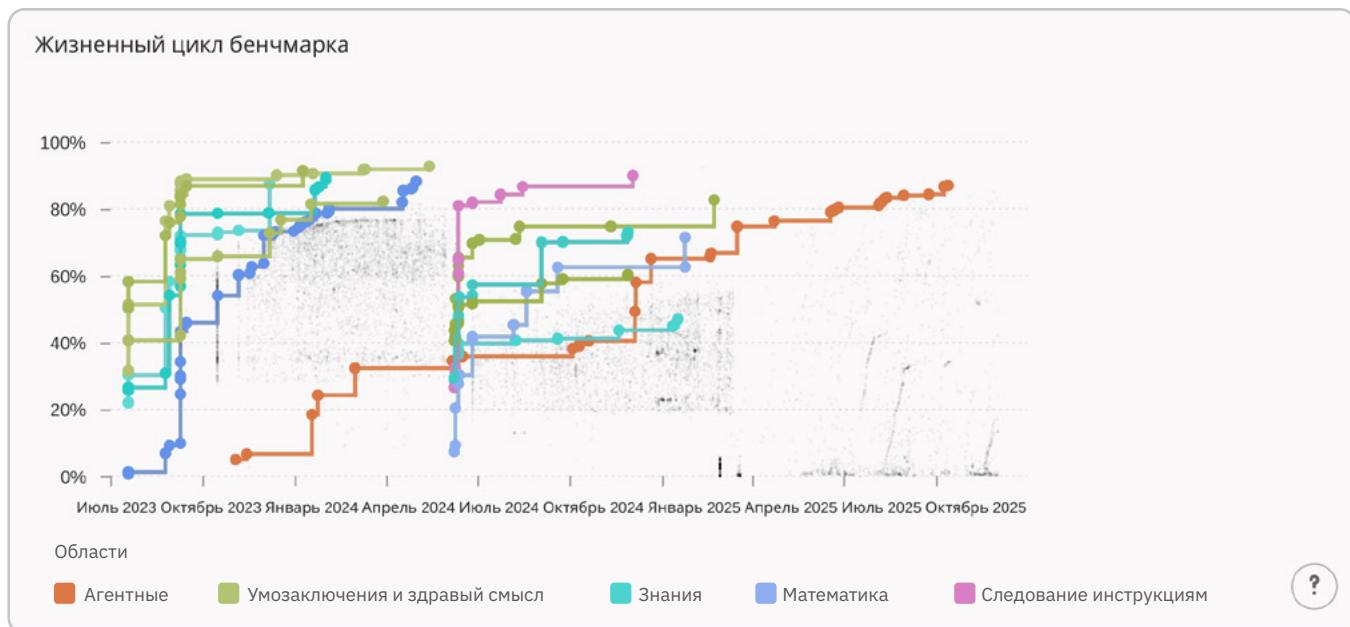


Руководство по оценке больших языковых моделей



Hugging Face

Всё, что вы хотели бы знать
об оценке больших языковых моделей
на основе нашего опыта оценки
15 000 моделей за 3 года

[Переведено - t.me/aivkube](#)

АВТОРЫ

Клемантин Фуррье
Тибо Фрер
Гильерме Пенедо
Томас Вулф

ОРГАНИЗАЦИЯ

Hugging Face

ДАТА ПУБЛИКАЦИИ

3 декабря 2025

[Переведено - t.me/aivkube](#)

Содержание

Что такое оценка модели?	4
ПЕРСПЕКТИВА СОЗДАТЕЛЯ МОДЕЛИ: СТРОЮ ЛИ Я СИЛЬНУЮ МОДЕЛЬ?	4
ПЕРСПЕКТИВА ПОЛЬЗОВАТЕЛЯ МОДЕЛИ: КАКАЯ МОДЕЛЬ ЛУЧШАЯ ДЛЯ ЗАДАЧИ <TASK>?	6
Основы больших языковых моделей для понимания оценки	7
ТОКЕНИЗАЦИЯ	7
Основы токенизации: почему и как мы токенизуем текст?	7
Как токенизация может испортить вашу оценку	9
ВЫВОД	11
Оценки логарифмических вероятностей	12
Генеративные оценки	14
Оценка на существующих бенчмарках	15
БЕНЧМАРКИ, КОТОРЫЕ СТОИТ ЗНАТЬ В 2025 ГОДУ	15
Разум и здравый смысл	15
Знания	16
Math	17
Код	18
Длинный контекст	19
Следование инструкциям	20
Вызов инструментов	20
Задачи для ассистента	22
Поиск информации в реальных условиях	22
Оценки на основе игр	23
Прогнозисты	24
Рекомендации	25
ПОНЯТИЕ СОДЕРЖИМОГО	26
Процесс создания данных	26
Проверка выборки	26
Задачи и метрики	27
Различная основа кода	28
Незначительные различия в реализации или загрузке	28
Разный промпт	29
АВТОМАТИЧЕСКИЙ ПОДБОР ЭФФЕКТИВНЫХ БЕНЧМАРКОВ ДЛЯ ОБУЧЕНИЯ МОДЕЛЕЙ	31
Монотонность	32
Низкий уровень шума	33
Неслучайная производительность	34
Метрики	36
Создание собственной оценки	38
НАБОР ДАННЫХ	38

Использование существующих данных	38
Использование человеческих аннотаторов	38
Создание датасета искусственно	41
Управление контаминацией	42
ВЫБОР ПРОМПТА	43
ВЫБОР МЕТОДА ВЫВОДА ДЛЯ ВАШЕЙ МОДЕЛИ	43
Основная проблема оценки: оценка свободного текста	45
АВТОМАТИЧЕСКИ	45
Метрики	45
Нормализация	47
Сэмплирование	48
Функциональные оценители	50
С УЧАСТИЕМ ЛЮДЕЙ	50
С МОДЕЛЯМИ-СУДЬЯМИ	53
Преимущества и недостатки использования judge-langs	53
Получение модели-судьи	55
Проектирование промпта для оценки	56
Оценка вашего оценщика	58
Советы и рекомендации	59
Что насчёт моделей награждения?	60
ОГРАНИЧЕНИЕ ВЫХОДНЫХ ДАННЫХ МОДЕЛИ	62
Использование промпта	62
Few shots и обучение в контексте	62
Структурированная генерация текста	62
Забытые дети оценки	63
СТАТИСТИЧЕСКАЯ ЗНАЧИМОСТЬ	63
СТОИМОСТЬ И ЭФФЕКТИВНОСТЬ	63
Заключение	65
БЛАГОДАРНОСТИ	66

Что такое оценка модели?

Погружаясь в мир больших языковых моделей — будь то обучение или дообучение собственных моделей, выбор модели для приложения или попытка понять текущее состояние области — вы наверняка столкнулись с одним вопросом:

«Как можно определить, является ли модель *хорошей*?»

Ответ — (что удивительно, учитывая тему блога) — оценка! Она повсюду: лидерборды с ранжированием моделей, бенчмарки, претендующие на *измерение логики, знаний, навыков программирования или математических способностей*, статьи с объявлениемами о новых передовых результатах...

Но что же такое оценка, на самом деле? И что она действительно может вам показать?

Это руководство поможет вам разобраться во всём: что такое оценка и что она не может сделать, когда стоит доверять тем или иным подходам (а также каковы их ограничения и смещения), как выбирать бенчмарки для оценки модели (и какие из них будут актуальны в 2025 году), а также как разработать собственную оценку, если вы этого захотите.

В ходе руководства мы также выделим распространённые ошибки, поделимся советами и приёмами от команды Open Evals и поможем вам научиться критически анализировать утверждения, основанные на результатах оценки.

Прежде чем углубиться в детали, давайте кратко рассмотрим, зачем проводится оценка — ведь кто вы и над чем работаете, определяет, какие именно оценки вам следует использовать.

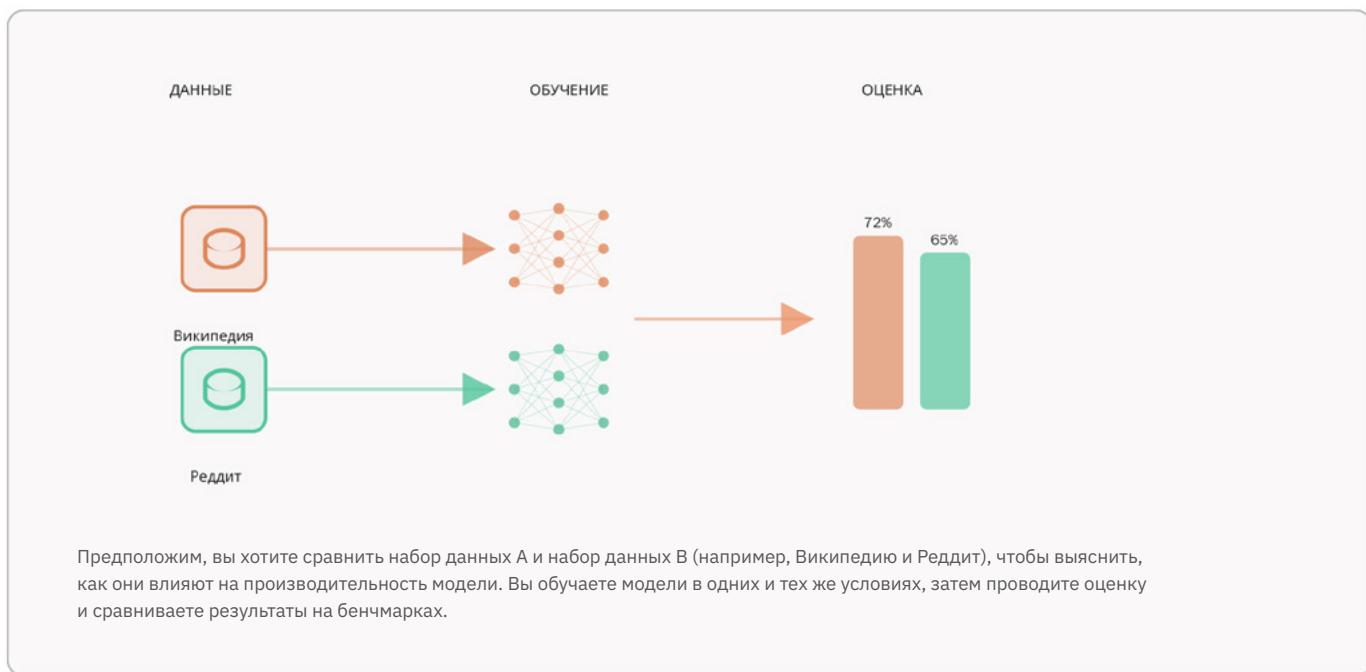
В этом руководстве мы сосредотачиваемся на оценках для языка (в основном естественного языка), но многие принципы также применимы и к другим модальностям

ПЕРСПЕКТИВА СОЗДАТЕЛЯ МОДЕЛИ: СТРОЮ ЛИ Я СИЛЬНУЮ МОДЕЛЬ?

Если вы исследователь или инженер, создающий новую модель, ваша цель, скорее всего, — построить сильную модель, которая демонстрирует хорошие результаты на наборе задач. Для базовой модели (обучение с нуля) вы хотите, чтобы модель хорошо справлялась с общими задачами, оценивая разнообразные способности. Если вы дообучаете базовую модель для конкретного применения, вас, вероятно, больше интересует производительность именно на этой задаче. В любом случае способ измерения производительности — через оценки.

Экспериментируя с разными архитектурами, смесями данных и рецептами обучения, вы хотите убедиться, что ваши изменения (выбор других данных для обучения, архитектуры, параметров и т. п.) не «сломали» ожидаемую производительность модели с такими характеристиками и, возможно, даже улучшили её. Способ проверить влияние различных проектных решений — абляции: абляция — это эксперимент, в котором обычно обучают модель в конкретных условиях, оценивают её на выбранном наборе задач и сравнивают результаты с базовой моделью. Таким образом, выбор задач оценки критически важен для **абляций**, поскольку они определяют, на что вы будете ориентироваться при создании модели.

Пример абляции



Для базовых моделей обычно выбирают стандартные задачи бенчмарков, используемые другими разработчиками (например, классический перечень бенчмарков, который всегда приводится при выпуске новой модели — мы рассмотрим их ниже). Для конкретного сценария использования можно либо применить существующие задачи оценки, если они доступны — и, скорее всего, стоит внимательно их изучить, если они не являются «стандартными» — либо разработать собственные (обсуждается ниже). Поскольку, вероятно, вы проведёте много аблаций, задачи оценки должны обеспечивать достаточно сильный сигнал (а не просто бессмыслицеские шумовые результаты) и выполняться экономно и быстро, чтобы вы могли проводить итерации с высокой скоростью. Кроме того, можно предсказывать производительность более крупных моделей, опираясь на результаты меньших, используя законы масштабирования.

Помимо аблаций для экспериментов, скорее всего, вам также захочется проводить оценки на промежуточных контрольных точках в процессе обучения модели, чтобы убедиться, что она корректно учится и улучшает результаты по разным задачам, а также не начинает деградировать из-за скачков или других проблем. В конце необходимо провести оценку финальной контрольной точки, чтобы объявить вашу модель SOTA при её выпуске.

Небольшое предупреждение

Несмотря на зачастую грандиозные заявления, для любой сложной способности мы пока не можем просто сказать «эта модель лучшая в этом», а должны говорить «эта модель лучшая на этих примерах для конкретной задачи, которую мы считаем адекватным приближением данной способности, без каких-либо гарантий».

(Тем не менее, вы всё равно можете заявлять о статусе SOTA, имея в виду это предупреждение.)

ПЕРСПЕКТИВА ПОЛЬЗОВАТЕЛЯ МОДЕЛИ: КАКАЯ МОДЕЛЬ ЛУЧШАЯ ДЛЯ ЗАДАЧИ <TASK>?

Вы хотите использовать модель, обученную кем-то другим для вашей конкретной задачи, без проведения дополнительного обучения, или возможно, вы планируете дополнительное обучение и ищете лучшую из существующих моделей для использования в качестве базы.

Для распространённых тем, таких как математика, программирование или общие знания, обычно существуют несколько таблиц лидеров с разными наборами данных, и, как правило, достаточно протестировать основных претендентов, чтобы выбрать подходящую модель (если они вам не подходят, маловероятно, что подойдут следующие по рейтингу модели).

Вы можете захотеть самостоятельно провести оценку и сравнения (используя уже существующие бенчмарки), чтобы получить более подробный анализ успехов и неудач модели, что мы рассмотрим ниже.

Так же, как разработчики моделей повышают конкретные возможности, для менее распространённых тем может потребоваться создание собственных оценок, что подробно рассмотрено в нашем последнем разделе.

Основные выводы

Разработчик модели: вам нужны быстрые, информативные бенчмарки, охватывающие интересующие вас области и навыки, которые можно многократно запускать во время аблейций.

Пользователь модели: вам нужны бенчмарки, соответствующие вашему конкретному применению, даже если для этого придётся создавать собственные.

В своей работе о выводах, сделанных из эпохи ImageNet в области бенчмаркинга и проектирования датасетов, авторы утверждают, что, поскольку метрики подвержены нестабильности, единственный надёжный способ оценивать модели — это ранжирование, и в особенности поиск широких групп оценок, которые дают согласованные и стабильные ранги.

Я считаю, что поиск устойчивости ранжирования действительно является чрезвычайно интересным подходом к оценке моделей, поскольку мы показали, что результаты LLM на автоматизированных бенчмарках крайне чувствительны к мельчайшим изменениям в промптах, а человеческие оценки не более согласованы — при этом ранжирование оказывается более стабильным при использовании устойчивых методов оценки.

А как насчёт измерения ИИ общего назначения (AGI)?

Нам остро не хватает хороших определений и структурных основ того, что такое интеллект для моделей машинного обучения, и как его оценивать (хотя некоторые пытались, например, Chollet в 2019 году и Hendrycks et al в этом году). Трудности с определением интеллекта — это не проблема, уникальная для машинного обучения! В исследованиях человека и животных также достаточно сложно дать чёткое определение, а метрики, пытающиеся предоставить точные оценки (например, IQ и EQ), вызывают горячие споры и остаются спорными — и это понятно.

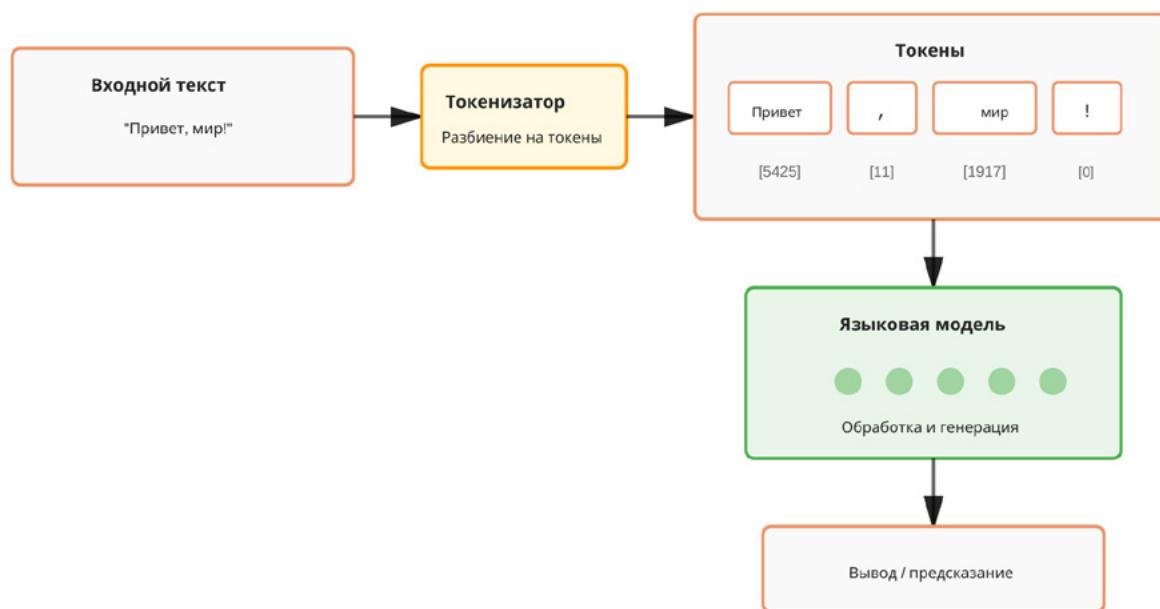
Однако существуют определённые проблемы с фокусировкой на интеллекте как на цели. 1) Интеллект часто оказывается движущейся целью, поскольку каждый раз, когда мы достигаем способности, считавшейся уникальной для человека, мы переопределяем этот термин. 2) Наши текущие рамки разрабатывались с учётом человека (или животного) и, скорее всего, плохо подходят для моделей, поскольку лежащие в их основе поведенческие паттерны и предпосылки различаются. 3) Это также довольно бессмысленная цель — следует ориентироваться на создание моделей, способных эффективно выполнять конкретные, чётко определённые и полезные задачи (например, бухгалтерия, отчётность и пр.), а не стремиться к AGI ради самой идеи.

