ;

4. Апробировать систему на существующих популярных БЯМ, построить их качественное сравнение.

2. Обзор

В данной главе представлен обзор аналогов разрабатываемой мною системы и анализ популярных валидационных наборов данных в индустрии.

2.1. Нужные понятия из машинного обучения

Языковая модель — это распределение вероятностей по последовательностям слов. Любой последовательности слов длины m присваивается вероятность $P(w_1, w_2, ..., w_m)$. Обычно ЯМ представляет собой рекуррентную нейронную сеть.

Эта нейронная сеть учится предсказывать вероятности $P(w, context)$ $\forall w \in V$, где V — множество всех возможных слов, а $context$ — какой-то набор слов. $P(w, context)$ — вероятность того, что при добавлении к тексту $context$ следующего слова w, получившийся текст больше всего похож на тексты, на которых обучалась ЯМ. То есть модель, получив какой-то текстовый запрос $context$, считает вероятности $P(w, context)$ $\forall w \in V$, после этого выбирает наиболее вероятное следующее слово w_i для текста $context$ и теперь уже считает вероятность $P(w, context + w_i)$ $\forall w \in V$, где $context + w_i$ — строка $context$, сконкатенированная со словом w_i. Далее ЯМ выбирает наиболее вероятное следующее слово для строки $context + w_i$. Так ЯМ предугадывает следующие слова до того момента, пока самым вероятным следующим словом не станет специальное слово, обозначающее конец генерации текста. Накопившаяся последовательность слов является ответом ЯМ на изначальный текстовый запрос $context$.

Возникает вопрос, как ЯМ могут для любого слова w и последовательности слов $context$ посчитать вероятность того, что w является наиболее подходящим следующим словом для текста $context$? Данное распределение вероятностей нейронная сеть учит благодаря корректировки значений своих весов с помощью алгоритма градиентного спуска. Подробнее о самой популярной архитектуре языковой модели на основе нейронной сети написано в статье [1].

Большие языковые модели — неофициальный термин для ЯМ на основе нейронный сетей, которые имеют более миллиарда весов. БЯМ обучаются очень долго и для хорошего обучения им требуется большой текстовый набор данных, который измеряется петабайтами.

Валидация БЯМ — важный этап, который помогает оценить качество и эффективность БЯМ. Процесс валидации включает в себя использование отдельного набора данных, отличного от того, на котором проводилось обучение модели. Это необходимо для того, чтобы проверить её способность разумно отвечать не только на вопросы, на которых модель обучалась, но и на вопросы, которые она видит в первый раз.

Набор данных для валидации должен быть репрезентативным, то есть должен отражать разнообразие данных, которые модель может встретить в реальных условиях. Также данные для валидации не должны пересекаться с данными, использованными в процессе обучения модели, так как в ином случае высока вероятность того, что после валидации мы получим ложные представления о работоспособности модели, так как она легко может ответить на вопросы, на которых обучалась, но на вопросы, которые она видит впервые, её ответы с большой вероятностью будут хуже.

Процесс валидации БЯМ:

- Выбор валидационного набора данных;
- Получение ответов БЯМ на вопросы из этого набора данных;
- Оценка схожести данных ответов с правильными ответами на задания;

Возникает вопрос, как происходит оценка схожести ответов БЯМ с настоящими ответами? Для этого используются метрики. Метрика — способ измерения того, насколько хорошо работает модель на данных (это определение не связано с математическим определением метрики). Метрика для одного задания из валидационного набора данных, правильного ответа на это задание и ответа модели на это задание выдаёт число, которое должно отражать, насколько ответ модели близок

к правильному ответу на задание. В зависимости от оценки эффективности модели с помощью метрик, в процессе валидации делается вывод о том, стоит ли дорабатывать модель, или же она достаточно хорошо справляется с заданиями из валидационного набора данных.

В качестве примера метрики на заданиях, в которых надо выбрать один вариант ответа из четырёх, можно привести функцию, которая равна единице, если модель выбрала правильный ответ на задание, или равна нулю, если модель ответила на вопрос неправильно.

2.2. Обзор аналогов

В ходе обзора похожих систем по автоматической валидации БЯМ было выделено два open-source проекта — MERA [1] и Open LLM Leaderboard [2].