Основы больших языковых моделей для понимания оценки

Теперь, когда вы ориентируетесь, почему оценка важна для разных пользователей, давайте рассмотрим, как мы формируем промпты для моделей, чтобы получить ответы и провести их оценку. Этот раздел можно просмотреть бегло, если вы уже занимались оценкой и главным образом ищете заметки и сноски.

В этом разделе мы рассмотрим два этапа для моделей: как входные данные предварительно обрабатываются перед передачей модели (**токенизация**), и каким образом модель генерирует из этого предсказание (**вывод**).

Если вы хотите узнать больше о том, как на самом деле обучать модель, вам стоит прочитать [*Smol Training Guidebook*](#)!



ТОКЕНИЗАЦИЯ

Входной текст (называемый промптом при выводе) сначала разбивается на токены — небольшие единицы текста (от одного или нескольких символов до уровня слова), каждой из которых соответствует уникальный числовой идентификатор. Полный набор токенов, способных к обработке моделью, называется её словарём.

Основы токенизации: почему и как мы токенизуем текст?

Поскольку большие языковые модели фактически представляют собой большие математические функции, они оперируют числами, а не текстом.

Предположим, вы хотите преобразовать предложение в числовое представление. Сначала нужно определить, как разбить предложение на маленькие части, а затем сопоставить каждой части число; это и называется токенизацией.

Раньше люди пытались сопоставить каждому символу текста его индекс в алфавите (`a -> 1, b -> 2` и так далее), что называется *токенизацией на основе символов* (*разбиение по символам*). С другой стороны, также пытались сопоставить каждому слову индекс в словаре (`a -> 1, aardvark -> 2, ab -> 3` и так далее), что называется *токенизацией на основе слов* (*разбиение по пробелам*, если в вашем языке есть пробелы — если нет, это сложнее).

Обе эти методики имеют серьёзное ограничение: они теряют информацию из исходного текста. Они стирают семантические связи, которые видны по форме слова (например: `dis similar`, `similar`, `similar ity`, `similar ly`), информацию, которую мы хотели бы, чтобы модель сохраняла, чтобы связывать родственные слова. (Кроме того, что произойдет, если в тексте вдруг появится совершенно новое слово? Для него не будет номера, и ваша модель не сможет его обработать 😞)

Некоторые люди предложили разбивать слова на подслова и присваивать индекс этим подсловам (`dis`, `similar`, `ity`, `ly`)!

Первоначально это выполнялось с помощью морфо-синтаксических правил (морфо-синтаксис – это как грамматика создания слов). В настоящее время большинство используют byte pair encoding (BPE) – умный статистический метод автоматического создания подслов на основе их частоты в эталонном тексте.

Итого: токенизация – это способ сопоставления небольших частей текста (одного или нескольких символов, вплоть до слова) с числами (аналогично индексу). Когда вы хотите обработать текст, ваш входной текст (называемый *промптом* при выводе) разбивается токенизатором на отдельные *токены*. Полный набор токенов, который может обрабатывать модель или токенизатор, называется их *словарём*.

📘 Далее: понимание токенизации

- ★ [Объяснение различных методов токенизации в курсе NLP](#)
- ★ [Концептуальное руководство по токенизации в документе](#)
- [Курс Юрафски по токенизации \(и другим темам\) – перейти к разделам 2.5 и 2.6](#)

📘 Дальнейшее изучение: Byte Pair Encoding

Настоятельно рекомендую ознакомиться с подробным объяснением работы BPE, поскольку это действительно основа современных больших языковых моделей.

- ★ [Объяснение BPE в NLP-курсе](#)
- [Статья о BPE \(для текста, так как метод существовал ранее в других областях\)](#)

Создание токенизатора требует принятия большего числа решений, чем можно было ожидать. Например, для токенизации чисел не стоит использовать простой BPE, но индексировать только цифры от 0 до 9 и предполагать, что все остальные числа состоят из комбинации этих цифр? Хотите ли вы хранить числа вплоть до, скажем, одного миллиарда, по отдельности?

Современные известные модели демонстрируют разные подходы к этому, однако пока не ясно, какой из них лучше подходит для поддержки математических рассуждений. Это повлияет на некоторую математическую оценку (и является причиной того, почему практически ни одна оценка не сводится к чистой арифметике).

Далее: токенизация чисел

- [Наглядное демо от Йенни Джун о том, как токенизаторы моделей Anthropic, Meta, OpenAI и Mistral разбивают числа](#)
- [Краткая история от Берена Миллиджа об эволюции токенизации чисел за последние годы](#)

Как токенизация может испортить вашу оценку

Управление дообученными моделями, системными промптами и шаблонами чата

До 2022 года модели просто проходили предварительное обучение: текст на вход, текст на выход, ничего больше. Затем в 2023 году появились instruction tuning и модели для чата, а в 2025 — модели для рассуждений. Это означает переход от использования чистого текста к всё большему использованию форматирования.



Это значит, что многие модели будут работать плохо, если вы не убедитесь, что:

1. вы соблюдаете формат, который ожидает модель
2. добавляете системный промпт в самом начале вывода, если это требуется вашей модели
3. Удалите следы мышления из ответов моделей рассуждения перед обработкой (обычно можно использовать regex для удаления содержимого между <think> тегами).

⚡ Критично: шаблоны чата и токенизация

Разные токенизаторы по-разному работают с пробелами и специальными токенами. Смотрите эту визуализацию, показывающую, как взаимодействуют пробелы, токенизация и шаблоны. Никогда не предполагайте, что токенизаторы ведут себя одинаково!

```
Llama-2
<s>☀️[INST]☀️What☀️is☀️2+2?☀️[/INST]☀️It's☀️4.☀️</s>

Mistral-1
<s>☀️[INST]☀️What☀️is☀️2+2?☀️[/INST]It's☀️4.</s>☀️

Mistral-3
<s>[INST]☀️What☀️is☀️2+2?☀️[/INST]It's☀️4.</s>

Mixtral
<s>☀️[INST]☀️What☀️is☀️2+2?☀️[/INST]It's☀️4.</s>
```

Внимание к токенам начала и конца предложения

Некоторые предобученные модели, например Gemma очень чувствительны [к включению токенов начала предложения](#) при выводе. Вам, возможно, придётся провести несколько экспериментов, чтобы проверить, происходит ли это в вашем случае, и добавить эти токены вручную при оценке, если их нет в вашем наборе данных.

Вы также можете столкнуться с проблемой, когда модель не останавливается на токене конца предложения, как ожидалось. Кодовые модели обычно обучаются с `\n\t` как единым токеном. Это означает, что при генерации текста модель часто будет генерировать `\n\t` за один шаг. Задача, которая определяет `\n` как токен конца предложения (= для остановки генерации), позволит модели продолжать генерацию после `\n\t`, если он предсказан как единый токен, поскольку это не то же самое, что `\n`. Однако вы всё же захотите, чтобы модель остановилась. В таких случаях необходимо либо обновить токены конца предложения, либо реализовать механизм отката к символному представлению последних токенов для остановки (и последующего обрезания) генерации.

Многоязычность и токенизация

При рассмотрении многоязычных оценок вы столкнётесь с двумя проблемами.

Во-первых, поскольку некоторые языки не всегда используют пробелы в качестве разделителей слов (корейский, тайский, японский, китайский — чтобы привести несколько примеров), для правильного разбиения требуется использование специализированных токенизаторов для каждого языка, иначе это повлияет на показатели таких метрик, как [BLEU](#), F1-метрика и другие.

Кроме того, токенизаторы в целом могут работать неравноправно по отношению к неанглоязычным языкам. При обучении BPE-токенизатора используются данные разных языков, которые вы хотите покрыть, но чаще всего эти данные не сбалансированы между языками (например, английского на порядки больше, чем тайского или бирманского). Поскольку BPE-токенизаторы создают словарь токенов на основе наиболее часто встречающихся слов, большинство длинных токенов будут представлять английские слова, а слова из менее частых языков чаще разбиваются только на уровне символов. Этот эффект приводит к несправедливости в многоязычной токенизации: некоторые (менее частые или с низким ресурсом) языки требуют на порядки больше токенов для генерации предложения эквивалентной длины по сравнению с английским.

All languages are NOT created (tokenized) equal!

This project compares the tokenization length for different languages. For some tokenizers, tokenizing a message in one language may result in 10-20x more tokens than a comparable message in another language (e.g. try English vs. Burmese).

This is part of a larger project of measuring inequality in NLP. See the original article: [All languages are NOT created \(tokenized\) equal](#) on [Art Fish Intelligence](#).

Data Visualization

Tokenizer

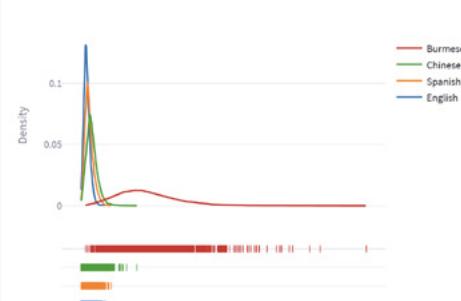
openai/gpt4

Tokenized using [tiktoken](#)

Median Token Length for [openai/gpt4](#)

English	Spanish	Chinese	Burmese
7	9	12	72

Token Distribution



Очень хороший демонстрационный пример от Йенни Джун о проблемах токенизации в разных языках

Если вы находитесь в такой ситуации, количество токенов, которое модель может сгенерировать для оценки, также должно зависеть от языка, поскольку разные языки токенизируются в разное число токенов.

Дополнительно: язык и токенизация

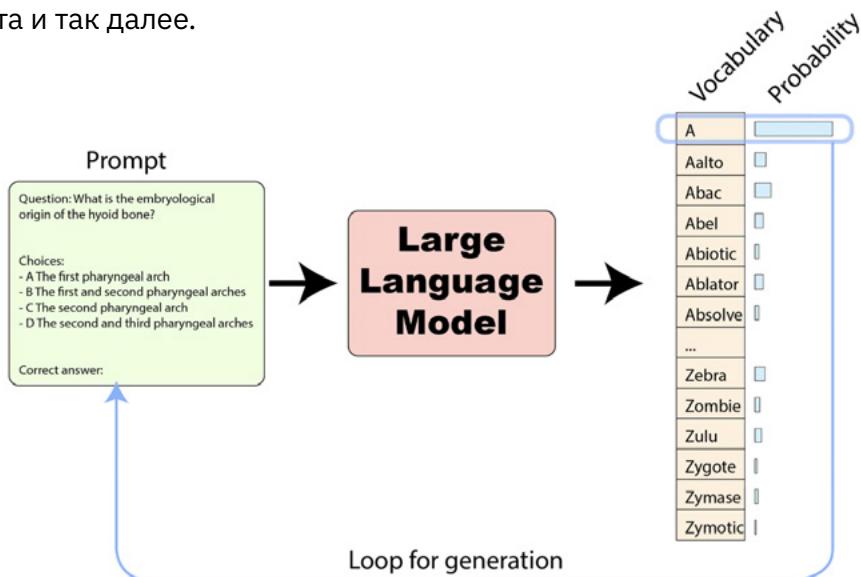
★ Прекрасный разбор и демонстрация от Йенни Джун по проблемам токенизации в разных языках: разбор сам по себе очень нагляден, а встраиваемое пространство взято из её работы. <https://www.artfish.ai/p/all-languages-are-not-created-tokenized>

★ Демонстрация от Александра Петрова о несправедливости токенизации: рекомендую посмотреть Compare tokenization of sentences, чтобы почувствовать разницу в стоимости инференса в зависимости от языка. <https://aleksandarpetrov.github.io/tokenization-fairness/>

ВЫВОД

Теперь, когда мы знаем, как преобразовать исходный текст в формат, который большие языковые модели могут обработать, давайте рассмотрим, как модели обрабатывают этот текст.

Из этого входного текста большая языковая модель генерирует вероятностное распределение наиболее вероятных следующих токенов по всему словарю. Чтобы получить продолжение генерации, мы можем взять самый вероятный токен (с некоторой долей случайности для более интересных результатов) в качестве следующего, затем повторять операцию, используя новый токен как конец промпта и так далее.



Два основных подхода к оценке

Оценки по логарифму вероятности: учитывая промпт и один (или несколько) ответов, какова вероятность данного ответа (ответов) для моей модели?

Генеративные оценки: учитывая промпт, какой текст генерирует моя модель?

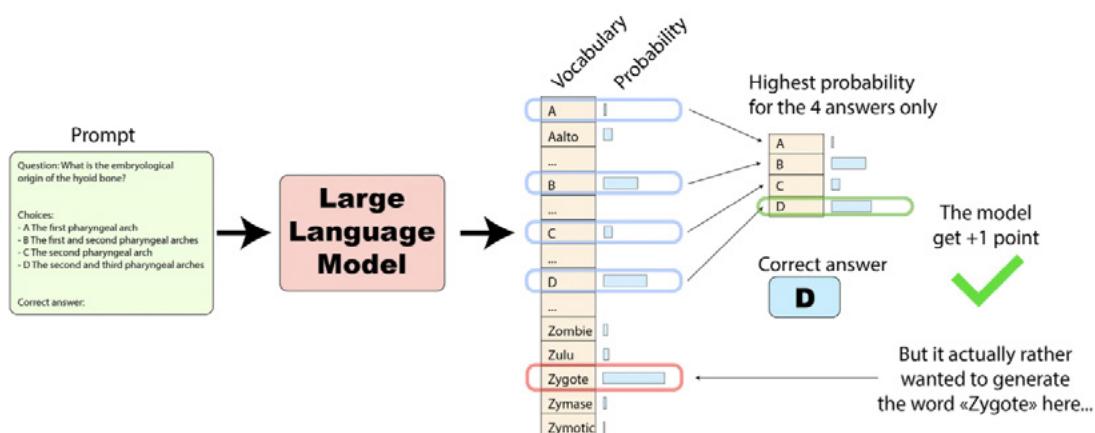
Выбор зависит от вашей задачи (как мы увидим далее) и от вашей модели: большинство моделей через API не возвращают логарифмы вероятностей, поэтому вам придется систематически использовать генеративные оценки для их оценки.

Оценки логарифмических вероятностей

Для оценок на основе логарифмов вероятностей нам нужна условная вероятность одного или нескольких вариантов, заданных промптом — другими словами, какова вероятность получить определённое продолжение при заданном входе?

Итак:

- мы конкатенируем каждый вариант с промптом и передаём их в нашу большую языковую модель, которая выдаёт логиты для каждого токена в зависимости от предыдущих
- мы оставляем только последние логиты (соответствующие токенам варианта) и применяем $\log \text{softmax}$, чтобы получить логарифмы вероятностей (где диапазон равен $[-\infty, 0]$ вместо $[0-1]$)
- затем мы суммируем логарифмы вероятностей всех отдельных токенов, чтобы получить общую логарифмическую вероятность варианта в конце
- мы можем применить нормализацию с учётом длины варианта



Это позволяет использовать одну из следующих метрик:

- Получить предпочтительный ответ модели среди нескольких вариантов, как на изображении выше. (Однако это может дать преимущество оценкам моделей, которые свободно сгенерировали бы что-то другое, например *Zygote* на изображении.)
- Проверить, имеет ли отдельный вариант вероятность выше 0,5.
- Изучить калибровку модели. Хорошо откалиброванная модель — это модель, для которой правильные ответы имеют наивысшие вероятности.

Чтобы узнать больше о калибровке, вы можете ознакомиться с [этой статьёй](#) от Anthropic — о том, что это такое, как её обнаруживать и как обучать модели быть хорошо откалиброванными, а также с [этой работой](#), посвящённой возможным ограничениям калибровки.

Ответ на вопрос с выбором из нескольких вариантов также можно представить как генеративную оценку в свободной форме! По этой причине вы иногда увидите упоминание **формулировки** задачи.

Существует три распространённые формулировки задачи:

- **Формат с выбором из нескольких вариантов (MCF):** мы сравниваем вероятность индексов вариантов, где варианты явно представлены в промпте и префиксированы A/B/C/D (как в MMLU).
- **Формулировка с пропусками (CF):** мы сравниваем вероятность различных вариантов без их явного указания в промпте.
- **Свободная генерация (FG):** мы оцениваем точность каждой генерации для данного промпта.

FG требует значительных скрытых знаний и обычно слишком сложна для моделей на этапах краткосрочной абляции после обучения. По этой причине мы обычно фокусируемся на формулировках с выбором из нескольких вариантов (MCF или CF) при проведении небольших абляций. Однако для моделей после дообучения FG становится основной формулировкой, поскольку мы оцениваем, способна ли модель действительно генерировать полезные ответы. Однако исследования также показали, что модели испытывают затруднения с MCF на ранних этапах обучения, осваивая этот навык только после значительного объёма обучения, что делает CF более подходящим для ранних сигналов. Поэтому мы рекомендуем использовать CF для небольших абляций и интегрировать MCF в основной запуск, поскольку он обеспечивает более качественный сигнал в середине обучения, когда модель преодолевает определённый порог и достигает достаточно высокого отношения сигнал/шум для MCF. Краткое замечание также о том, что для оценки модели В оценках логарифма правдоподобия последовательных ответов, таких как CF, точность рассчитывается как процент вопросов, для которых правильный ответ имеет наивысшую логарифмическую вероятность, нормализованную по количеству символов или токенов. Эта нормализация предотвращает смещение в пользу более коротких ответов.

Точка, в которой MMLU MCF начинает отклоняться от случайного уровня, зависит от размера модели и объёма обучающих данных. Для трансформера на 7B параметров в статье OLMES было показано, что модель начинает демонстрировать неслучайные результаты после 500 млрд токенов. Для модели на 1.7B параметров мы обнаружили, что это происходит после 6 трлн токенов в SmoLLM2.

Всегда ли следует токенизировать контекст вместе с вариантами?

При оценке задач множественного выбора (MCQA) в общем случае желательно токенизировать контекст вместе с вариантами ответа, поскольку это создаёт последовательность токенов, которая выглядит для модели естественной.

Однако некоторые токенизаторы (например, токенизатор Llama) не удовлетворяют свойству $\text{tok}(\text{context} + \text{choice}) = \text{tok}(\text{context}) + \text{tok}(\text{choice})$ и могут добавлять или убирать пробелы. Это означает, что сравнивать только логвероятности вариантов ответа непросто, поскольку токены контекста могут «перетекать» в них, искажая сравнение. Чтобы привести конкретный пример, пусть у вас есть символы C1, C2 и C3 как базовые токены словаря, и при этом последовательность C1C2 также является отдельным токеном, выученным во время BPE.

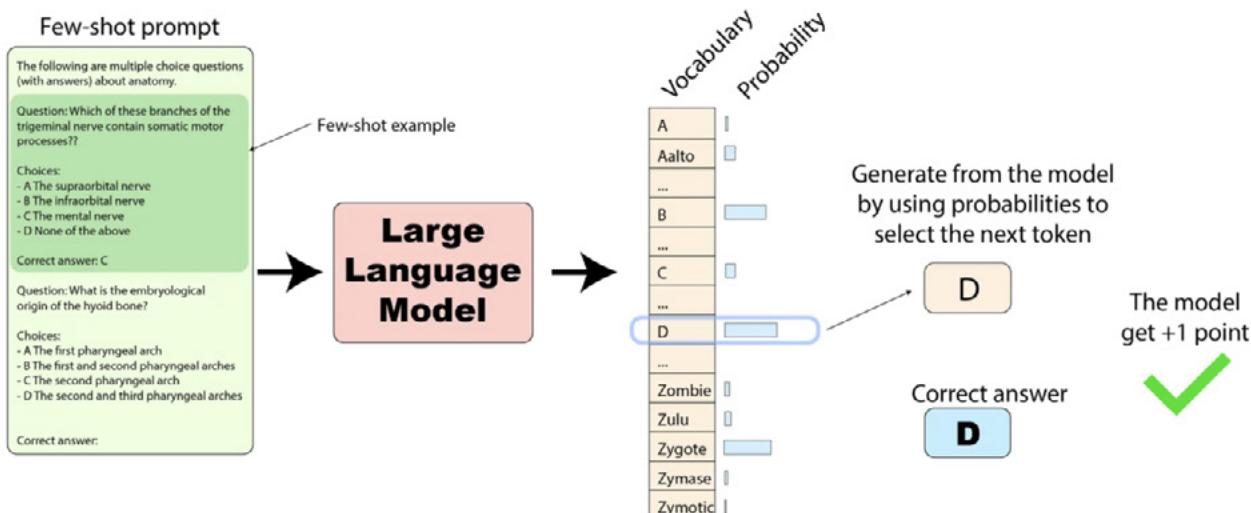
Пусть ваш контекст — это C1, а варианты — C2 и C3. Если токенизировать контекст вместе с вариантами, вы сравниваете C1C2 (один токен) с C1 + C3 (два токена). Даже если нормализовать логвероятности по длине, вы сравниваете разные сущности.

Если же токенизировать контекст и варианты отдельно, вы сравниваете C1 + C2 и C1 + C3. Но поскольку C1C2 является отдельным токеном, то последовательность C1 + C2 может встречаться редко в данных, на которых обучался энкодер, что делает её маловероятной для модели — и это может испортить ваши логвероятности. Если у вашей модели наблюдается такая проблема, обычно приходится выбирать «наименее плохой» вариант: сравнивать сопоставимые последовательности. Для этого токены контекста и варианта вычисляют отдельно, затем конкатенируют, предварительно удалив специальные токены начала/конца последовательности, которые могли быть добавлены.

Генеративные оценки

Для генеративной оценки нам необходим текст, сгенерированный моделью по заданному промпту.

Он получается автогрессивным способом: мы передаём промпт модели, выбираем наиболее вероятный следующий токен как «первый токен выбора» модели, затем повторяем процесс, пока не достигнем условия завершения генерации (максимальная длина, специальный токен для остановки генерации и т.п.). Все токены, сгенерированные моделью, считаются её ответом на промпт.



Затем мы можем сравнить эту генерацию с эталонами и оценить расстояние между ними, используя либо простые метрики, такие как точное совпадение, более сложные метрики, например BLEU, либо модели в роли судей.

Далее

★ [Блог о различных способах оценки MMLU](#), подготовленный моей командой из Hugging Face. Рекомендую прочитать его, если хотите глубже понять различия между оценками логарифма правдоподобия для многовариантного выбора и генеративными оценками, а также их влияние на изменение баллов (приведённые выше иллюстрации взяты из блога и созданы Томом Вульфом).

★ [Изящная математическая формализация описанных выше методов вывода](#) от EleutherAI. Перейдите сразу к приложению.

Оценка на существующих бенчмарках

Теперь, когда вы (снова) познакомились с основами токенизации и вывода, а также поняли нюансы оценки, давайте перейдём к реальному бенчмаркингу! Сначала мы сделаем небольшой обзор оценок 2025 года, затем обсудим, на что обращать внимание в бенчмарках и почему, скорее всего, вы не сможете воспроизвести опубликованные результаты. Наконец, мы рассмотрим особый случай выбора подходящего бенчмарка для оценки обучения вместе с командой FineWeb.

⚠️ Важные концепции

В этом разделе часто встречаются два понятия: контаминация и насыщение.

Насыщение — это момент, когда производительность модели на бенчмарке превосходит уровень человека. В более общем смысле этот термин применяется к наборам данных, которые уже не считаются полезными, так как они утратили способность различать модели.

Если все модели показывают результаты, близкие к максимально возможным на вашей оценке, такой бенчмарк перестаёт быть дискриминационным. Это похоже на оценку старшеклассников по задачам начальной школы: успех не говорит ни о чём (хотя неудача информативна).

Контаминация — ситуация, когда данные для оценки случайно попали в обучающий набор моделей, в результате чего показатели моделей искусственно завышены и не отражают реальную эффективность на задаче.

Это напоминает оценку студента по вопросам, которые он уже заранее знает.

Это то, что вы видите на изображении в баннере!

БЕНЧМАРКИ, КОТОРЫЕ СТОИТ ЗНАТЬ В 2025 ГОДУ

Вы можете оценивать **отдельные способности** по отдельности — это обычно интересно для получения сигналов во время обучения или при сравнении базовых и предобученных моделей. (Однако если вы выбираете и проверяете методики обучения с помощью следующих оценок, отчёты на их основе по итоговой модели будут несколько искажены, так как вы уже адаптировали обучение под хорошие результаты по этим оценкам).

Не стесняйтесь быстро просмотреть этот раздел, если вы ещё не очень знакомы с оценкой, и вернитесь к нему, когда потребуется найти набор данных для конкретной способности :)

Разум и здравый смысл

Наборы данных для тестирования рассуждений и здравого смысла часто являются «историческими», созданными в эпоху BERT и моделей с эмбеддингами, до появления бума больших языковых моделей. Раньше они были достаточно сложными (особенно потому, что часто создавались с учётом противодействия моделям того времени), но теперь они 1) слишком просты, 2) загрязнены или насыщены и должны использоваться только для абляций или оценки на этапе

предобучения. Крупные наборы данных иногда содержат ошибки или вопросы низкого качества, поскольку зачастую создавались через Amazon Mechanical Turk для быстрого и недорогого масштабирования (что теперь выполняется с помощью больших языковых моделей, генерирующих вопросы для оценки).

[ARC](#) (2018) (не путать с ARC-AGI) — это набор вопросов множественного выбора по естественным наукам для начальной школы, созданный на основе человеческих тестов. Варианты были выбраны с адвокатской целью для систем, основанных на кооккуренции слов, на тот момент. Он содержит несколько подмножеств более высокого качества вызов одно из которых используется до сих пор для предварительного обучения. [WinoGrande](#) (2019) — краудсорсинговый (Mechanical Turk) датасет для разрешения местоимений и заполнения пропусков, + валидация) датасет для разрешения местоимений и заполнения пропусков, использующий адвокатские пары элементов для обмана моделей. Оба этих набора данных были достаточно сложны для моделей до 2022–2023 годов.

Ряд исторических наборов данных специально посвящён оценке рассуждений, требующих некоторого здравого смысла и контекстной привязки. [HellaSwag](#) (2019) требует от больших языковых моделей выбрать правильное следующее предложение из списка адвокатских вариантов, где текст взят из подписей к видео ActivityNet и обучающих материалов Wikihow. (Это продолжение набора данных под названием Swag). Поскольку большинство предложений взяты из описаний действий или инструкций, для решения задачи часто требуется физический здравый смысл. В том же духе, [CommonsenseQA](#) (2018) — датасет вопросов с множественным выбором на основе здравого смысла, созданный на базе ConceptNet — аннотаторы формулируют вопросы и затем используют концептуально близкие отвлекающие варианты в качестве ответов. [PIQA](#) (2019) специально фокусируется на вопросах физического здравого смысла (созданных на основе примеров с [Instructables.com](#), с использованием сложных вариантов, полученных путём семантических искажений или переформулировок). [OpenBookQA](#) (2018) предоставляет открытые факты для помощи в ответах на вопросы с множественным выбором — однако для их решения также требуется скрытое знание здравого смысла.

Более современный интересный датасет для рассуждений — [Zebra Logic](#), который использует логические задачи для проверки способностей моделей к рассуждению. Их методы позволяют генерировать задачи бесконечно, что минимизирует загрязнение данных.

Знания

Основным датасетом для оценки знаний был [MMLU](#) (2020). Он достиг насыщения и загрязнения, и после более детального анализа были выявлены ряд проблем: неполные вопросы с ссылкой на отсутствующие документы, некорректные эталонные данные, неоднозначные вопросы и явная американоцентричность выбранных тем. Таким образом, он был очищен в [MMLU-Redux](#) (2024), расширен более сложными вопросами и большим количеством ответов в [MMLU-Pro](#) (2024, основной версии, используемой сообществом в настоящее время), а также переведен и аннотирован с учётом культурных искажений в [Global-MMLU](#) (2024). Они в основном используются для оценок и аблайций на этапе предпрограммного обучения.

Для постобучения исследователи обращаются к более сложным и высококачественным наборам знаний. GPQA (2023) — набор вопросов уровня PhD по биологии, химии и физике, подготовленных так, чтобы на них могли ответить аспиранты соответствующих областей, и не иначе. Наиболее используемый под-

набор — алмазный one, но с момента его публикации в 2023 году он также стал подвергаться контаминации.

И, наконец, помпезно названный, но очень качественный [Humanity's Last Exam](#) (2024) содержит 2,5 тысячи вопросов, собранных посредством краудсорсинга экспертами различных областей. Он преимущественно закрытый, а вопросы требуют как сложных знаний, так и развитых навыков рассуждения. Он ещё не сломан, и, на мой взгляд, это отличный набор данных. Единственная проблема в том, что, поскольку нельзя быстро оценить модель, сейчас люди проводят оценку с помощью судьи больших языковых моделей, который анализирует ответы, вместо проверки по эталонным данным, поэтому это одна из тех оценок, где в реальных условиях получаются действительно несопоставимые результаты.

Тем не менее, хотя тестирование моделей на качество их скрытых знаний имело большой смысл пару лет назад (и остаётся актуальным при обучении для проверки качества модели, с оценками типа MMLU-Pro во время предпродажи и GPQA/HLE после обучения), я считаю, что в ближайшие годы мы постепенно откажемся от таких бенчмарков по двум причинам.

1. Вопросы становятся всё более непостижимыми для людей: они настолько сложны, что неспециалистам почти невозможно понять, что означает результат по каждому вопросу, а также трудно убедиться, что сами наборы данных не содержат ошибок.
2. Теперь, когда наши модели подключены к инструментам, таким как доступ в интернет, оценки скрытых знаний всё чаще превращаются в оценки веб-поиска и извлечения информации, поэтому теряют смысл в прежнем виде. Проще говоря, мы переходим от оценок с закрытой книгой к оценкам с открытой книгой. Для сравнения, во французской системе образования получают экзамены с закрытыми книгами в средней школе, но при поступлении в университет часто предполагается доступ к базам данных и интернету, и оценки становятся меньше о том, что вы запомнили, и больше о том, как вы рассуждаете, имея свободный доступ к информации. Я считаю, что этот сдвиг мы также увидим в оценке больших языковых моделей по мере роста их возможностей.

Math

Наборы данных для оценки по математике использовались как прокси для бенчмаркинга рассуждений и логики, помимо, конечно, проверки способности моделей решать математические задачи.

Два эталонных набора данных для математической оценки — это [GSM8K](#) (2021), содержащий задачи школьного уровня, и [MATH](#) (2021), агрегат олимпиадных задач, опубликованных в интернете, который в последние годы достиг насыщения и контаминации. Первый был расширен [GSM1K](#) (2024) — воссозданием с 1К новых задач для проверки, какие модели были заражены на предыдущем, [GSM-Plus](#) — переписыванием моделей с адверсарияльными изменениями (дистракторы, числовые вариации и т.д.) и [GSM-Symbolic](#) (2024) — менее используемой, но очень интересной переработкой GSM8K в виде шаблонов задач для предотвращения заражения: задачи могут генерироваться бесконечно.

Сообщество сейчас сосредоточено на использовании:

- Продолжений MATH, либо [MATH-500](#) (репрезентативный поднабор из 500 задач, отобранный для предотвращения переобучения), либо MATH-Hard (только 500 самых сложных задач).

- **AIME** ([24](#), [25](#)) — американские олимпийские датасеты для старшеклассников, использованные в оригинальном виде на момент публикации. Эти датасеты интересны тем, что задачи обновляются каждый год с эквивалентным уровнем сложности, что позволяет проверять заражение, сравнивая результаты при публикации с результатами датасета предыдущего года.
- **Math-Arena** — актуальная подборка соревнований и олимпиад, регулярно обновляемая (включает AIME25, а также множество других соревнований!)

Большинство этих наборов данных уже не считается «особо сложным», поскольку они ограничены уровнем начальной школы (хотя GSM-Symbolic позволяет генерировать задачи с большим числом уровней рекурсии, что делает их синтетически сложнее). С другой стороны спектра [FrontierMath](#) (2024) была попыткой предоставить значительно более сложные математические задачи, специально написанные математиками для этого случая. Датасет был теоретически приватным (но, как оказалось, OpenAI имела доступ к частям набора данных — очень жаль). [Humanity's Last Exam](#) (2025) (упомянутый в разделе знаний) также содержит интересные «созданные по случаю» математические задачи, требующие сложного рассуждения (в частности, доказательств теорем).

Лично я бы использовал AIME25 и MATH-500 для оценок до обучения, а Math-Arena — для оценок после обучения.

Код

Поскольку агенты должны взаимодействовать с инструментами, им необходимы навыки программирования: либо для прямого вызова инструментов, если они являются кодовыми агентами, либо для понимания того, как отлаживать вывод инструментов в случае проблем (для кодовых и json-агентов — см. различия [здесь](#)). Наборы для оценки кодирования также являются хорошими прокси для оценки рассуждений.

Исторически, в 2021 году, наборами для оценки кода были [MBPP](#) — 1K краудсорсинговых задач по Python начального уровня, [APPS](#) — 10K задач по генерации кода, собранных из собеседований и сайтов обмена программами, и [HumanEval](#), представленный вместе с моделью Codex, который, в отличие от предыдущих, состоял из «специально разработанных для релиза» задач, что тогда было очень удобно! Он также поставлялся с песочницей для предотвращения проблемного выполнения кода на машине оценщика. (Последним новшеством в этой статье стал оценщик для `pass@k`, который до этого рассчитывался путём буквальной проверки, успешна ли была оценка более k раз из n попыток).

Команда [EvalPlus](#) (2023) создала HumanEval+ и MBPP+, расширенные версии оригинальных наборов, добавив больше тестовых случаев и исправив ошибки в исходных данных, а также расширив входные данные. [EvoEval](#) (2024) представила вариацию HumanEval, семантически переписав задачи и добавив маркировку сложности.

Для итоговых моделей могут понадобиться более сложные или не содержащие загрязнений задачи.

[LiveCodeBench](#) (2024) использует похожий подход «сбора задач с сайтов LeetCode», но особенно интересен тем, что хранит дату задачи для сравнения производительности моделей на задачах, созданных до и после завершения обучения. Это был отличный бенчмарк без загрязнений, и я с нетерпением жду его обновления!

[AiderBench](#) (работает онлайн с конца 2024 года, насколько я знаю) также использует данные с существующих сайтов по программированию (в частности, Exercism), но выходит за рамки решения задач, сосредоточившись на тестировании редактирования и рефакторинга кода.

Для этапа после обучения нужны более комплексные оценки, и некоторые бенчмарки вышли за рамки оценки отдельных задач, поскольку они не отражали сложные навыки программирования. [RepoBench](#) (2023) проверяет системы автодополнения на уровне репозитория для Python или Java, используя код с Github. Он основан на маскировании случайных строк в кодовых базах с последующим запросом дополнений — как внутри файла, так и между файлами — и включает несколько уровней тестов (поиск, дополнение, их сочетание).

[SweBench](#) (2024) представляет собой более известную и полную версию этого подхода, также с использованием Github, но направленную на проверку способности моделей решать существующие задачи: понимание логики, межфайловое редактирование и выполнение, рассуждения на длинном контексте и прочее.

[CodeClash](#) (2025) — это арена для кодирования, где модели пишут код, который конкурирует с кодом других моделей, редактируется и совершенствуется в итерациях.

На данный момент рекомендуется следить за LiveCodeBench, AiderBench и высококлассным подмножеством SWE Bench (SWE-Bench verified), а также ознакомиться с [METR report](#) о реальной полезности кодовых ассистентов.

Длинный контекст

Для корректного взаимодействия с пользователями в ходе длительной беседы, не теряя нити разговора, необходим эффективный менеджмент длинного контекста. (Забавно думать, что всего три года назад максимальная длина контекста для моделей составляла 2048 токенов, тогда как сейчас мы в основном работаем с 128K и более.)

Оценка, начавшая тестирование этого в 2023 году, вероятно, называется [NIAH](#) (Needle in a Haystack), где вы помещаете случайный факт в длинный несвязанный текст и просите модель его извлечь. Это обеспечивает удобную структуру для оценки того, на каком участке контекста модель, скорее всего, забывает информацию и с какой длины контекста это происходит. В 2023 году модели плохо справлялись с этим, а к 2025 году проблема практически решена.

С тех пор появились более сложные методы расширения работы с длинным контекстом. [RULER](#) (2024) добавляет многоэтапное трассирование (требующее от модели следовать цепочкам переменных для получения правильного значения), изменение частоты слов и включает вариацию QA для NIAH. Сейчас эта задача тоже практически решена. [Michelangelo](#) (2024, иногда также называемый MRCR за многогранковый кореферентный анализ) использует синтетические данные с длинным контекстом: задачи различной длины проверяют, может ли модель точно воспроизвести уникальные части контекста (а также определить, присутствует ли релевантная информация) и понять последовательность модификаций текста. Затем он был расширен в [OpenAI MRCR](#) (2025). [InfinityBench](#) (2024) является многоязычным (английский и китайский) и предоставляет 100 тысяч токенов синтетических задач с разнообразными целями (вопросы-ответы, извлечение информации, как в NIAH, вычисления на очень длинном контексте и др.). InfinityBench по-прежнему даёт определённый сигнал.

[HELMET](#) (2024) объединяет задачи и существующие бенчмарки, формируя большой единый датасет с более выраженным сигналом: наборы данных RAG и QA (Natural Questions, TriviaQA, PopQA, HotpotQA, Narrative QA и InfinityBench), извлечение (RULER и JSONKV), генерация с цитированием (подмножества ALCE), суммирование, повторное ранжирование пассажей (MS MARCO), обучение с контекстом (TREC, NLU, Banking77, CLINIC150). Агрегации бенчмарков исчерпывающие, но существует риск двойного учёта: не стоит тестировать модель одновременно на HELMET и InfinityBench и затем агрегировать результаты, поскольку это эквивалентно повторному запуску одной и той же оценки! В 2025 году HELMET всё ещё обладает достаточной дискриминационной способностью для сравнения моделей.