MERA — веб-приложение для облегчённой валидации БЯМ, нацеленных на русскоязычного пользователя, созданное Альянсом в сфере ИИ [3]. В него входят множество российских IT-компаний. Например, Яндекс, VK, Сбер.

Были выделены следующие преимущества данного проекта:

- В нём поддерживается валидация на нескольких популярных наборах данных как на русском языке, так и на английском;
- Есть валидационный набор данных USE[4], состоящий из заданий ЕГЭ по русскому языку;
- Есть либерборд БЯМ, который позволяет сравнивать их между собой и выбирать наиболее эффективные.

[1] https://mera.a-ai.ru/ru
[2] https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
[3] https://a-ai.ru/
[4] https://mera.a-ai.ru/ru/tasks/15

	Модель, команда	Общий результат	BPS	CheGeKa	LCS
1	Human Benchmark MERA	0.872	1.0	0.719 / 0.645	0.56
2	MTS AI Chat Medium MTS AI	0.536	0.23	0.05 / 0.022	0.178
3	GigaChat Pro SberDevices	0.514	0.224	0.451 / 0.363	0.12
4	MTS AI Chat 7B MTS AI	0.479	0.276	0.083 / 0.046	0.094

Рис. 1: Фрагмент лидерборда MERA

Также были выделены минусы MERA:

- Касательно заданий из государственных экзаменов, в проекте есть задания из ЕГЭ только по русскому языку;
- В проекте не поддерживается валидация на заданиях, требующих генерацию связного текста, что является большой проблемой, так как такие задания — самый частый сценарий использования больших языковых моделей;
- MERA делает валидацию более удобной, но не автоматизирует её полностью. Для получения результатов валидации с помощью MERA нужно сначала скачать локально их валидационные наборы данных и open-source бенчмарк для БЯМ. С помощью этого бенчмарка получить ответы модели на вопросы из скачанных наборов данных. После этого загрузить ответы модели на сайт MERA для получения финальных оценок валидации.

Основная проблема MERA в том, что валидация БЯМ через данный проект всё ещё требует большого количество усилий. Из-за этого проектом пользуются довольно редко. Сейчас в лидерборде MERA представлены всего 30 БЯМ.

Возникает потребность в системе, с помощью которой можно было бы валидировать нужную БЯМ нажатием одной кнопки.

Open LLM Leaderboard предоставляет такую возможность. Это популярное веб-приложение, интегрированное в сайт HuggingFace. Фактически, оно сейчас является стандартом для валидации open-source БЯМ, этим приложением пользуются ежедневно. Оно имеет $\approx 10k$ лайков на HuggingFace (аналог звёзд на GitHub).

Были выделены следующие преимущества данного проекта:

- С его помощью можно валидировать БЯМ сразу на нескольких популярных наборах данных на английском языке, которые являются стандартами для валидации в индустрии;
- В отличие от MERA, для валидации вашей модели в этом проекте достаточно указать ссылку на модель, и она автоматически провалидируется.

Model	Average	ARC	HellaSwag
davidkim205/Rhea-72b-v0.5	81.22	79.78	91.15
MTSAIR/MultiVerse_70B	81	78.67	89.77
MTSAIR/MultiVerse_70B	80.98	78.58	89.74
abacusai/Smaug-72B-v0.1	80.48	76.02	89.27

Рис. 2: Фрагмент лидерборда Open LLM Leaderboard

Также были выделены минусы Open LLM Leaderboard:

- В проекте не поддерживается валидация на заданиях, требующих генерацию связного текста;
- Валидационные наборы данных в этом проекте только на английском языке. Это большой минус, так как это значит, что валидация не проверяет специфичные для русской аудитории знания. Например, знания российских законов и российской истории. По этой причине данный проект не подходит для валидации БЯМ, нацеленных на русскоязычного пользователя;

После обзора аналогов были выделены два главных требования к моей системе:

- Система должна поддерживать валидацию на русскоязычных заданиях, требущих связный текст в качестве ответа, так как это самый популярный сценарий использования БЯМ;
- Система должна предоставлять удобный интерфейс для валидации. Для получения результатов валидации своей БЯМ с помощью моей системы пользователю должно быть достаточно только указать ссылку на его модель.