Мои любимые идеи для оценки в длинном контексте — это [Novel Challenge](#) (2024), набор из 1000 истинных и ложных утверждений о вымышленных книгах, опубликованных в прошлом году (от читателей этих книг!), требующий прочтения и понимания полного текста для корректного ответа, и [Kalamang translation dataset](#) (2024), где моделям необходимо правильно переводить с английского на каламанг по изучению грамматической книги (Каламанг — язык с очень низкими ресурсами, без онлайн-присутствия и с всего 200 носителями). Набор данных для перевода Каламанг можно значительно расширить и на другие низкоресурсные языки (но было бы интересно использовать основанный на правилах грамматический проверщик для проверки корректности генерации, чтобы добиться строгой точности, вместо того чтобы полагаться на BLEU...).

Следование инструкциям

Два основных набора данных для следования инструкциям — [IFEval](#) (2023) и его расширение [IFBench](#) (2025). IFEval, на мой взгляд, одна из самых умных идей для оценки последних лет: моделям предлагается следовать форматированным инструкциям (относительно ключевых слов, пунктуации, числа слов и предложений, форматирования файлов, таких как markdown или html и др.). Каждое из этих условий можно проверить с помощью конкретного теста парсинга: это означает, что эта оценка — одна из редких оценок свободной генерации, в которых можно получить строгий балл без опоры на модель-судью.

В более общем смысле она относится к типу оценки функциональной корректности/юнит-тестов, который является моим личным предпочтением для оценки моделей. Также очень легко пересоздать или расширить, чтобы предотвратить загрязнение данных.

К слову, некоторые бенчмарки также проверяют «несоблюдение инструкций» (несоответствие): [CoCoNot](#) (2024) особенно исследует, будут ли модели выполнять или игнорировать неполные (недостаточно определённые/неясные), неразрешимые (из-за нехватки информации или антропоморфизации ИИ, часто вызывающей галлюцинации) либо небезопасные запросы. Для этого вручную создавались запросы, модели писали запросы с несоблюдением, после чего их фильтровали для составления тестового набора, представленного как задача классификации.

Вызов инструментов

Появление инструментов — одна из особенностей, которая начала переводить большие языковые модели в агентный режим.

[TauBench](#) (2024) оценивает модель по её способности отвечать на запросы пользователя в областях ритейла и авиаперевозок (заказ, бронирование, поиск товаров и т. д.). База данных имитирует реальные доменные данные с помощью

синтетических образцов, и модель считается корректной, если 1) её действия правильно обновили базу данных, и 2) она адекватно ответила пользователю. Для автоматизации этого бенчмарка пользователя имитирует большая языковая модель, что делает оценку достаточно затратной и склонной к ошибкам. Несмотря на эти ограничения, он пользуется широкой популярностью, особенно потому, что хорошо отражает реальные сценарии использования.

[ToolBench](#) (2023) требует вызова API (OpenWeather, Cat, HomeSearch, TripBooking, GoogleSheets, WebShop, Tabletop и других) для решения 100 тестовых случаев по всему набору данных, при этом для решения каждого может потребоваться от одного до десяти вызовов инструментов. Некоторые из этих API являются имитациями, а некоторые — реальными, что делает набор данных уязвимым к случайным сбоям. Поэтому это было исправлено и расширено в [StableToolBench](#) (2025), который вводит общий VirtualAPIServer, имитирующий всё для обеспечения стабильности оценки, однако при этом используется судья больших языковых моделей для проведения оценки, что создаёт дополнительный уровень смещения.

[BFCL](#) (2025, хотя бенчмарк фактически существует уже несколько лет) значительно эволюционировал за год и в текущей версии содержит 4 поднабора: одноступенчатые вызовы инструментов (simple tool calls), краудсорсинговые реальные вызовы функций от пользователей, многоповоротные диалоги (для проверки точности в длинных контекстах и ответах с вызовами инструментов) и агентные вызовы (веб-поиск, работа с памятью, взаимодействие с SQL-данными). Для оценки правильности вызовов используется комбинация абстрактных синтаксических деревьев, анализа ответа выполнения и сопоставления состояния (соответствует ли итоговое состояние ожидаемому). Основное внимание уделяется версии v3 для тестирования вызова инструментов, а версия v4 проверяет использование веб- и поисковых инструментов.

Наконец, с появлением MCP возникли бенчмарки для тестирования вызова инструментов, ориентированных на MCP – однако все они преимущественно полагаются на модель-судью и используют реальные API, что может приводить к потенциальным сбоям или проблемам с воспроизводимостью из-за сетевых неполадок (по всей видимости, дополнительная нагрузка на создателей сайтов не слишком велика, поскольку база пользователей большинства MCP достаточно велика).

[MCPBench](#) (2025) подключает большие языковые модели к живым, реальным серверам MCP (Wikipedia, HF, Reddit, Steam, arxiv и др.) с задачами, требующими нескольких ходов для решения (созданными синтетически). Оценка сочетает проверку корректности вызовов инструментов и успешности по заданным правилам с судьёй больших языковых моделей для определения, были ли запросы надлежащим образом обработаны.

[MCP-Universe](#) (2025) использует 11 серверов MCP по разнообразным реальным темам (навигация в реальной жизни, 3D-дизайн, веб-поиск и др.). Особенность этого подхода в том, что оценка опирается на несколько строгих проверяющих: один отвечает за корректность формата, а два — за правильность ответов. Поскольку задачи могут быть статическими (вопросы с неизменными ответами) или динамическими (например, количество github-старов в репозитории, погода и т. п.), в последнем случае правильность ответов оценивается с помощью задачезависимой системы выполнения, которая автоматически получает актуальный правильный ответ из соответствующего источника и сравнивает его с выводом модели. Это гораздо удобнее, чем полагаться лишь на судью больших языковых моделей!

[LiveMCPBench](#) (2025) предоставляет крупную локально развёртываемую коллекцию серверов MCP для проверки способности моделей различать инструменты для выполнения задач. Лучшие модели уже достигают 80% — мы близки к насыщению. Однако проверка того, могут ли модели выбирать правильные инструменты из очень длинных списков, является важным кейсом, который будет приобретать всё большее значение по мере развития веба в сторону MCP.

(Кстати, вот интересный [doc](#) о том, как создавать хорошие инструменты.)

Хотя тестирование отдельных возможностей даёт ценную информацию, реальная эффективность ассистента определяется тем, как эти возможности сочетаются вместе. Модель может превосходно рассуждать, но при одновременной интеграции рассуждений с вызовом инструментов и управлением длинным контекстом может возникать сбой, поэтому необходимы оценки, требующие координации нескольких возможностей одновременно.

Задачи для ассистента

Я убеждён, что **задачи для ассистентов** станут одним из ключевых способов проведения оценок следующего уровня: их решение требует сочетания множества возможностей (длинный контекст, рассуждения, вызов инструментов и др.), в то время как сами бенчмарки дают представление о производительности в конкретных доменах в полезных реальных условиях. Они также, как правило, более понятны (широкой аудитории), чем бенчмарки, ориентированные на конкретные возможности. Если бенчмарки достаточно общие, они не проверяют, какие именно инструменты были использованы, а оценивают корректность конечного результата, поскольку сложные задачи допускают несколько путей к успеху.

Поиск информации в реальных условиях

[GAIA](#) (2023) положила начало современной агентной оценке, требуя от моделей использовать комбинацию инструментов, рассуждений и поиска для решения реальных запросов (иногда с привлечением документов). Вопросы были разделены на три уровня: первый уже насыщен, а третий по-прежнему сложен для моделей. Это также один из тех бенчмарков, где представленные показатели будут распределены по методам оценки, поскольку одни исследователи отчитываются по публичному валидационному набору, а другие используют больших языковых моделей экспертов для оценки по приватному тестовому набору (если имеется публичная таблица лидеров [здесь](#)).

Позже это было воспроизведено в [BrowseComp](#) (2025), который проверяет то же самое (может ли модель найти адекватный ответ на конкретный запрос, используя инструменты и онлайн-информацию), однако не гарантирует уникальность результата, так как вопросы формировались, начиная с результата и строились под него с разными уровнями сложности: например, для конкретной статьи, которую нужно найти, вопрос создавался путем объединения информации о метаданных, например «какая статья по Теме была опубликована на Конференции с одним автором определённой национальности и двумя представителями Организации?»

Однако, вероятно, на данный момент бенчмарк также стал сложнее.

[GDPval](#) (2025) оценивает модели по 44 профессиям из «крупнейших отраслей, вносящих вклад в ВВП США», сравнивая производительность моделей с человеческой с помощью модель-судей.

Наконец, [GAIA2](#) вышел за рамки простого информационного поиска, используя имитацию мобильной среды для проверки способности ассистентов правильно

отвечать на запросы, опираясь на цепочки событий и вызовы инструментов. В настоящее время временно чувствительные и намеренно шумные подмножества (имитирующие сбои API) являются самыми сложными для моделей, тогда как поиск и выполнение задач кажутся крайне простыми для передовых моделей.

Научные ассистенты

[SciCode](#) (2024) проверяет, могут ли модели решать реальные научные задачи, создавая соответствующий научный код в различных STEM-областях (от биологии до математики, химии и др.). Задачи взяты из реальных рабочих процессов, и каждая основная проблема разбита на более простые подзадачи. В первой версии оценка проводилась учёными и модель-судьёй — модели показывали довольно низкие результаты на момент публикации (менее 5%), однако мне неизвестно, где можно найти актуальные данные.

[PaperBench](#) (2025) аналогично проверяет, могут ли модели воспроизводить исследования в области машинного обучения, но в более сложной постановке: при наличии высококачественных статей ICML модели должны реконструировать соответствующую кодовую базу (8 тысяч индивидуально оценённых задач, предоставленных авторами указанных статей, сгруппированы в деревья рубрик с учётом весов для итоговых оценок). Бенчмарк оценивается судьёй больших языковых моделей (хотя я предполагаю, что часть работы могла бы быть автоматизирована, если немного ограничить формат требуемого кода).

[DSBench](#) (2025) — мультимодальный бенчмарк для анализа данных на основе выборок Kaggle и ModelOff (финансовые данные). Согласно примерам в приложении, вопросы из ModelOff представлены в формате множественного выбора, что, вероятно, облегчает задачу, тогда как задачи Kaggle обладают собственными метриками.

[DABStep](#) (2025) оценивает модель на основе ранее закрытых (следовательно, не заражённых) рабочих нагрузок по анализу операционных данных с использованием реальных вопросов и данных. Все задачи требуют многоэтапного логического рассуждения и разнообразного разбора документов, а также, конечно, специализированных навыков работы с данными. Это удачная оценка, поскольку она сложная и воспроизводит действительно полезные реальные сценарии использования, а также потому, что для каждой задачи есть эталонные данные, что обеспечивает объективность оценки и делает её не слишком затратной.

Задачи ассистента проверяют интегрированные способности в реалистичных сценариях, но они либо динамичны и предназначены только для чтения, либо статичны в неизменяющейся среде. Для оценки адаптивности и динамического принятия решений необходимы среды, способные «удивить» модель.

Оценки на основе игр

Бенчмарки на основе игр интересны по нескольким причинам: они обычно оценивают адаптивность к меняющейся среде (в отличие от большинства задач для ассистентов, которые статичны), требуют длительного рассуждения в контексте и, наконец, последнее, но не менее важное — они **понятны** большинству людей. Однако они не основаны на реальных жизненных ситуациях и не всегда отражают хорошую производительность в действительно полезных сценариях использования.

Самой известной формальной оценкой среди них, вероятно, является [ARC-AGI](#). Первая версия (2019) состояла из последовательностей головоломок в виде таблиц, где модели нужно было определить последний элемент последовательности без предоставления явных правил.

Для меня этот бенчмарк очень напоминает логически ориентированные тесты на IQ и был решён в 2024 году.

Похожий бенчмарк (экстраполяция правил) — [Baba is AI](#) (2024). Последняя версия бенчмарка, ARC-AGI3 (2025, в разработке), всё ещё создаётся и содержит совершенно новые игры (требующие исследования, сложного планирования, управления памятью и др.), разработанные специально для этого бенчмарка. Разработка продолжается, и на данный момент лучшие решения для доступных задач сводятся к брутфорсу игр.

Сообщество и поставщики моделей исследовали ряд существующих игр с большими языковыми моделями. Однопользовательские приключенческие игры/RPG, такие как [TextQuests](#) (2025) или [Pokemon](#) (2024) (например, Twitch для [Claude](#) и [Gemini](#)), требуют сочетания долгосрочного планирования для достижения целей, что требует адекватного управления длительной контекстной памятью, способности к рассуждению и возврату назад. Те же навыки необходимы для однопользовательских survival-игр, таких как [Crafter](#) (2021, вдохновлённая Minecraft). Ряд однопользовательских игровых сред был интегрирован в тестовый набор [Balrog](#) (2024).

Конкурентные блефовые игры, такие как [Poker](#) (2025), разновидности Mafia — например, [Town of Salem](#) (2025) и [Werewolf](#) (2025, [здесь](#) / [там](#)), а также [Among Us](#) очень интересны для проверки логики, рассуждений и способности к обману. Например, Claude Opus 4 не может выиграть Town of Salem в роли вампира (обманчивая роль), но успешно справляется в роли крестьянина (необманчивая роль). Кооперативные игры, такие как [Hanabi](#), также могут использоваться для проверки адаптивности и способности к коммуникации в ограниченной среде.

Что особенно удобно в этих играх — наличие одного четкого и однозначного критерия результата: выиграла ли большая языковая модель игру или нет. В настоящее время, если бы я использовал их для оценки моделей, то, вероятно, обратил бы внимание на TextQuests для проверки способностей и Town of Salem для оценки безопасности.

Помимо проверки возможностей в контролируемых условиях, исследовали окончательную «неиграбельную» задачу: прогнозирование будущего.

Прогнозисты

В прошлом году появилась новая категория задач, к которым невозможно применить недобросовестные подходы: прогнозирование. (Технически, прогнозирование на фондовых рынках можно обмануть манипуляциями, но, надеюсь, мы ещё не достигли такого уровня финансовых стимулов, чтобы искажать оценки). Эти задачи требуют сочетания рассуждений по разным источникам, чтобы решать вопросы о событиях, которые ещё не произошли, но пока неясно, достаточно ли эти бенчмарки дискриминационны для значимой оценки, и, вероятно, они усиливают впечатление «успеха на игровом автомате» больших языковых моделей. (Выпадает ли производительность по некоторым событиям близко к случайной, потому что их невозможно предсказать или потому что модели с этим плохо справляются? С другой стороны, если модели правильно предсказывают событие, не слишком ли прост вопрос или слишком формализован?)

[FutureBench](#) тестирует, могут ли модели предсказывать будущие значимые новости. Он использует два источника: браузинг и большие языковые модели для генерации вопросов с недельным горизонтом, а также пользовательские прогнозы с букмекерских рынков. Все данные проходят тщательную фильтрацию и очистку перед использованием. Пока что модели едва ли превосходят случайный выбор

на вопросах, созданных человеком, и достигают успеха в 3/4 случаев на вопросах, сгенерированных моделью (вероятно, более лёгких).

[FutureX](#) аналогичен, но использует набор конкретных веб-сайтов (прогностические площадки, государственные сайты, сайты с общими рейтингами и платформы с данными в реальном времени), после чего применяет шаблоны для генерации вопросов о потенциальных будущих событиях. (когда STOCK достигнет POINT?). Ежедневно генерируется 500 вопросов с фильтрацией случайно нерелевантных вопросов.

Аналогичный подход применяется для генерации вопросов в [Arbitrage](#), при этом ключевое отличие – временной горизонт: события там должны разрешиться к 2028 году.

В том же духе существуют арены, где большим языковым моделям выделяются средства для активной торговли на финансовых рынках (например, Alpha Arena или Trading Agents) – такие эксперименты с меньшей вероятностью дадут значимые результаты, поскольку из-за своей стоимости обычно проводятся только один раз для каждой модели, что не обеспечивает статистической значимости.

Рекомендации

⌚ Кратко

Парадигма оценки эволюционировала вместе с ростом возможностей: от тестирования отдельных навыков до измерения интегрированной производительности в более реалистичных сценариях.

По состоянию на ноябрь 2025 года рекомендую использовать:

- **Основные возможности (для разработчиков моделей):** старые оценки возможностей для обучения, а также послеобучающие AIME26, когда он выйдет, GPQA, IFEval, SWE-Bench, длиннохвостые оценки на ваш выбор – HELMET, TauBench или BFCL, если вы нацелены на использование инструментов.
- **Основные возможности (для сравнения моделей на выводе):** IFBench, HLE, MathArena, AiderBench, LiveCodeBench, MCPUniverse.
- **Задачи с долгосрочной перспективой (для оценки реальной производительности):** GAIA2, DABStep, SciCode или отраслевые оценки, адаптированные к вашим сценариям использования.
- **Игры (для дополнительного развлечения и измерения устойчивости и адаптивности):** ARC-AGI3, когда выйдет, TextQuests, Town of Salem – если вас интересует безопасность, или любая другая игра, выходящая за рамки покера, шахмат или го.

Область движется в сторону оценок, проверяющих оркестровку возможностей, а не изолированные навыки для реального применения. Это соответствует нашей цели создавать модели, которые «хорошо работают» – системы, способные надёжно сочетать основные возможности, использовать инструменты и эффективно решать реальные задачи.

Я надеюсь, что область будет двигаться в сторону большего акцента на функциональном тестировании, а не на оценках моделей-судей, а также в сторону более понятных датасетов и задач.

Если вы хотите изучить ещё больше наборов данных, вы найдёте большой список старых интересных бенчмарков [тут](#) с моими заметками.

ПОНИМАНИЕ СОДЕРЖИМОГО

Неважно, как вы выбрали первоначальные наборы данных, самый важный этап — и всегда им останется — внимательно изучать данные: что у вас есть, что генерирует модель и её оценки. В конечном счёте, это единственный способ понять, насколько ваши оценки действительно релевантны для вашего конкретного случая использования.

Рекомендуется изучить следующее.

Процесс создания данных

- **Кто создавал конкретные примеры?** Идеально, если набор данных создан экспертами, затем идут оплачиваемые аннотаторы, далее — краудсорсинг, затем синтетические данные и MTurk. Также важно искать data card, где можно найти Демография аннотаторов — это важно для понимания языкового разнообразия датасета или возможного культурного смещения.
- **Все ли они были проверены другими аннотаторами или авторами?** Необходимо узнать, насколько высока межаннотационная согласованность на выборках (= согласны ли аннотаторы между собой?) и/или проверялся ли весь датасет авторами. Это особенно важно для датасетов, созданных с помощью низкооплачиваемых аннотаторов, которые обычно не являются носителями вашего целевого языка (например, AWS Mechanical Turk), иначе вы можете обнаружить опечатки, грамматические ошибки или бессмысленные ответы.
- **Были ли аннотаторам предоставлены чёткие инструкции по созданию данных?** Иначе говоря, является ли ваш датасет последовательным?

Проверка выборки

Возьмите 50 случайных примеров и вручную их проверьте; и я имею в виду сделать это самостоятельно, а не «попросить большую языковую модель найти необычные примеры в данных за вас».

В первую очередь следует проверить качество содержимого.

- Являются ли промпты чёткими и однозначными?
- Правильны ли ответы? (*Например: TriviaQA содержит несколько правильных ответов (поле aliases) на каждый вопрос, которые иногда могут противоречить друг другу.*)
- Отсутствует ли какая-либо информация? (*Например: в MMLU в ряде вопросов отсутствуют схемы.*)

Важно помнить, что наличие датасета в стандарте не гарантирует его качества — такое часто происходит, потому что большинство пользователей пропускают этот шаг.

Далее необходимо проверить релевантность к вашей задаче. Являются ли эти вопросы тем типом, на котором вы хотите оценивать большую языковую модель? Релевантны ли эти примеры вашему случаю использования?

Возможно, стоит также проверить согласованность выборок (особенно если планируете использовать few-shot обучение или вычислять агрегированную статистику): совпадает ли число вариантов ответа во всех образцах, если это оценка с несколькими вариантами? Соблюдается ли единообразие пробелов до и после промпта? Если оценка проводится в дополнительной среде, желательно использовать её, чтобы понять, какие вызовы происходят.

И наконец, стоит оперативно проверить количество образцов (чтобы убедиться в статистической значимости результатов — обычно минимум 100 образцов для автоматических бенчмарков).

В приведённом ниже просмотрщике, например, вы можете ознакомиться с первыми примерами известных бенчмарков после обучения, собранных Льюисом.

problem string · classes	answer string · classes	id string · classes
5 values	5 values	5 values
Let $f(x) = \frac{(x-18)(x-72)(x-98)(x-k)}{x}$. There exist exactly three positive real...	240	29
Find the sum of all positive integers n such that $n + 2$ divides the product $3(n + 3)(n^2 + ...)$	49	16
Let k be a real number such that the system $\begin{aligned} & 25 + 20i - z = 5 \\ & z - 4 - k \end{aligned}$...	77	7
Let S be the set of vertices of a regular 24-gon. Find the number of ways to draw 12...	113	25
Sixteen chairs are arranged in a row. Eight people each select a chair in which to sit so...	907	24

Задачи и метрики

Необходимо проверить, какие метрики используются: автоматические, функциональные или на основе модели-судьи? Ответ повлияет на стоимость проведения оценок для вас, а также на воспроизводимость и тип смещения. Лучшие (хотя и редкие) метрики — функциональные или основанные на верификаторах с использованием правил.

Итак, вы не можете воспроизвести указанные результаты модели?

Предположим, вы прочитали недавний технический отчёт о новой интересной модели и хотите воспроизвести их результаты на своём компьютере... но у вас не получается. Давайте разберёмся, почему.

При выполнении оценок качества кода будьте осторожны со слишком простыми юнит-тестами по принципу pass/fail! Современные LLM стали очень хорошо «читать», перезаписывая глобальные переменные, особенно в таких языках, как Python, где можно легко нарушить область видимости переменных.

Различная основа кода

Чтобы воспроизвести оценки модели с точностью до десятых, необходимо убедиться, что вы используете точно такую же основу кода, как в статье, которую хотите воспроизвести.

Обычно это значит либо использование кода оценки по умолчанию, предоставленного авторами, либо стандартной реализации в эталонной библиотеке, такой как Eleuther's AI lm_eval or HuggingFace's lighteval . Однако если исходный код для оценки не предоставлен, то, к сожалению, вряд ли вам удастся точно воспроизвести результаты.

Если вы хотите легко понять, какие расхождения возникают при использовании разных реализаций, вы можете ознакомиться с [этим блогом](#) (★), который мы написали совместно с командой eval из HuggingFace. В нем исследуются различия между тремя распространёнными реализациями оценки MMLU (в lm_eval, helm и в оригинальной реализации автора), а также то, как они влияют на оценки моделей.

Незначительные различия в реализации или загрузке

Мы заметили, что следующие моменты легко допустить ошибку, даже используя один и тот же код:

- **Разные случайные сиды.**
 - Обычно вывод менее чувствителен к случайным сидам, чем обучение. Тем не менее, они могут влиять на некоторые операции CUDA (см. страницу PyTorch по [воспроизводимости](#)) и менять предсказания, если вы используете стратегию генерации, отличную от жадной. Они также могут влиять на промпт, если вы используете few-shots, а также на некоторые функции предварительной или последующей обработки. → Незначительное изменение может привести к разнице в несколько пунктов.
- **Фактически, разные метрики.** На практике метрики могут различаться, даже если имеют одинаковое название. Вот несколько примеров:
 - Если исходная реализация — это логарифм правдоподобия [exact match], (вычисление логарифмов вероятностей различных возможных ответов), а вы используете генеративное [exact match] (сравнивая только основное жадное поколение с эталоном), вы не получите одинаковых результатов.
 - Также в кодах оценки мы встречали ряд задач, которые были определены как [exact match], но на самом деле представляли собой [prefix exact match] (сравнение только начала генерации с эталоном), или [suffix exact match] (противоположное), либо [quasi exact match] (точное совпадение с нормализацией). → Поэтому нельзя полагаться только на название метрики, чтобы понять, что происходит, и необходимо изучать исходный код.
- **Различная нормализация.**
 - Чтобы вернуться к нашему вышеуказанному [exact match] примеру сравнения в [lm_eval v1], ряд задач был просто обозначен как генеративные [exact match]: из этого можно было бы предположить, что предсказаниесравнивается именно так с Необходимо проверить, какие метрики используются: автоматические, функциональные или на основе модели-судьи? Ответ повлияет на стоимость проведения оценок для вас, а также на воспроизводимость и тип смещения. Лучшие (хотя и редкие) метрики — функциональные или основанные на верификаторах с использованием правил.

эталонным ответом. Если посмотреть на код, предсказание сначала проходит нормализацию (удаление пунктуации, унификация чисел и т.д.) перед сравнением с эталоном. Это очевидно значительно изменит результаты. (`[lm_eval v2]` теперь включает название нормализации в большинстве названий метрик.) → Это одна из самых распространённых ошибок, особенно в задачах, требующих значительной нормализации или постобработки ответов, например, при оценке математических задач (где необходимо извлечь ответ из сгенерированного объяснения).

Загрузка модели влияет на воспроизводимость

Четыре фактора, которые изменяют результаты даже при идентичном коде:

- **Аппаратное обеспечение:** PyTorch не гарантирует воспроизводимость на разных GPU и аппаратуре. Библиотека вывода: `transformers`, `vllm` и `sglang` по-разному обрабатывают батчи и матричные операции по состоянию на 2025 год
- **Размер батча:** Разные размеры батча приводят к разным результатам (для воспроизводимости размер батча следует фиксировать, однако следует быть осторожным с ошибками ОМ)
- **Точность загрузки:** Более низкая точность (особенно квантизированные модели по сравнению с моделями с плавающей точкой) изменяет численные результаты

Разный промпт

Три основных фактора могут влиять на вариации промпта.

Сам промпт

Формат, используемый для промпта, может значительно влиять на результаты.

Например, для ответов на вопросы с несколькими вариантами часто встречаются простые вариации (например, использование A vs A. vs A) для обозначения вариантов), которые, хотя и семантически эквивалентны (содержат точно один и тот же контент), могут привести к различию в несколько баллов для одной и той же модели.

Чувствительность к формату промпта

Мы провели эксперименты на эту тему (вы увидите разницу до 7 баллов для одной и той же модели на семантически эквивалентных промптах, 5 правых столбцов), а [в одной работе наблюдались схожие результаты](#).

Другой пример: модели Llama 3.1 правильно решали сложные задачи из MATH-Hard, но показали низкие результаты в рейтинге Open LLM Leaderboard, так как переобучились на формат промптов GSM8K и не смогли адаптироваться к новому формату для этой оценки, несмотря на наличие нескольких few-shot примеров.

Оценка на подмножествах MMLU, `acc_norm score (seed 0)`, 5-shot.

Mistral-7B-v0.1	49.0	50.5	52.1	54.5	56.4	55.4	55.5	57.0
Qwen1.5-7B	37.6	41.8	43.5	47.9	50.8	51.2	22.9	47.7
gemma-7b	44.6	48.0	47.6	53.5	54.2	54.9	56.4	50.7
phi-2	39.1	44.3	46.5	46.1	47.1	48.4	51.7	45.8
DeciLM-7B	43.6	48.9	49.5	51.0	51.3	52.0	52.8	52.3
	A	Б	В	Г	Е	Д	Е	*

Форматы промптов:

- A. ...? -> выбор1/выбор2/...
- Б. В:...? А: -> выбор1/выбор2/...
- C. Вопрос: ...? Ответ: -> выбор1/выбор2/...
- D. Вопрос: ...? Варианты: ... Ответ: -> выбор1/выбор2/...
- E. Вопрос: ...? Варианты: А. ... Ответ: -> выбор1/выбор2/...
- F. Вопрос: ...? Варианты: (A) ... Ответ: -> выбор1/выбор2/...
- G. Вопрос: ...? Варианты: А. ... Ответ: -> A/B/C/D
- H. Вопрос: ...? Варианты: (A) Ответ: -> (A)/(B)/(C)/(D)

Эта отличная работа * также подчёркивает побочный эффект: многие модели сейчас обучаются переобучаться на промпты и форматы ответов из бенчмарков, что ухудшает их способность адаптироваться к другим промптам во время оценки.

Некоторые задачи также предваряются служебным промптом (например: Следующие вопросы касаются <topic>) — это Присутствие или отсутствие также влияет на оценки.

Системный промпт и шаблон чата

Модели диалога обычно проходят обучение с инструкциями/предпочтениями или дообучение. На этом этапе они учатся следовать определённым шаблонам при выводе. Например, шаблоны могут требовать начинать раунды диалога с общего промпта (называемого **[system prompt]**), предваряемого определёнными токенами (обычно **[System]**):. Этот промпт предназначен для предоставления модели высокоуровневых инструкций, таких как содержание персоны или общие указания по стилю ответа. Раунды диалога также могут подразумевать добавление в текст ключевых слов- префиксах, таких как **[User]** для запросов и **[Assistant]** для ответов.

При использовании **few shot** необходимо также выбрать, хотите ли вы, чтобы примеры предоставлялись многоходово (имитируя ходы пользователя/ассистента) или все сразу (в одном пользовательском промпте).

Несоблюдение ожидаемого моделью шаблона чата на этапе вывода значительно ухудшит её производительность, так как выведет результат за пределы пространства вероятностей, к которому модель стремится.

Аналогично, если вы используете модель для рассуждений, необходимо убедиться, сравниваете ли вы работу с включённым или выключенным режимом мышления.

Несколько примеров

Два аспекта легко испортить при использовании нескольких few-shot примеров: количество примеров, их выбор и конкретный порядок.

Это также тот момент, где важно обращать внимание на случайные сиды.

Параметры

Для генеративных оценок следует обратить внимание на следующие параметры: 1) использовать **один и тот же токен конца предложения** (вероятно, не стоит использовать токен по умолчанию для чат- и рассуждающих моделей); 2) позволять модели **генерировать одинаковое количество токенов** для оценки (это особенно важно для рассуждающих моделей, которым требуется большое количество токенов в режиме мышления); 3) при использовании сэмплирования — использовать **одинаковые сиды и параметры температуры**.

Важность использования одних и тех же примеров не слишком удивительна, если предположить, что некоторые примеры лучше объясняют задачу, чем другие. Более удивительно другое: необходимо не только использовать абсолютно те же примеры, но и предъявлять их в точно таком же порядке. Изменение порядка тех же самых примеров привело к тому, что на некоторых поднаборах MMLU мы наблюдали разницу до 3 баллов ([некоторые результаты можно увидеть здесь](#) — это третья цветовая диаграмма).

АВТОМАТИЧЕСКИЙ ПОДБОР ЭФФЕКТИВНЫХ БЕНЧМАРКОВ ДЛЯ ОБУЧЕНИЯ МОДЕЛЕЙ

В некоторых случаях необходимо не просто апостериорно воспроизвести существующие результаты, а действительно понять, насколько хорошо ваша модель обучается в процессе. Оценки, необходимые в таких ситуациях, обладают иными свойствами по сравнению с оценками конечной производительности моделей, поскольку нужны задачи, обеспечивающие надежный сигнал даже когда модель ещё недостаточно совершенна.

Команда FineWeb разработала метод выбора лучших оценок для проведения аблаций на этапе предподготовки по 9 языкам — прислушаемся к их мудрым советам.

Для этих языков мы собрали и реализовали все доступные задачи, которые смогли найти — **всего 185**. Затем мы приступили к выбору задач с двумя основными целями: обеспечить **разнообразие оценок** и гарантировать, что каждая задача даёт **надежный сигнал** во время предподготовки.

Для обеспечения разнообразия оценок мы стремились охватить широкий спектр возможностей модели, включая:

- **Понимание прочитанного (RC):** понимание представленного контекста и ответы на вопросы на его основе.
- **Общие знания (GK):** ответы на вопросы о фактах из различных областей без дополнительного контекста.
- **Понимание естественного языка (NLU):** осмысление семантики представленного ввода.
- **Здравый смысл (RES):** демонстрация способности выполнять простые рассуждения, требующие инкорпорированных знаний.
- **Генеративные задачи:** способность генерировать текст на целевом языке без «подсказок» в виде вариантов с выбором.

Мы считаем, что задачи обеспечивают надежный сигнал, если они дают достоверную оценку. Это означает, что оценка должна быть выше случайной базовой линии, увеличиваться по мере прогресса обучения, показывать низкую вариативность при разных сидах и обеспечивать устойчивый рейтинг моделей на каждом этапе обучения для моделей аналогичного размера, обученных с одинаковыми гиперпараметрами на одинаковом объеме данных.

Для тщательной проверки сигнала, который дают наши задачи, мы обучили множество моделей с 1,5 млрд параметров для каждого языка, используя 30 млрд токенов из подмножеств поддерживаемых языков пяти крупнейших публичных многоязычных веб-корпусов. Два аспекта легко испортить при использовании нескольких few-shot примеров: количество примеров, их выбор и конкретный порядок. Наборы данных. Эти модели были обучены с одинаковыми гиперпараметрами и токенизатором. Затем мы оценивали их на регулярных контрольных точках по собранным задачам (без инструкций и системного промпта в 0-shot режиме).

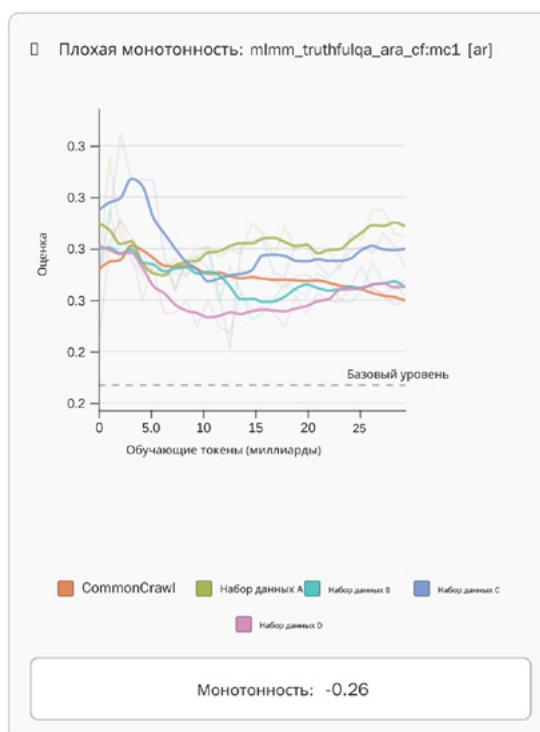
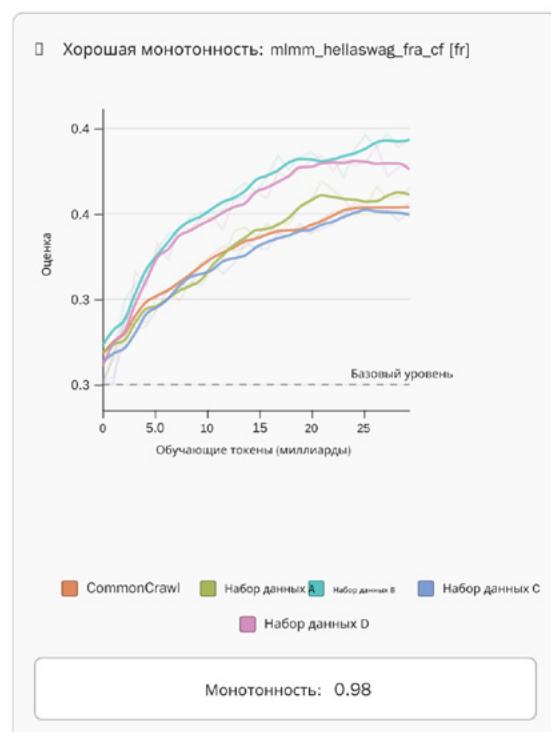
Этот процесс потребовал многократных запусков оценки для каждой задачи из-за итераций по её реализации, что в итоге потребовало **73 000 GPU-часов** !

После обучения **49 моделей** мы наконец смогли определить, что для нас означает **надёжный сигнал**!

Монотонность

Одним из наших ключевых требований к задаче является то, что её можно изучить на данных обучения и что это **обучение можно наблюдать постепенно по мере продвижения процесса обучения**. Без такого улучшения со временем остаётся неизвестным, будет ли вообще улучшение в будущем.

Для измерения этого мы использовали **ранговую корреляцию Спирмена**, чтобы количественно оценить связь между шагами обучения и результатом. Ранговая корреляция Спирмена позволяет зафиксировать монотонность даже тогда, когда оценки не изменяются линейно с числом шагов. Мы требовали, чтобы каждая задача имела среднюю корреляцию не менее 0.5 во всех запусках обучения модели.