2.3. Выбор набора данных для валидации

Мною было изучено несколько популярных русскоязычных и англоязычных валидационных наборов данных для БЯМ. MMLU [4] и MT-Bench [3] — одни из наиболее часто используемых наборов данных для валидации БЯМ.

MMLU был впервые представлен на конференции A* уровня ICLR[5] в 2021 году. Он состоит из заданий из экзаменов для поступления в магистратуру и аспирантуру в США, из экзамена на получение медицинской лицензии и из различных задач университетского уровня сложности. Вопросы в этом наборе данных охватывают 57 различных тем и делятся на несколько уровней сложности. Все вопросы имеют один вид — выбор одного правильного ответ из четырёх возможных ответов на задание.

As of 2017, how many of the world's 1-year-old children today have been vaccinated against some disease?
(A) 80%
(B) 60%
(C) 40%
(D) 20%

Рис. 3: Пример задания из MMLU [4]

[5] https://iclr.cc/

MT-Bench включает в себя 80 высококачественных вопросов, классифицированных на восемь типов. Каждая категория содержит десять многозадачных вопросов, специально разработанных для того, чтобы проверить способности моделей к ведению диалога. Каждый вопрос состоит из двух частей — начальный вопрос и уточняющий вопрос по теме начального вопроса.

Начальный вопрос	Уточняющий вопрос
Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions.	Rewrite your previous response. Start every sentence with the letter A.

Таблица 1: Пример вопроса из MT-Bench

Минусы этих наборов данных в том, что они не подходят для валидации БЯМ, обученных на русскоязычных текстах, так как задания в этих наборах данных на английском языке. Возникает потребность в наборе данных для валидации БЯМ, нацеленном на русскоязычного пользователя, при этом содержащем задания, схожие как с MMLU, так и с MT-Bench, для валидации БЯМ в нескольких сценариях использования.

Таким набором данных могут послужить задания из российских государственных экзаменов. Преимущества такого набора данных заключаются в том, что задания из государственных экзаменов охватывают множество различных тем и делятся на уровни сложности. Задания в ЕГЭ по одному предмету обычно сложнее, чем задания ОГЭ по этому же предмету. Кроме того, в одном варианте ЕГЭ есть как очень простые задания, которые не требуют длинных рассуждений, так и задания, для правильно ответа на которые надо провести большую цепочку математических вычислений и знать множество теорем. Также в интернете есть общедоступные огромные базы заданий из ЕГЭ и ОГЭ с ответами и с объяснениями того, как ответ был получен. Ещё одним плюсом такого выбора набора данных является то, что задания в ЕГЭ и ОГЭ могут быть разных типов. В экзаменах встречаются как задания

на выбор ответа, так и задания на установление соответствия и задания с текстовым ответом. Такие задания позволяют оценить работу модели в различных сценариях использования. Таким образом, задания из государственных экзаменов могут послужить хорошим набором данных для валидации БЯМ.

3. Метод

В ходе работы было создано следующее:

- Парсер заданий из государственных экзаменов;
- Скрипт для валидации БЯМ на собранном с помощью парсера наборе данных;
- Веб-приложение с возможностью отправлять запросы на валидацию своей БЯМ и с лидербордом БЯМ, в котором модели отсортированы по правильности ответов на задания из собранного мною набора данных;

3.1. Парсер

Было принято решение извлекать задания с сайта sdamgia.ru[6], так как на этом сайте есть наибольшее количество заданий из государственных экзаменов с ответами, сами задания имеют унифицированный вид и на этот сайт можно отправлять множество частых GET запросов с одного ip адреса.

Для парсинга данного сайта был выбран Python фреймворк Scrapy[7], потому что у него удобный интерфейс, в рамках которого можно легко сохранять извлечённые с сайта задания в различных форматах. Также он парсит страницы намного быстрее, чем BeautifulSoup[8], так как парсинг одних и тех же страниц на BeautifulSoup у меня занимал около 30-ти минут, а на Scrapy около пяти минут.

Парсер ищет на сайте sdamgia.ru нужный вариант экзамена, извлекает оттуда задания и сохраняет их в файл. Возникает вопрос, почему парсер не просто извлекает все задания с сайта, а именно парсит вопросы из вариантов экзаменов?