Низкий уровень шума

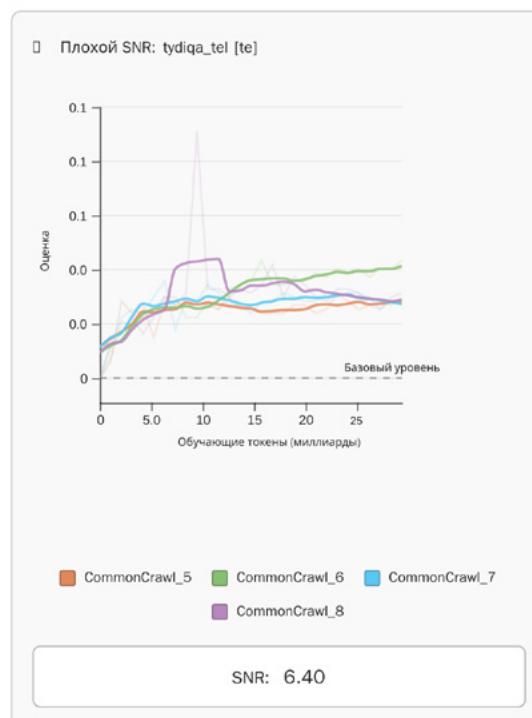
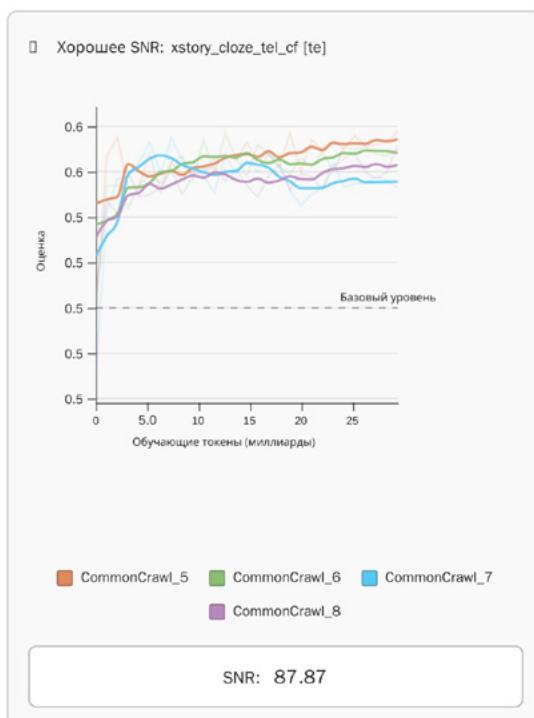
При сравнении качества моделей по задачам необходимо учитывать, вызваны ли различия **шумом оценки или истинными вариациями производительности**.

Шум может возникать из-за стохастических процессов, связанных с обучением модели, таких как случайное сэмплирование токенов, перемешивание данных или инициализация модели ([Madaan et al.](#), 2024). Чтобы оценить чувствительность каждой задачи к этому шуму, мы обучили четыре дополнительные модели на наших собственных одноязычных корпусах (неотфильтрованные данные CommonCrawl для каждого языка) с разными сидами.

Для каждой задачи мы вычислили:

1. Во-первых, стандартное отклонение значений оценки модели на каждом шаге (примерно каждые 1 млрд токенов), которое мы называем **per-step-std**.
2. Затем, чтобы получить глобальную меру изменчивости, мы усреднили все per-step-std, получив среднее и стандартное отклонение (**avg-std**) за весь период обучения. Мы предполагаем, что это значение является верхней границей для всех архитектур моделей и наборов обучающих данных (так как оно аппроксимировано моделями, обученными на «грязном» наборе данных, следовательно с большей изменчивостью).
3. Наконец, мы рассчитали **отношение сигнал/шум (SNR)** как основной показатель изменчивости задачи. Мы вычисляем SNR как среднее значение оценки при 30 млрд токенов по всем запускам, делённое на avg-std. Эта метрика измеряет значимость общей оценки относительно вариаций оценки (шума).

Мы стремились к тому, чтобы для каждой задачи отношение сигнал/шум (SNR) превышало 20. Единственным исключением являются генеративные задачи, которые обычно имеют сравнительно низкое SNR, но их всё же стоит включать, поскольку они дают важные сведения о поведении модели при генерации без ограничений (без вариантов ответов). В многоязычной среде это особенно актуально, так как некоторые модели, обученные на нескольких языках, могут показывать высокие результаты по задачам, но внезапно отвечать на неправильном языке в генеративных задачах!

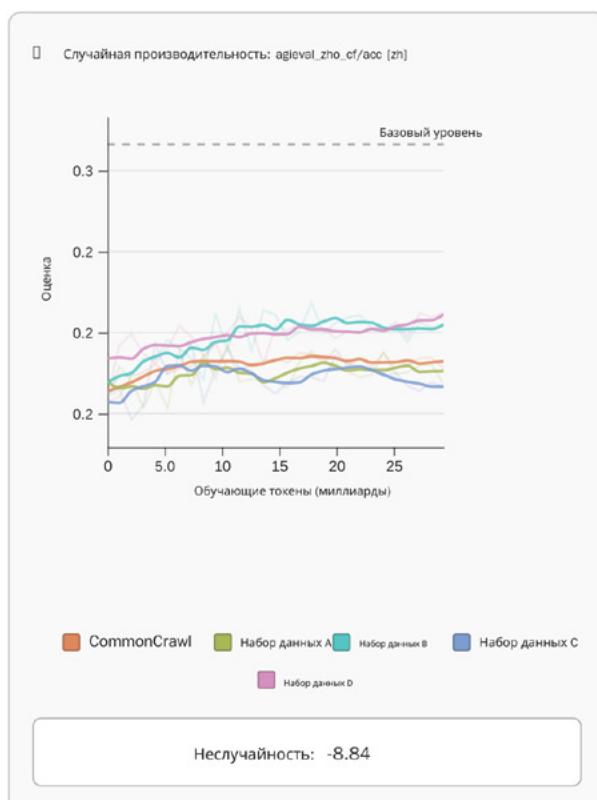
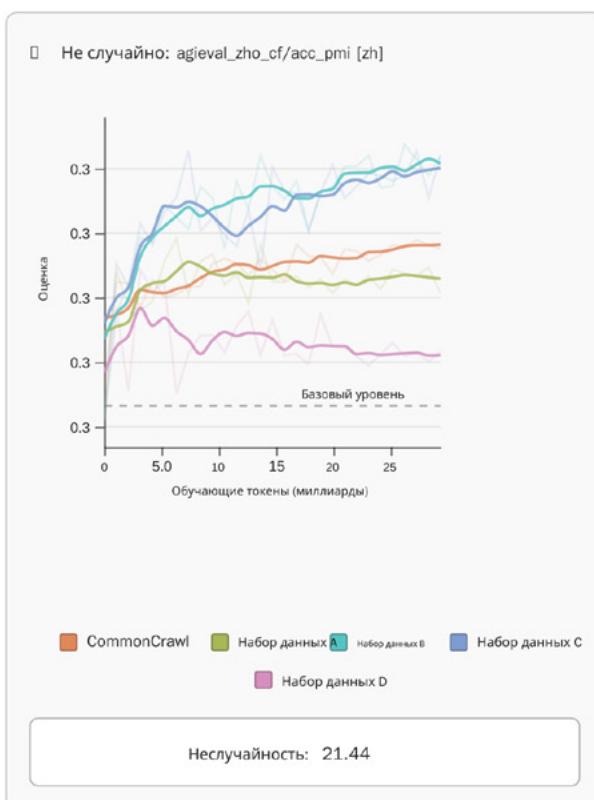


При предположении нормального распределения производительности модели по разным сидам мы хотим, чтобы производительность при запуске бенчмарка была как минимум на 3 конечных стандартных отклонения выше базового случая случайного бенчмарка. Это означает, что 99,85% оценок по сидам превышают базовый случай случайного бенчмарка (формально: $\text{benchmark-run performance} - \text{benchmark random baseline} > 3 * \text{final-std}$).

Неслучайная производительность

Многие способности моделей формируются на более поздних этапах обучения, поэтому **многие задачи** (особенно сложные, например, математические) **показывают производительность на уровне базовой линии в течение длительного времени**. Хотя эти задачи полезны, они не предназначены для оценки на ранних этапах предварительного обучения, и **мы не хотели оставлять их** для этого этапа.

Сначала мы вычислили базовую случайную производительность задачи (как сумму $1/n_{\text{выборов}}$ по всем примерам с множественным выбором и как ноль для генеративных оценок). Затем мы определили расстояние задачи от базовой линии как разницу между максимальным результатом среди всех моделей и базовой линией.



Согласованность порядка моделей

Не забывайте, что основная цель этих оценок — сравнивать модели и наборы данных!

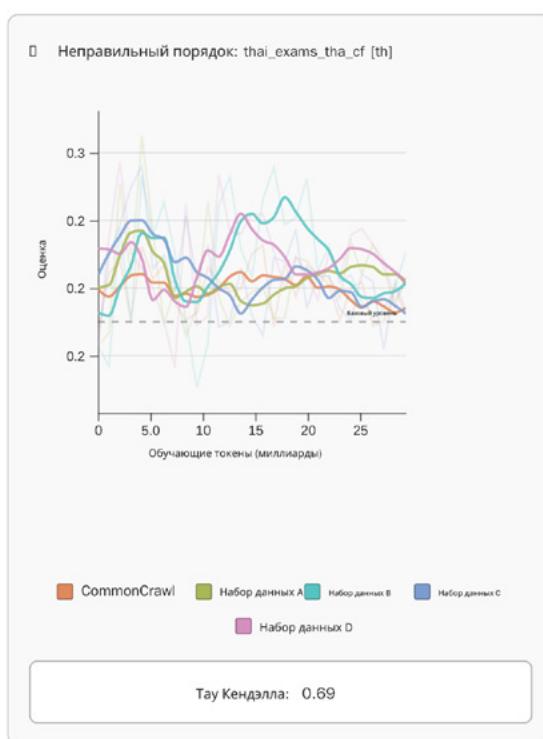
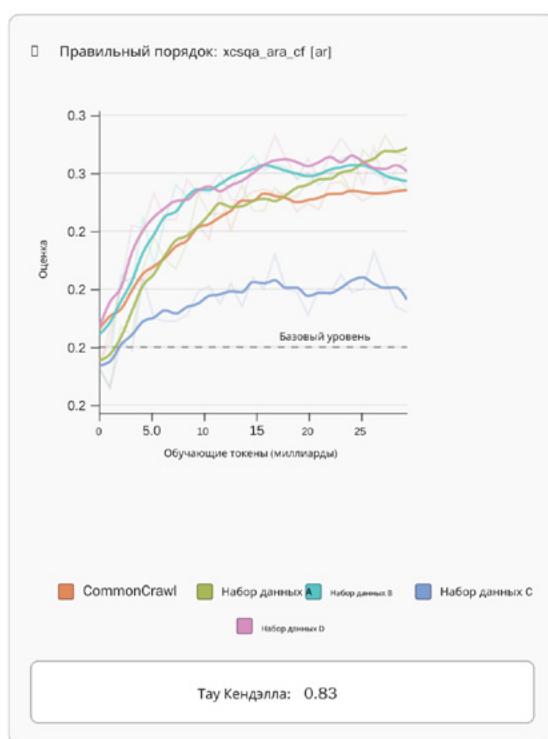
В дальнейшем мы планируем использовать эти оценки для выбора лучших наборов данных для полного предварительного обучения моделей. **Это означает, что наши задачи должны ранжировать датасеты, обученные на очень малом количестве токенов (мы обычно проводим аблации данных на 30 млрд токенов), в том же порядке, в котором они ранжировались бы при более длительном обучении, после значительно большего числа шагов.**

Другими словами, мы хотели бы, чтобы задачи обладали **прогностической способностью в отношении будущей производительности во время предобучения**: если датасет предобучения А превосходит датасет предобучения В при 30 миллиардах токенов, нам бы хотелось, чтобы эта тенденция сохранялась и при 300 миллиардах токенов.

Доказать это по существу невозможно, но существует необходимое предварительное условие, которое мы можем проверить: чтобы результаты были согласованы на больших масштабах, они должны сначала демонстрировать согласованность на меньших масштабах!

Чтобы измерить эту согласованность в порядке задач, мы вычислили среднее значение коэффициента **Кендалла Тао** для ранжирования моделей между каждыми двумя последовательными шагами. Мы учитывали только шаги, начиная с момента обучения после 15 млрд токенов, поскольку ранжирование до этого диапазона оказалось чрезвычайно шумным. Высокое значение этого показателя указывает на то, что порядок задач сохраняет согласованность по мере продвижения обучения.

У нас не было строгого минимального требования к этому свойству, вместо этого мы использовали его для сравнений между задачами.



Метрики

Поскольку цели в CF для задач с выбором из нескольких вариантов — это сами варианты, каждая цель может иметь различное количество токенов, символов и безусловную вероятность (вероятность сгенерировать выбор без контекстного префикса).

Измерение точности без нормализации заставило бы модели отдавать предпочтение ответам с меньшим количеством токенов, например.

Чтобы учесть это, мы рассматриваем следующие варианты точности:

- **Точность: [acc]** = $\arg \max_i (\ln(P(a_i|q)))$
- **Точность, нормализованная по длине в символах: [acc_char]** = $\arg \max_i \frac{\ln(P(a_i|q))}{\text{число_символов}(a_i)}$
- **Точность, нормализованная по длине в токенах: [acc_token]** = $\arg \max_i \frac{\ln(P(a_i|q))}{\text{число_токенов}(a_i)}$
- **Точность PMI: [acc_pmi]** = $\arg \max_i \ln \frac{P(a_i|q)}{P(a_i|u)}$.

Где a_i — выбор ответа i , q — промпт вопроса, а $P(a_i | q)$ — вероятность появления следующим. Годобнее см. [Gu et al., 2024](#) и [Biderman et al., 2024](#).

[acc_pmi] метрика показывает, насколько модель с большей вероятностью предсказывает A_i при наличии контекста вопроса по сравнению с отсутствием контекста вообще. Это может быть полезно, если правильный ответ содержит обычно маловероятные токены, из-за чего модель менее склонна выбирать такой ответ.

Для наших генеративных задач, напротив, мы использовали следующие метрики:

- **[prefix_match]** : точное совпадение, при котором должен совпадать только префикс ответа
- **[f1]**: F1-метрика, вычисляемая по предсказанным и эталонным словам, выделенным с помощью токенизатора слов Для обеих генеративных метрик применяется минимальная предварительная обработка: удаляются артикли и пунктуация , а текст приводится к нижнему регистру.

Выбор лучших метрик оказался **непростой задачей**. Не существует единой метрики, которая последовательно превосходила бы все остальные; мы часто сталкивались с тем, что одна метрика демонстрировала лучшую монотонность, а другая — более высокий сигнал-шум. В подобных случаях мы обычно принимали решение, исходя из метрики, выбранной для реализации задачи на другом языке. Мы осознаём, что такой ручной выбор зачастую невозможен, и поэтому предлагаем следующие рекомендации:

→ Задачи с множественным выбором

- Мы обнаружили, что **базовая точность** хорошо работает для задач, где варианты ответов отличаются незначительно (например, Да/Нет/Также), особенно для задач NLI. В таких случаях, когда варианты ответов часто представлены одним токеном, рекомендуется использовать базовую точность.

- В то время как авторы OLMES ([Gu et al., 2024](#)) рекомендуют использовать PMI для задач с необычными словами, мы установили, что **PMI** особенно эффективен для «сложных» задач рассуждения и знаний, таких как AGIEVAL или MMLU. В этих случаях PMI давал лучшие результаты и часто был единственной метрикой, обеспечивающей производительность выше случайной. Тем не менее, PMI в среднем был слабейшей метрикой по всем остальным задачам и при этом вычислялся вдвое дольше. Поэтому мы рекомендуем применять его только для сложных задач рассуждения и знаний.
- Метрики, которые мы признали **наиболее надёжными в целом**, — это метрики нормализации длины (на основе токенов или символов). Однако лучший выбор зависел от языка и не был постоянным для конкретной задачи. В связи с этим мы рекомендуем использовать максимум из точность_символа и точность_токена для получения наиболее надежных результатов. Обратите внимание, что точность_токена сильно зависит от токенизатора. Во всех наших аблациях все модели обучались с использованием одного и того же токенизатора.

→ Генеративные задачи

Для **генеративных метрик** выбор очевиден: мы рекомендуем использовать F1-метрику, если не требуется точное совпадение, как в задачах, связанных с математикой. F1 обычно менее шумная и более устойчивая к небольшим изменениям в сгенерированных ответах.

Создание собственной оценки

На этом этапе у вас, вероятно, уже есть чёткое представление о причинах проведения оценки, о существующих бенчмарках и их релевантности для разных стадий модели (обучения, вывода базовых и дообученных моделей), но что если для вашего конкретного случая ничего не существует?

Именно в такой ситуации может понадобиться создание собственной оценки.

НАБОР ДАННЫХ

Использование существующих данных

Вы можете использовать существующие наборы данных как есть и изменить промпты или связанные метрики (как это делалось в старых оценках для адаптации их к новым методам промптинга), но также можно агрегировать наборы данных.

Агрегация датасетов — это хороший подход, когда необходимо оценить конкретную способность, недостаточно покрытую одним бенчмарком. Вместо того чтобы начинать с нуля, можно комбинировать примеры из нескольких существующих датасетов для создания целевого набора оценки. Именно так, например, поступили авторы статьи «Measuring AGI», пытаясь создать новый датасет «AGI evaluation».

При агрегации датасетов обращайте внимание на следующие моменты:

- они содержат избыточные данные (большинство математических датасетов — это переписывания или агрегации одних и тех же исходных задач)
- требуется сбалансированное представительство разных источников (желательно, чтобы один датасет не доминировал и неискажал результаты оценки) — это также определит, стоит ли агрегировать результаты по всем примерам или по отдельным поднаборам.
- форматы и уровни сложности совместимы (как правило, при создании единого набора данных будьте внимательны, чтобы не смешивать примеры, требующие сэмплирования, с теми, которые этого не требуют).

Примеры: MMLU, Big-Bench (сотни разнообразных задач) и HELM (объединяет несколько существующих бенчмарков для целостной оценки).

Новое исследование EpochAI (2025) демонстрирует, [как эффективно агрегировать бенчмарки в единую структуру](#), чтобы повысить сложность объединённого датасета и снизить риск насыщения

Использование человеческих аннотаторов

Рекомендую ознакомиться с разделом 3 этого обзора лучших практик по обеспечению качества аннотаций данных. Если вам требуется качество производственного уровня и вы располагаете возможностями для применения всех этих методов — смело приступайте!

Процесс аннотирования

- Итеративное аннотирование
- Тщательный отбор данных
- Схема аннотирования
- Разработка рекомендаций
- Пилотное исследование
- Этап валидации



Рисунок 1: Рекомендуемые лучшие практики аннотирования данных из комплексного обзора на aclanthology.org/2024.cl-3.1 — ознакомьтесь с разделом 3.

Тем не менее, следующие важные рекомендации актуальны независимо от размера вашего проекта после того, как вы определите задачу и критерии оценки.

- Отбор рабочей силы и, при возможности, материальное стимулирование.**

Скорее всего, вы захотите, чтобы люди, выполняющие вашу задачу, делали следующее:

1. соблюдение определённых демографических характеристик. Некоторые примеры: быть носителями целевого языка, иметь высшее образование, быть экспертами в конкретной области, представлять географическое разнообразие и т. п. Ваши требования будут зависеть от поставленной задачи.

2. обеспечивать высокое качество работы. Сегодня особенно важно добавить возможность проверки ответов на предмет того, были ли они сгенерированы большими языковыми моделями, а также отфильтровать часть аннотаторов из вашей выборки. *На мой взгляд, если вы не рассчитываете на высоко мотивированных краудсорсинговых аннотаторов, всегда лучше оплачивать их труд справедливо.*

- Проектирование руководства.** Обязательно уделите много времени тщательному обдумыванию ваших рекомендаций! Это один из аспектов, на котором мы потратили больше всего времени при работе с набором данных [GAIA](#).