На сайте sdamgia.ru есть множество заданий, которые не входят в варианты нынешних экзаменов и выбиваются из текущих стандартов

[6] https://sdamgia.ru/
[7] https://scrapy.org/
[8] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

заданий. Для того, чтобы валидация языковой модели была схожа с реальной проверкой знаний учеников на государственных экзаменах, было принято решение на данный момент извлекать с сайта только актуальные задания, которые могут быть в вариантах государственных экзаменов. Это удобно делать с помощью парсинга вариантов с сайта sdamgia.ru, так как функциональность этого сайта позволяет создать один вариант, который будет содержать все актуальные задания с этого сайта по конкретному предмету и по конкретному экзамену.

Парсер имеет несколько параметров:

- `subject` — предмет, по которому проводится экзамен. На данный момент можно парсить варианты экзаменов по предметам "Обществознание" и "Литература";
- `exam_type` — тип экзамена, вариант которого будет парситься. На данный момент можно парсить варианты из ЕГЭ и ОГЭ;
- `test_id` — номер варианта экзамена на сайте sdamgia.ru. Задания из этого варианта будут извлечены с сайта и сохранены в файл;
- `output_file` — название и формат файла, в который будут сохранены извлечённые из варианта задания.

Инструкции по вызову парсера описаны в репозитории проекта[9].

В Scrapy за парсинг сайтов отвечает класс `scrapy.Spider`, поэтому для парсинга сайта sdamgia.ru был создан класс `SdamgiaSpider`, который наследуется от `scrapy.Spider`. При вызове парсера сначала вызывается метод `start_requests()`, внутри которого определяются ссылки на каждое задание из варианта экзамена, номер которого был передан при вызове парсера. После этого из каждого задания извлекается нужная информация в методе `parse(...)`. Вся извлечённая информация записывается в файл, имя которого передаётся при вызове парсера.

Мною с помощью данного парсера были извлечены все актуальные задания из экзаменов ЕГЭ и ОГЭ по предметам "Обществознание" и "Литература" с сайта sdamgia.ru.

[9] https://github.com/deepvk/goat/

Извлечённые задания были очищены от специфичных для html символов и разделены на три класса в зависимости от типа ответа на задание:

- Задания с выбором ответа;
- Задания на установление соответствия;
- Задания с текстовым ответом.

Также из каждого задания была извлечена дополнительная информация, которая может пригодится при поиске по заданиям или может послужить основой для более подробной классификации заданий. В эту информацию входят:

- Номер задания в извлечённом варианте;
- Тип задания в варианте государственного экзамена;
- Булевое значение, которое истинно, когда задание предполагает поиск информации по большому тексту, который есть в варианте государственного экзамена, и ложно, когда для решение задания достаточно знать только текст самого задания;
- Ссылка на задание;
- Раздел кодификатора ФИПИ, к которому принадлежит данное задание;
- Максимальное количество баллов, которое можно получить за задание;
- Таблица с критериями оценивания задания, если такая имеется;
- Html код задания.

Всего с сайта sdamgia.ru было извлечено 6211 заданий по обществознанию и литературе.

Архив с извлечёнными заданиями доступен по ссылке[10].

[10] https://drive.google.com/file/d/1bjLl2RD3CC6vN6zf4Yvo9K-IHCNRVH6C/view?usp=sharing

Тип экзамена	Предмет	Количество вопросов	Выбор ответа	Соответствие	Текстовый ответ
ЕГЭ	Обществознание	2936	1751	580	605
ЕГЭ	Литература	1174	108	66	1000
ОГЭ	Обществознание	1673	1130	126	417
ОГЭ	Литература	428	0	0	428

Таблица 2: Результаты парсинга

3.2. Система для валидации БЯМ

Разработанная мною система для валидации БЯМ состоит из трёх компонент:

- Скрипт, который валидирует подаваемую на вход языковую модель;
- База данных с двумя таблицами — лидерборд моделей и таблица запросов на валидацию;
- Веб-приложение с возможностью отправлять новые запросы на валидацию своей БЯМ в БД и с отображением таблицы лидерборда БЯМ из БД.

3.2.1. Архитектура

На рисунке приведена диаграмма активности моей системы. Через веб-приложение пользователь добавляет запрос на валидацию его модели в БД. Далее БД добавляет этот запрос в таблицу и уведомляет бэкенд о добавлении нового запроса.