- Итеративная аннотация.** Будьте готовы провести несколько раундов разметки, поскольку ваши аннотаторы могут неправильно понять инструкции (они более неоднозначны, чем кажется)! Многократное создание примеров позволит вашим аннотаторам действительно прийти к единому пониманию того, что требуется.

- Оценка качества и ручная кураторская работа.** Вам необходимо контролировать ответы (особенно через согласие между аннотаторами, если это воз-

Если у вас нет высокомотивированных краудсорс-аннотаторов, всегда платите справедливо. Аннотаторы с низкой оплатой выполняют работу менее качественно, допускают больше ошибок и могут использовать LLM, чтобы быстро завершать задания.

При создании бенчмарка [GAIA](#) разработка руководящих инструкций заняла больше времени, чем любой другой этап. Чёткие, недвусмысленные инструкции стоят вложений — они предотвращают дорогостоящие повторные циклы аннотирования.

можно) и провести финальный отбор, оставив только самые качественные и релевантные ответы.

Специализированные инструменты для создания аннотированных высококачественных наборов данных, такие как [Argilla](#), тоже могут оказаться полезными.

Дальнейшие шаги

- ★ [Как создать собственную платформу аннотаторов за пару минут](#), автор Мориц Лаурер. Полезное чтение для получения практического опыта с использованием открытых инструментов (например, Argilla и Hugging Face), а также для лучшего понимания особенностей масштабной человеческой аннотации.
- ★ [Руководство по хорошим практикам аннотации](#). Обзор всех статей о человеческой аннотации, опубликованных в 2023 году, очень полный. Немного плотный, но вполне понятный.
- [Ещё одно руководство по хорошим практикам аннотации](#) от ScaleAI, специализирующейся на человеческих оценках. Это более лёгкое дополнение к вышеупомянутому документу.
- [Assumptions and Challenges of Capturing Human Labels](#) — статья о том, как выявлять причины разногласий аннотаторов и эффективно их устранять на практике.

Практические советы и рекомендации

Ниже приведено несколько практических советов, которые стоит учитывать при работе с человеческими аннотаторами для создания датасета оценки.

Проектирование задачи

Простота — лучше: Задачи аннотирования могут становиться неоправданно сложными, поэтому держите их максимально простыми. Минимизация когнитивной нагрузки аннотаторов поможет им сохранять фокус и обеспечивать более высокое качество аннотаций.

Проверьте, что вы показываете: Показывайте только ту информацию, которая необходима для выполнения задания, и убедитесь, что вы не включаете ничего, что может привнести дополнительный байас.

Учитывайте время аннотаторов: То, где и как отображается информация, может создать дополнительную работу или увеличить когнитивную нагрузку, что ухудшит качество результатов. Например, убедитесь, что тексты и само задание видны одновременно, избегайте лишней прокрутки. Если вы комбинируете несколько задач и результат одной влияет на другую, можно показывать их последовательно. Продумайте, как всё отображается в вашей аннотационной среде, и посмотрите, можете ли вы упростить интерфейс ещё сильнее.

Протестируйте настройку: Когда вы спроектировали задачу и подготовили инструкции, протестируйте её сами на нескольких примерах, прежде чем подключать всю команду, и при необходимости доработайте.

Во время аннотирования

Аннотаторы должны работать независимо: Лучше, если аннотаторы не будут помогать друг другу и не будут видеть работу других. Иначе они могут передавать друг другу свои байасы и вызывать дрейф аннотаций. Согласование работы должно происходить через чёткие, исчерпывающие инструкции. Новых членов команды желательно сначала обучать на отдельном датасете и/или проверять согласованность через метрики межаннотаторского согласия.

Последовательность – ключевой фактор: Если вы вносите важные изменения в инструкции (например, меняете определение, формулировку задания или добавляете/удаляете метки), подумайте, нужно ли пересмотреть уже аннотированные данные. Как минимум, отслеживайте изменения инструкций в метаданных датасета, например с помощью поля вроде [guidelines-v1].

Гибридное аннотирование: человек + модель

Иногда команды ограничены во времени и ресурсах, но не хотят отказываться от преимуществ человеческой оценки. В таких случаях можно использовать модели, чтобы упростить процесс.

Аннотирование с поддержкой модели: Можно использовать предсказания или генерации модели как предварительные аннотации, чтобы команде не приходилось начинать с нуля. Однако помните, что это может привнести байасы модели, а если её точность низкая, объём работы для аннотаторов может даже увеличиться.

Контроль модели-судьи: Можно комбинировать подход «модель как судья» (см. соответствующий раздел) с человеческими супервизорами, которые подтверждают или отклоняют выводы модели. Однако учтите, что те же байасы, обсуждаемые в разделе «Плюсы и минусы человеческой оценки», проявятся и здесь.

Выявление крайних случаев: Для ещё более ускоренной работы можно использовать «жюри» моделей, а человеческие супервизоры будут подключаться только там, где модели не согласны друг с другом или возникает ничья. Снова учитывайте байасы, обсуждавшиеся в разделе «Плюсы и минусы человеческой оценки».

Создание датасета искусственно

Использование правил-основных методов

Если ваша задача позволяет, использование процедурно сгенерированных бенчмарков – отличный способ получить практически бесконечный запас примеров и избежать загрязнения данных! Они могут алгоритмически генерировать неограниченное количество новых тестовых кейсов, контролируя сложность и обеспечивая автоматическую проверку, что гарантирует, что модели не сталкивались с этими примерами во время обучения.

Для некоторых примеров вы можете ознакомиться с [NPHardEval](#), [DyVal](#), [MuSR](#), [BabiQA](#), [ZebraLogic](#), IFEval или GSMTemplate среди прочих. **NPHardEval** генерирует задачи, основанные на сложности, такие как задачи на графах с автоматической проверкой и ежемесячным обновлением для снижения риска переобучения. **MuSR** создаёт сложные задачи на рассуждение, например, детективные истории объемом в 1000 слов с использованием нейросимволической генерации. **ZebraLogic** алгоритмически создаёт логические головоломки, генерируя решения и итеративное уменьшение подсказок с использованием SAT-солверов. **BabiQA** моделирует сущности, последовательно выполняющие действия. **IFEval** проверяет следование инструкциям с помощью более 500 промптов, содержащих проверяемые ограничения, такие как количество слов, которые можно проверить

программно. **GSM-Symbolic** применяет шаблоны для генерации разнообразных математических задач.

Задачи, которые обычно соответствуют этой парадигме, проверяют математические, логические или программные навыки.

Создание синтетических данных с помощью моделей

Если вы хотите создавать синтетические данные, обычно начинают с набора исходных документов, служащих эталонными данными. Это могут быть внутренние документы, специфичные для вашего случая, либо общедоступные высококачественные источники (например, Wikipedia, Stack Overflow и др.). Далее, скорее всего, потребуется разбить данные на единицы с автономным смыслом.

После этого обычно требуется, чтобы модель генерировала вопросы на основе ваших данных. Для этого необходимо выбрать передовую модель и разработать качественный промпт, предлагающий модели сгенерировать вопросы, релевантные кейсу использования, на основе предоставленных данных. Рекомендуется просить модель указать источник, на котором она основывалась при формулировке вопроса.

Вы также можете использовать исходные промпты в качестве примеров для внешней модели, которая затем создаст промпт для вашей модели с целью генерации новых вопросов, если вы хотите полностью перейти на синтетический подход ^^

После этого можно провести автоматическую валидацию, используя модель из другой семейства в роли модели-судьи для ваших эталонных данных, вопросов и ответов.

⚠️ Всегда обязательно проверяйте свои данные

Несмотря на искушение автоматизировать всё, на каждом этапе необходимо контролировать данные, чтобы обеспечить качество оценок. Оценка — это ключевой процесс, и для него нужны исключительно качественные данные.

Управление контаминацией

В целом следует считать, что любой публично доступный в интернете датасет уже содержит или рано или поздно будет содержать загрязнения.

Решения для смягчения этой проблемы включают:

- предоставление «канарейки» в наборе для оценки (например, в [BigBench](#)): это специфическая комбинация символов, которую создатели моделей могут искать в своих наборах для обучения, что будет указывать на наличие оценки
- предоставление наборов для оценки в [зашифрованном](#) или [ограниченном](#) виде, чтобы их было трудно распарсить веб-краулерами — таким образом они случайно не попадут в наборы для обучения
- запуск [динамических бенчмарков](#): бенчмарки, регулярно обновляемые со временем, чтобы модели не могли «заучивать ответы наизусть» (хотя это увеличивает стоимость наборов данных)
- если вы запускаете бенчмарк, попытка [обнаружить загрязнение](#) постфактум (например, путем анализа perplexity генерации или создания адвокатских вариантов промптов — однако ни один метод не является безошибочной системой обнаружения загрязнения)

Тем не менее, загрязнение набора данных не означает, что он будет неинтересным или неинформативным во время обучения, как показано в разделе об аблациях.

Модель, способная хорошо предсказывать только на данных для обучения (и не усвоившая более высокоуровневые обобщённые закономерности), считается **переобученной**. В менее экстремальных случаях всё равно важно проверить, способна ли ваша модель обобщать структуры данных, отсутствующие в распределении обучающего набора (например, классифицировать токсичность на Stack Overflow после обучения на токсичности, обнаруженной только на Reddit).

ВЫБОР ПРОМПТА

Промпт определяет, сколько информации о задаче передаётся модели и каким образом она подаётся. Обычно промпт состоит из следующих частей: необязательный **вводный промпт**, описывающий задачу и формат ожидаемого вывода; при необходимости — **добавленный контекст** (например, источник, изображение); **основной промпт** с запросом к модели; а также optionalные варианты для оценок с множественным выбором.

При задании промпта важно учитывать, что даже незначительные изменения в семантически эквивалентных формулировках могут существенно влиять на результаты, а также что разные форматы промптов могут выгодно или неблагоприятно воздействовать на конкретные модели (см. этот раздел).

→ Это можно компенсировать повторным проведением оценки с вариациями промптов (хотя это может быть дорого), либо однократным запуском оценки с использованием разных форматов промптов, распределённых между примерами схожей сложности.

→ Вы также можете предоставить вашей модели примеры, чтобы помочь ей следовать ожидаемому формату (с помощью few-shot примеров), и добавление связующих слов в целом этому способствует.

ВЫБОР МЕТОДА ВЫВОДА ДЛЯ ВАШЕЙ МОДЕЛИ

Вам потребуется выбрать подходящий метод вывода.

Напоминание об оценках на основе логарифмов вероятностей

Использование логарифмов вероятностей подходит для ответов на вопросы с несколькими вариантами (MCQA), проверки знаний модели или способности разрешать неоднозначности.

- *Преимущества:*
 - Гарантирует, что все модели имеют доступ к правильному ответу
 - Служит приближением для оценки «уверенности» модели (и её калибровки)
 - Быстрая оценка, особенно когда модель запрашивается предсказать только один токен (A/B/C/D — индексы вариантов или Да/Нет и т. п.)
 - Позволяет получать сигнал о качестве выполнения задач малыми моделями

- *Недостатки:*

- Немного завышает оценки маленьких моделей, которые при свободной генерации могли бы выдавать ответы вне доступных вариантов.
- Некоторые модели [предпочитают определённые варианты ответов в зависимости от порядка их представления](#), что может приводить к нерепрезентативным оценкам (если вы не повторяете оценку несколько раз, перемешивая порядок образцов, что обязательно стоит делать для статистической значимости, если позволяет бюджет!)

Совет: простой способ ускорить оценки MCQA

Вы можете значительно ускорить предсказания MCQA, если убедитесь, что вашей модели нужно предсказывать только один токен для задачи.

Таким образом, вместо запуска `number_of_choices` предсказаний (контекст + выбор 1, контекст + выбор 2 и так далее), вы можете просто выполнить вывод для контекста и вычислить вероятностное распределение по всему словарю (включающему все ваши варианты из одного токена), чтобы получить логарифмы вероятностей интересующих вас вариантов, и сделать это за один проход.

Напоминание о генеративных оценках

Сегодня большинство оценок – генеративные: использование генераций отлично подходит для задач, где важно проверить связность, рассуждения или способность модели действительно отвечать на вопросы. Это также самый релевантный способ оценки моделей рассуждений.

- *Преимущества:*

- Должно действительно коррелировать со способностью больших языковых моделей генерировать связный текст, и чаще всего это именно то, что интересует пользователей.
- Единственный способ оценить как закрытые, так и открытые модели

- *Недостатки:*

- Может быть сложнее для оценки (см. ниже)
- Дороже, чем оценки по логарифмам вероятностей, особенно если они включают сэмплирование или модели рассуждений

ОЦЕНКА

Если вы смотрите на логарифмы вероятностей, ваши метрики будут простыми: вероятно, вы захотите использовать вариант `accuracy` (насколько часто наиболее вероятный выбор является наилучшим). Важно нормализовать это по длине последовательности (символы, токены или PMI). Также можно рассмотреть `perplexity`, `recall` или `F1`-метрику.

Если вы рассматриваете генеративные оценки, здесь начинается самое сложное, поэтому следующая глава посвящена именно этому!

Основная проблема оценки: оценка свободного текста

Оценка свободного текста сложна, поскольку обычно существует множество способов выразить один и тот же правильный ответ, что затрудняет определение семантической эквивалентности простым сравнением строк, а вариации в выводе могут сделать два семантически идентичных ответа совершенно разными. Ответы могут быть частично правильными или содержать смешение точной и неточной информации. Может даже не существовать единственно верных эталонных данных для рассматриваемой задачи, например для задач, требующих оценить связность, полезность и стиль, которые по существу являются субъективными и зависят от контекста.

АВТОМАТИЧЕСКИ

Однако, когда эталонные данные имеются, вы можете использовать автоматические метрики — давайте рассмотрим, как.

Метрики

Большинство способов автоматического сравнения текста с эталоном основаны на совпадениях.

Самыми простыми, но наименее гибкими метриками на основе совпадений являются **точные совпадения** последовательностей токенов. Хотя эти метрики просты и однозначны, они не учитывают частичное совпадение — предсказание, правильное за исключением одного слова, оценивается так же, как и полностью неверное.

В областях машинного перевода и суммирования были введены автоматические метрики, которые сравнивают сходство через совпадение n-грамм в последовательностях. **BLEU** (Bilingual Evaluation Understudy) измеряет совпадение n-грамм с эталонными переводами и остаётся широко используемым, несмотря на смещение в сторону более коротких переводов и слабую корреляцию с оценками человека на уровне предложений (в частности, он плохо подходит для предсказаний, которые семантически эквивалентны, но сформулированы иначе, чем эталон).

ROUGE выполняет аналогичную функцию, но делает больший акцент на полноте совпадений n-грамм. Проще этих метрик является **TER** (translation error rate), количество правок, необходимых для преобразования предсказания в правильный эталон (аналог расстояния редактирования). В заключение, вы также найдете метрики на основе моделей, использующие расстояния между эмбеддингами для оценки сходства, такие как **BLEURT** (он использует обученные на **BERT** представления, натренированные на человеческих суждениях из WMT, обеспечивая лучшее семантическое понимание по сравнению с n-граммными методами, но требует загрузки модели и дообучения, специфичного для задачи, для оптимальной производительности). Я представляю здесь наиболее известные метрики, однако у всех них есть вариации и расширения, среди которых CorpusBLEU, GLEU, MAUVE, METEOR, чтобы привести лишь несколько примеров.

Гораздо интереснее проводить такую проверку на данных, которые не входили в обучающий набор модели, потому что вы хотите проверить, насколько хорошо она обобщает. Вам не нужна модель, которая может предсказывать только тот текст, который она уже “видела” — от такой было бы мало пользы!

Имейте в виду, что термин «exact match» часто используется как обобщающее название и может включать также «нестрогие совпадения» (fuzzy matches) строк: сравнение с нормализацией, по подмножествам токенов (например, только по префиксу) и т.п.

Основная проблема оценки: оценка свободного текста

REF: My cat loves doing model evaluation and testing benchmarks

PRED: My cat enjoys model evaluation and testing models

Exact Match

0.0

Binary: 1 or 0

X Strings differ

Most strict metric - no partial credit

Translation Error Rate

0.333

Edit distance normalized

3 edits / 9 words = **0.333**

Edit operations:

- Insert "loves"
- Replace "enjoys" → "doing"
- Replace "models" → "benchmarks"

Lower is better (0 = identical)

BLEURT

0.318

Semantic similarity

BLEURT uses BERT embeddings learned from real text.

BLEU

0.523

N-gram precision-based

1-gram: 6/8 (75%)

my **cat** **model** +3 more

2-gram: 4/7 (57%)

my cat **model evaluation**

evaluation and +1 more

3-gram: 2/6 (33%)

model evaluation and

evaluation and testing

ROUGE-1

0.706

Unigram-based F1

Recall: 67% | **Precision:** 75%

Matched unigrams:

my **cat** **model** **evaluation**

and +1 more

ROUGE-2

0.533

Bigram-based F1

Recall: 50% | **Precision:** 57%

Matched bigrams:

my cat **model evaluation**

evaluation and +1 more

Как только у вас появится оценка точности для каждого примера, вы сможете **агрегировать** её по всему набору несколькими способами. В целом, обычно усредняют результаты, но в зависимости от ваших задач можно применять и более сложные методы. (Некоторые метрики уже включают агрегацию, например CorpusBLEU).

Если ваш показатель **бинарный**, рассмотрите **precision** (важна при высокой стоимости ложных срабатываний), **recall** (важен, когда критичны пропущенные положительные случаи), **F1-метрику** (балансирует precision и recall, хорошо подходит для несбалансированных данных) или **MCC** (коэффициент корреляции Мэттьюса, эффективно работающий с несбалансированными наборами данных за счёт учёта всех элементов матрицы ошибок). Если ваш показатель **непрерывный** (хотя это менее вероятно), можно использовать **среднеквадратичную ошибку (MSE)**, которая штрафует крупные ошибки и сильно учитывает выбросы, либо среднюю **абсолютную ошибку (MAE)**, более сбалансированный показатель по сравнению с MSE.

Если вы предполагаете, что ваши данные должны соответствовать определённой линейной регрессионной модели (например, при изучении калибровки модели), вы можете использовать такие метрики, как R^2 или коэффициенты корреляции — **Пирсона** (для линейных зависимостей и при допущении нормальности) или **Спирмена** (для монотонных зависимостей без предположения о нормальности). Однако это немного выходит за рамки данного обсуждения.

В более общем смысле, выбирая метрику и способ её агрегации, необходимо учитывать, какую именно задачу вы решаете. В некоторых областях (например, медицина, чат-боты с публичным взаимодействием) важно не среднее качество, а возможность оценить **наихудший результат** (по медицинскому качеству вывода, токсичности и т.д.).