Бэкенд — это сервер с GPU, который по уведомлению БД достаёт из неё новые запросы на валидацию и запускает валидацию моделей из этих запросов на моём наборе данных с помощью open-source бенчмарков для языковый моделей.

После получения значений нужных метрик валидации на моём наборе данных, бэкенд кладёт эти значения в БД и из них на фронтенде формируется лидерборд моделей по правильности работы на моём наборе данных.

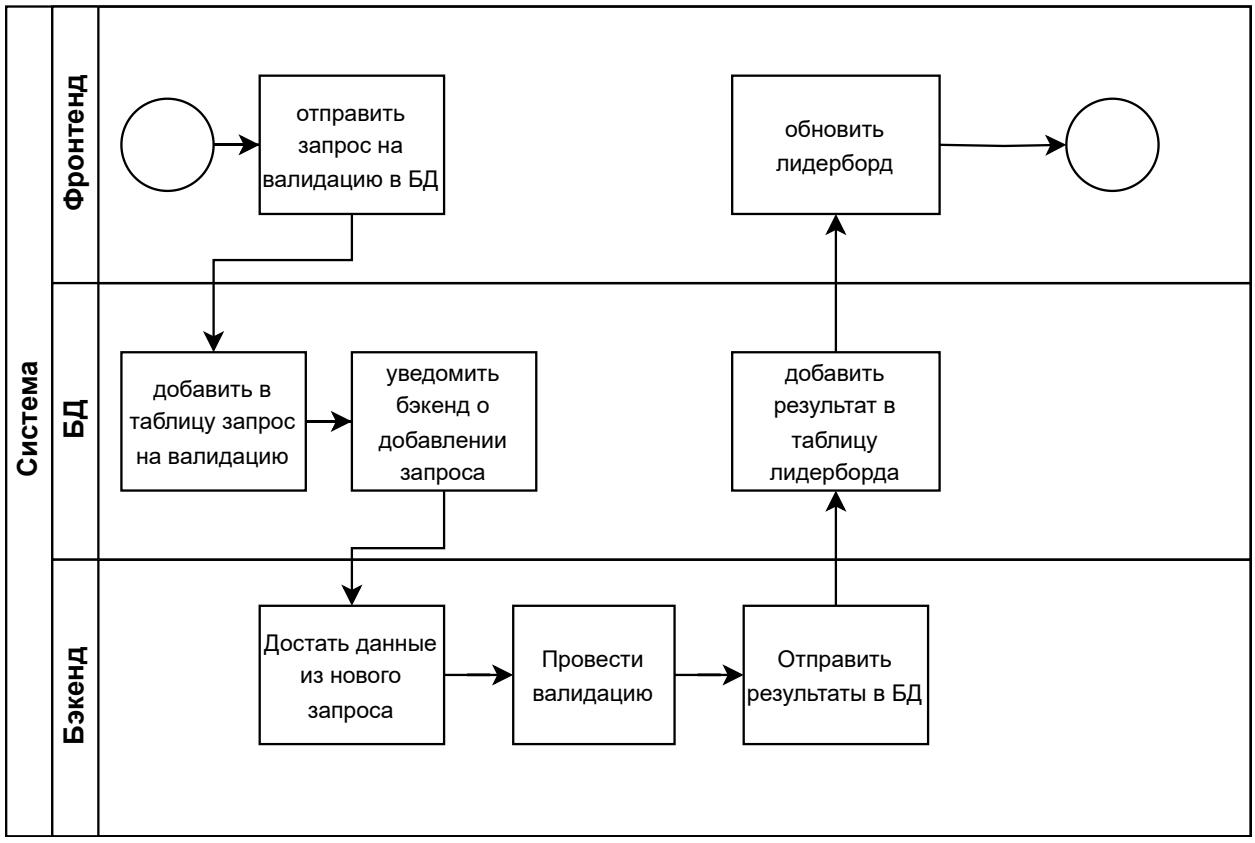

Рис. 4: Диаграмма активности системы

3.2.2. Используемые технологии

В ходе создания системы были использованы следующие технологии:

- Веб-приложение было сделано в виде HuggingFace Space[11], так как HuggingFace выдаёт бесплатную ВМ для быстрой развёртки приложения;
- В качестве фреймворка для построения интерфейса веб-приложения был выбран gradio[12], так как gradio специально был разработан для построения легковесных веб-приложений в сфере машинного обучения и этот фреймворк — принятый стандарт для построения веб-приложений, развёрнутых на HuggingFace;
- В проекте используется база данных PostgreSQL. Данная СУБД была выбрана из-за своей большой функциональности и наличия простого механизма отправки уведомлений об обновлении таблиц на удалённый хост;
- Для валидации моделей на бэкенде было решено не создавать новый бенчмарк с нуля, а задействовать open-source решения, зарекомендовавшие себя в индустрии. Это бенчмарки LM evaluation harness[13] ($\approx 5k$ звёзд) и FastChat[14] ($\approx 35k$ звёзд).
  - LM evaluation harness предоставляет удобный механизм автоматической валидации на заданиях, которые требуют односложный ответ (число, слово или словосочетание);
  - FastChat позволяет автоматически валидировать модель на заданиях, требующих большой связный текст в качестве ответа.
- Для поддержки микросервисности, БД и скрипт для валидации БЯМ на бэкенде запускаются в отдельных Docker контейнерах.

[11] https://huggingface.co/spaces
[12] https://www.gradio.app/
[13] https://github.com/EleutherAI/lm-evaluation-harness
[14] https://github.com/lm-sys/FastChat

3.2.3. Детали реализации системы

В бенчмарке LM Evaluation Harness каждый валидационный набор данных имеет свой конфигурационный файл, в котором определяются метрики, используемые для валидации, и параметры запуска БЯМ на заданиях из этого набора данных. Для валидации БЯМ на собранном мной наборе данных в форк LM Evaluation Harness[15] были добавлены соответствующие конфигурационные файлы.

Мною были заданы следующие метрики для валидации БЯМ на наборе заданий из государственных экзаменов, требующих односложный ответ:

- Для заданий, в которых надо выбрать один вариант ответа, использовалась метрика accuracy, которая считает процент заданий, в которых БЯМ выбрала правильный вариант ответа, от всех заданий в наборе данных. Данная метрика использовалась в бенчмарке MMLU [4];

- Для заданий, ответом на которые является число, слово или словосочетание, использовалась метрика Exact match, которая считает, что задание выполнено правильно, только тогда, когда ответ БЯМ полностью совпадает с правильным ответом на задание. Такая метрика используется в бенчмарке SQuAD [5];

- Для заданий, в которых надо выбрать несколько вариантов ответа, метрика Exact Match была модифицирована так, что порядок цифр, означающих варианты ответа в задании, не учитывался;

В FastChat валидация на заданиях, требующих связный текст в качестве ответа, происходит через метод LLM Judge. Его суть в том, что после того, как валидируемая БЯМ сгенерировала ответ на задание, этот ответ передаётся на оценивание другой БЯМ-судье (желательно наиболее эффективной в индустрии на данный момент, например — GPT-4). БЯМ-судья в ходе оценивания выставляет баллы за ответ ва-

[15] https://github.com/deepvk/lm-evaluation-harness/tree/goat

лидируемой БЯМ на задание. Таким образом, FastChat позволяет валидировать БЯМ даже на заданиях, требующих целое сочинение в качестве ответа.

Мною в форке FastChat[16] был добавлен промпт для БЯМ-судьи на моём наборе данных. Этот промт включает в себя текст задания, ответ модели на задание и таблицу с официальными критериями ФИПИ по оценке данного задания. Таким образом, БЯМ-судья оценивает ответ на задание в соответствии с принятыми комиссией критериями, на подобии реальной оценки ответов учеников во время ЕГЭ и ОГЭ.

На данный момент в бэкенде с помощью FastChat происходит только генерация промптов для БЯМ-судьи для моего набора данных. Сами оценки не генерируются автоматически, так как использование достаточно хорошей БЯМ-судьи (в данной работе используется GPT-4) не является бесплатным. Пользователь системы должен запросить подтверждение у сообщества Vk Lab[17] на оценку ответов своей БЯМ на заданиях, требующих генерацию связного текста. После одобрения этого запроса сообществом, ответы БЯМ пользователя будут оценены с помощью GPT-4 и полученные оценки будут добавлены в лидерборд веб-приложения.

Важно уточнить, что если пользователь хочет валидировать свою модель только на заданиях, требующих односложный ответ, то ему не нужно писать запрос сообществу VK Lab и оценки его БЯМ будут получены и отображены в лидерборде полностью автоматически.

[16] https://github.com/deepvk/FastChat/tree/goat
[17] https://vk.com/lab

4. Эксперимент

С помощью созданной системы были провалидированы три популярны open-source БЯМ. Каждая выбранная модель повсеместно используется в индустрии. В Таблице 3 приведеных характеристики валидируемых моделей. Стоит отметить, что модель TinyLlama-1.1B-Chat-v1.0 весит в ≈8 раз меньше, чем две другие, поэтому от неё ожидается меньшая эффективность работы.

Модель	Дата выхода	Количество параметров	Скачивания за месяц
Mistral-7B-Instruct-v0.2	12.2023	7.24 B	≈2 M
Llama-2-7b-chat-hf	07.2023	6.74 B	≈1.2 M
TinyLlama-1.1B-Chat-v1.0	12.2023	1.1 B	≈600 K

Таблица 3: Характеристики валидируемых моделей

От моей системы ожидается, что результаты валидации данных моделей будут сопоставимы с результатами валидации, полученными через Open LLM Leaderboard.

Из рисунка 5 и таблицы 4 видно, что распределение оценок на заданиях, требующих выбрать один правильный вариант ответа из четырёх, схоже с распределением оценок из Open LLM Leaderboard. Также стоит отметить, что на заданиях, требующих выбрать нескольких правильных ответов из списка, Mistral справился примерно в четыре раза лучше, чем Llama. Тем временем, TinyLlama очень плохо справилась с не single choice заданиями, что не является удивительным, так как в Open LLM Leaderboard на валидационных наборах данных, подразумевающих генерацию более одной цифры (например, GSM8k), TinyLlama имеет точность 2 %, что схоже с оценками моей системы.

Модель	Средняя оценка
Mistral-7B-Instruct-v0.2	66 %
Llama-2-7b-chat-hf	52 %
TinyLlama-1.1B-Chat-v1.0	37 %

Таблица 4: Результаты валидации выбранных ЯМ в Open LLM leaderboard

Рис. 5: Результаты валидации выбранных ЯМ в моей системе

Таким образом, с помощью созданной системы были сравнены три популярные open-source БЯМ. Полученные с помощью системы оценки удовлетворили ожиданиям.

Заключение

По итогам данной работы были достигнуты следующие результаты:

- В ходе обзора предметной области были рассмотрены системы для облегчённой валидации ЯМ. Их преимущества и недостатки были проанализированы и в результате были сформированы требования, которым моя система для валидации должна удовлетворять. Также в результате анализа популярных валидационных наборов данных были сформированы требования к набору данных для моей системы;
- Был собран набор данных, состоящий из 6211 заданий государственных экзаменов. Все извлечённые задания были разбиты на 3 типа, в зависимости от способа взаимодействия с ЯМ;
- Было создано веб-приложение на HuggingFace для автоматической валидации ЯМ на собранном наборе данных, удовлетворяющее сформированным ранее требованиям;
- Три популярных open-source моделей были оценены с помощью созданной системы. Результаты валидации удовлетворили ожиданиям;

Код проекта доступен в репозитории: https://github.com/deepvk/goat/tree/leaderboard.

Список литературы

[1] Attention Is All You Need / Ashish Vaswani, Noam Shazeer, Niki Parmar et al. // CoRR. — 2017. — arXiv : 1706.03762.

[2] OpenAI, :, Achiam Josh et al. GPT-4 Technical Report. — 2023. — 2303.08774.

[3] Judging LLM-as-a-judge with MT-Bench and Chatbot Arena / Lianmin Zheng, Wei-Lin Chiang, Ying Sheng et al. // CoRR. — 2023. — Vol. abs/2306.05685. — arXiv : 2306.05685.

[4] Measuring Massive Multitask Language Understanding / Dan Hendrycks, Collin Burns, Steven Basart et al. // CoRR. — 2020. — 2009.03300.

[5] SQuAD: 100, 000+ Questions for Machine Comprehension of Text / Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang // CoRR. — 2016. — Vol. abs/1606.05250. — URL: http://arxiv.org/abs/1606.05250.