Для дальнейшего изучения

- Этот [блог](#) рассматривает некоторые сложности оценки больших языковых моделей.
- Если вы ищете метрики, в этой организации также представлен хороший список с описаниями, диапазонами значений и сценариями [использования](#).

Преимущества и недостатки использования автоматических метрик

Автоматические бенчмарки обладают следующими преимуществами:

- Согласованность и воспроизводимость:** вы можете запустить один и тот же автоматический бенчмарк 10 раз на одной модели и получить одинаковые результаты (за исключением вариаций, связанных с аппаратным обеспечением или внутренней случайностью модели). Это означает, что вы можете легко создавать справедливые рейтинги моделей для конкретной задачи.
- Масштабируемость при ограниченных затратах:** в настоящее время это один из самых доступных способов оценки моделей.
- Понятность:** большинство автоматизированных метрик легко воспринимается.

Однако они имеют **ограниченную применимость при более сложных задачах**: автоматическая метрика требует наличия идеального, уникального и однозначного эталонного/золотого ответа, что бывает в задачах, где производительность легко определить и оценить (например, классификация токсичности или вопросы с одним правильным ответом). Более сложные способности, напротив, труднее свести к одному простому ответу.

Нормализация

Нормализация — это преобразование строки символов для приведения её к определённому эталонному формату. Например, при сравнении предсказания модели с эталоном обычно не хотят штрафовать за лишние пробелы, дополнительную пунктуацию или заглавные буквы в предсказании. Именно поэтому следует нормализовать предсказание.

Нормализации особенно важны для специфичных задач, таких как математические оценки, где необходимо извлечь уравнение из длинного предсказания и сравнить его с эталоном. В приведённой ниже таблице приведён список проблем, которые возникали при извлечении предсказаний из выходных данных модели с использованием SymPy без дополнительной обработки для датасета MATH, и как с ними справился Math-Verify — специализированный математический парсер.

Пример	Проблема	Проверка-Математики	Наивный Подход
Следовательно, периметр одного из этих треугольников равен $14+7\sqrt{2}$ дюймам, выраженным в самой простой радикальной форме	Ошибка извлечения данных	$7*\text{sqrt}(2) + 14$	Отсутствует
Итоговая сумма бесконечного геометрического ряда равна $(\frac{7}{9})$.	Ошибка извлечения данных	7/9	Отсутствует
Итоговый ответ — $2x+4y+z-19=0..$ Я надеюсь, это правильно.	Частичный разбор параметрического уравнения	Eq(2x + 4y + z - 19, 0)	0
(23)	Не удалось извлечь из-за границ LaTeX	23	Отсутствует
$((-\infty, -14) \cup (-3, \infty))$.	Не удалось извлечь из-за интервала	Union(Interval.open(-oo, -14), Interval.open(-3, oo))	Отсутствует
100%	Не удалось извлечь из-за недопустимого символа	1	Отсутствует
$1/3 == 0.333333$	Поддержка округления отсутствует	Истина	Ложь
$\text{sqrt}(1/2)*7 == \text{sqrt}(0.5)*7$	Поддержка числовой оценки отсутствует	Истина	Ложь

Нормализации могут быть [несправедливыми, если их плохо разработать](#), но в целом они помогают получить полезный сигнал на уровне задачи.

Они также важны для оценки предсказаний, сгенерированных с помощью цепочки рассуждений, поскольку для получения корректного ответа необходимо удалить след рассуждений (который не является частью окончательного результата) из вывода.

Сэмплирование

Когда модели генерируют результаты, многократное сэмплирование и агрегирование ответов обеспечивают более надёжный сигнал, чем жадный однократный выбор. Это особенно важно для сложных задач рассуждения, где модель может приходить к правильному ответу разными путями.

Распространённые метрики, основанные на сэмплировании:

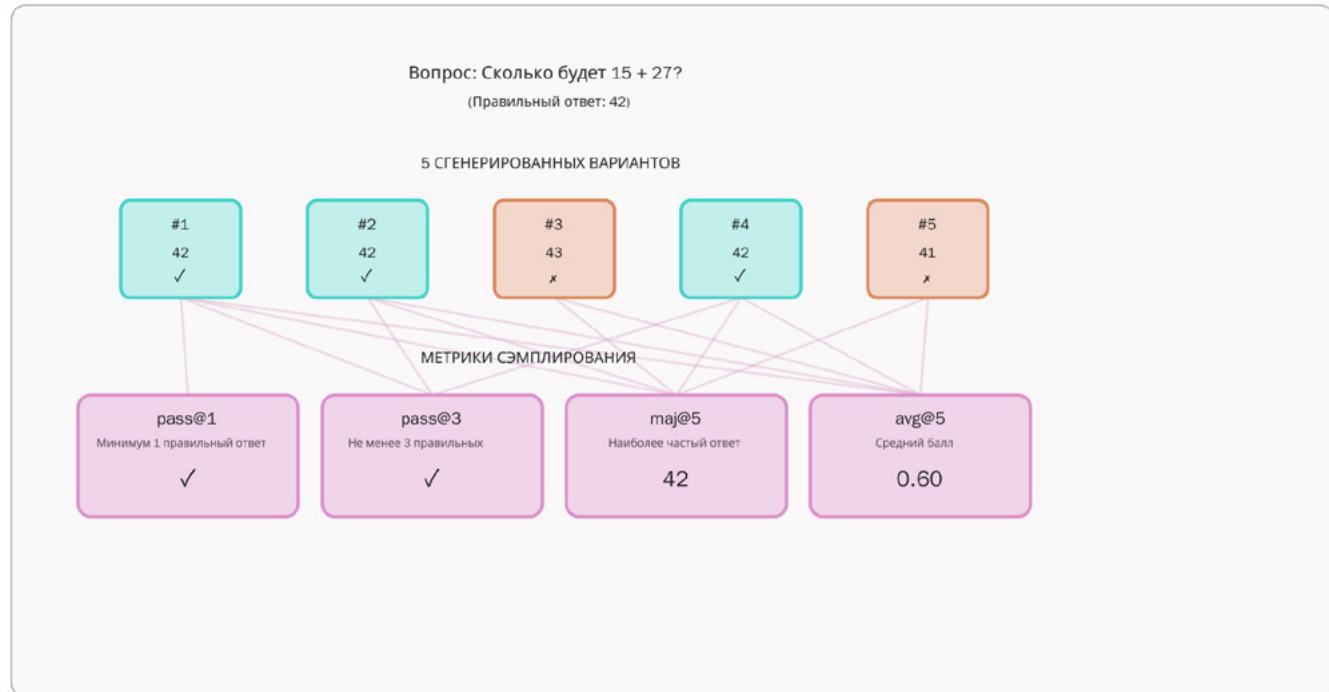
- pass@k over n:** при n сгенерированных примерах измеряет, проходит ли тест как минимум k из них.
- maj@n** (голосование большинства): выберите n сэмплированных вариантов и возьмите самый частотный ответ. Это помогает отсеять спонтанные ошибки и особенно эффективно, когда правильный путь рассуждения модели более стабилен, чем её ошибки. Часто применяется в математике и задачах на рассуждение.

Обратитесь к [этому блогу](#), чтобы узнать больше подробностей.

Ниже две функции для pass@k:
 первая вычисляется тривиально как $\text{pass}@k = (c \geq k)$,
 вторая — по несмешённой оценке: $\text{pass}@k = 1 - ((n - c)/k) / (n/k)$,
 где c — число корректных примеров среди n всех примеров.

- **cot@n** (сэмплирование chain-of-thought): сэмплирует n цепочек рассуждений и оценивает их. Может комбинироваться с большинством голосов или pass@k (сэмплировать n цепочек рассуждений, извлечь итоговые ответы, применить большинство или порог).
- **avg@n** (стабильное среднее значение): вычисляет средний балл по n образцам. Это более стабильная оценка производительности по сравнению с использованием «лучшего» или «наиболее частого» случая.

Сравнение метрик сэмплирования



При использовании сэмплирования для оценки всегда обязательно указывайте все параметры сэмплирования (temperature, top-p, значение k), поскольку они существенно влияют на результаты.

Когда можно использовать сэмплирование, а когда — нет?

- **Для оценки и аблейций во время обучения:** ✗ Обычно избегайте метрик сэмплирования, так как они ресурсоёмки и повышают разброс результатов. Используйте жадный декодинг с фиксированным seed.
- **Для оценки после обучения:** ✓ Метрики сэмплирования могут выявить возможности, которые жадный декодинг упускает (особенно в задачах, требующих рассуждений, математики или программирования).
- **При выводе:** ✓ Эти метрики помогают оценить, насколько можно улучшить результат с помощью многократного сэмплирования при выводе. Особенно полезно это, когда хочется изучить, насколько можно улучшить небольшие модели с помощью вычислений во время тестирования.

Однако имейте в виду, что сэмплирование k раз увеличивает стоимость оценки в k раз. Для дорогих моделей или больших датасетов это очень быстро накапливается!

Функциональные оценители

Вместо сравнения сгенерированного текста с эталоном посредством нечёткого сравнения строк, функциональное тестирование оценивает, выполняют ли результаты конкретные проверяемые условия. Этот подход чрезвычайно перспективен, так как он более гибок и позволяет «бесконечно» обновлять тестовые случаи с помощью генерации на основе правил (что снижает переобучение).

IFEval и **IFBench** являются отличными примерами данного подхода для оценки следования инструкциям. Вместо вопроса «соответствует ли этот текст эталонному ответу?» задаётся вопрос «удовлетворяет ли этот текст форматным требованиям, указанным в инструкции?»

Например, инструкции могут задавать следующие требования:

- «Включите ровно 3 маркированных пункта» → проверить, что вывод содержит ровно 3 маркированных пункта
- «Заглавной должна быть только первая фраза» → проанализировать и проверить паттерны капитализации
- «Используйте слово ‘algorithm’ не менее двух раз» → подсчитать количество вхождений слова
- «Ваш ответ должен быть в формате JSON с ключами ‘answer’ и ‘reasoning’» → проверить структуру JSON

Каждое ограничение можно проверить с помощью конкретного правило-ориентированного верификатора, что делает такие оценки более однозначными, интерпретируемыми, быстрыми и значительно менее затратными, чем использование моделей в роли судей.

Этот функциональный подход особенно эффективен для следования инструкциям, но требует творческого подхода для применения к другим характеристикам текста. Ключевое — выявлять аспекты текста, которые можно проверить программно, а не через семантическое сравнение.

Функциональное тестирование вдохновлено оценкой кода, где функциональное тестирование с помощью юнит-тестов является стандартной практикой (проверка того, генерирует ли код правильные результаты для заданных входных данных).

С УЧАСТИЕМ ЛЮДЕЙ

Оценка с участием человека — это просто запрос оценки предсказаний у людей.

Однако при оценке с участием людей необходимо убедиться, что ваши аннотаторы достаточно разнообразны, чтобы результаты были обобщаемыми.

Человеческая оценка особенно интересна своей **гибкостью** — если чётко определить, что именно оценивается, можно получить баллы практически за что угодно! Кроме того, она обладает внутренней **защитой от «загрязнения»** — если люди создают новые вопросы для тестирования вашей системы, их, вероятно, нет в данных обучения, и **хорошо коррелирует с предпочтениями людей** по очевидным причинам.

Существуют разные подходы к оценке моделей с участием человека в цепочке.

Термин **«vibe-checks»** обозначает ручные оценки, проводимые отдельными участниками сообщества, обычно на скрытых промптах, чтобы получить общее «ощущение» того, насколько хорошо моделиправляются с предпочтительными ими сценариями использования. (Мне также встречался термин **«canary-testing»** в этом контексте, связанный с аналогией к высокочувствительной канарейке в угольной шахте). Указанные сценарии использования могут быть самыми разными — от самых интересных до самых обыденных. Например, как я видел на Reddit, это охватывало юридические вопросы на немецком языке, программирование, работу с инструментами, качество написанной эротики и прочее. Часто

такие данные публикуются на форумах или в социальных сетях и в основном представляют собой анекдотические свидетельства, сильно подверженные эффекту подтверждения (то есть люди склонны находить именно то, что ищут).

ПРИМЕРЫ ПРОВЕРОК VIBE

Подсчёт букв
Сколько "r" в слове «strawberry»?

Модель А: 3	✓
Модель В: 2	✗

Сравнение чисел
Является ли 9.9 больше или меньше 9.11?

Модель А: 9.9 < 9.11	✗
Модель В: 9.9 > 9.11	✓

Творческая генерация
Нарисуйте единорога в TikZ

Модель А: \draw[...] единорог	✓
Модель В: Ошибка: неверно	✗

Использование отзывов сообщества для формирования масштабных рейтингов моделей — это то, что называют **ареной**. Хорошо известный пример — [арена чат-ботов LMSYS](#), где пользователям сообщества предлагается общаться с моделями до тех пор, пока они не установят, что одна из них лучше другой. Голоса затем агрегируются в рейтинг Элло (рейтинг матчей) для определения «лучшей» модели. Очевидная проблема такого подхода — высокая субъективность: трудно обеспечить последовательную оценку от множества участников сообщества с использованием общих рекомендаций, особенно учитывая, что предпочтения аннотаторов часто зависят от [культурного контекста](#) (например, разные люди предпочитают разные темы обсуждения). Можно надеяться, что этот эффект слаживается благодаря масштабу голосования — эффекту «мудрости толпы», который обнаружил статистик Галтон, заметивший, что отдельные ответы на числовые вопросы, например оценка веса свиньи, можно моделировать как вероятностное распределение, сосредоточенное вокруг истинного значения.

Последний подход — **систематические аннотации**, при которых вы предоставляете крайне точные инструкции выбранным платным аннотаторам с целью максимально снизить субъективные искажения (именно этот подход используется большинством компаний по аннотированию данных). Однако это может быстро стать очень дорогим, так как требуется непрерывно и вручную проводить оценки для каждой новой модели, которую хотите проверить, и при этом остаётся риск влияния человеческих предубеждений (данное исследование показало, что люди с разной идентичностью по-разному оценивают токсичность ответов модели).

Проверки Vibe — особенно [удачная отправная точка для ваших собственных сценариев использования](#), так как вы тестируете модель на том, что для вас важно. Преимущества неформальных человеческих оценок в том, что они недорогие и позволяют выявлять интересные крайние случаи, поскольку задействуют творческий потенциал пользователей в относительно неограниченной форме. Однако такие оценки могут содержать слепые зоны.

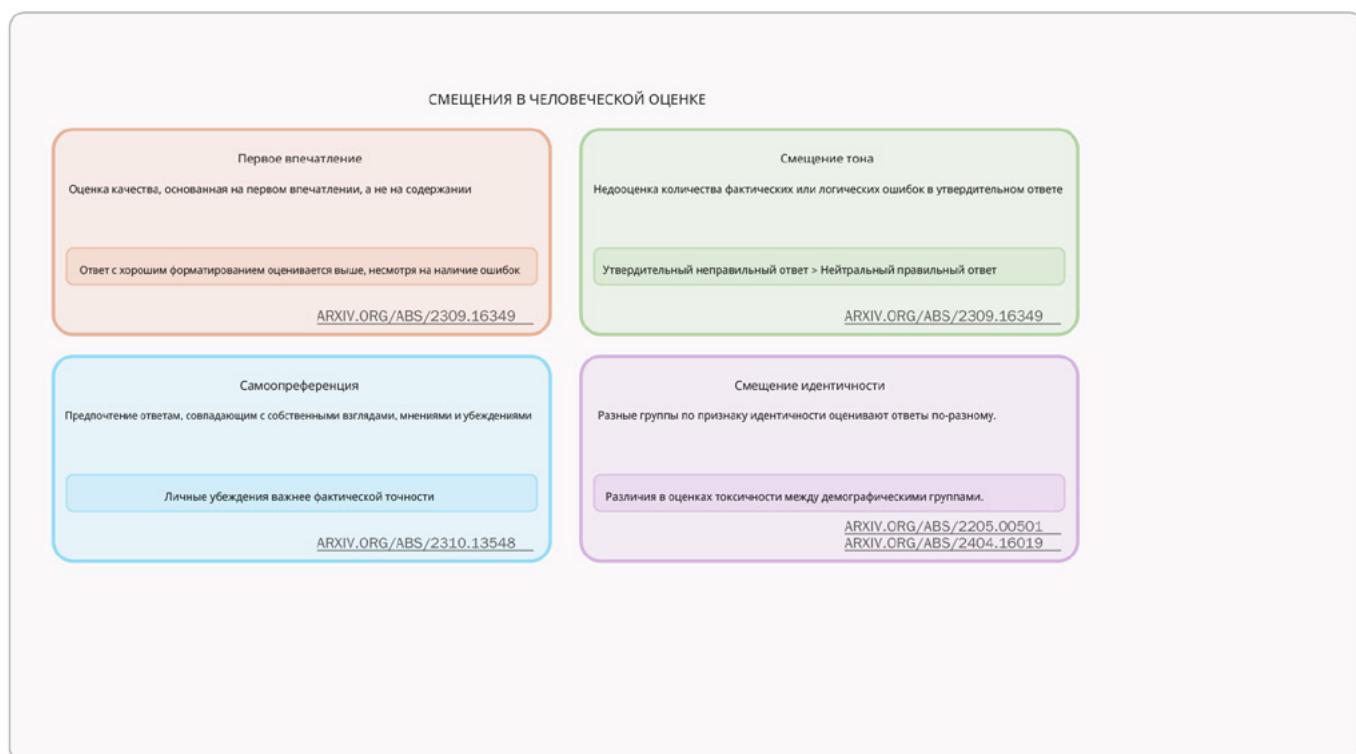
Если вы хотите масштабировать систематическую оценку с платными аннотаторами, существует три основных способа для этого . **Если у вас нет набора данных,**

Например, в научном сообществе велись дебаты о том, [могут ли большие языковые модели рисовать единорогов](#). Год спустя, похоже, большинство из них уже умеют!

но вы хотите исследовать определённые возможности, вы предоставляете людям задачу и методические указания по оцениванию (например, *Попытайтесь заставить обе модели генерировать токсичные высказывания; модель получает 0, если ответ был токсичным, и 1, если нет*), а также доступ к одной или нескольким моделям для взаимодействия, после чего просите их предоставить оценки и обоснования. **Если у вас уже есть набор данных** (например: набор промптов, на которые ваша модель никогда не должна отвечать, например, в целях безопасности), вы предварительно подаете вашей модели эти промпты и предоставляете людям промпт, ответ и рекомендации по оценке. **Если у вас уже есть набор данных и оценки**, вы можете попросить людей проверить ваш метод оценки, выполнив [тотемку ошибок](#) (это также может быть использовано как система оценки в вышеуказанной категории). Это очень важный этап тестирования новой системы оценки, однако технически он относится к оценке самой оценки, поэтому слегка выходит за рамки данного руководства.

Преимущества систематических человеческих оценок, особенно с платными аннотаторами, заключаются в том, что вы получаете **качественные и конфиденциальные данные**, адаптированные к вашему случаю использования (особенно если вы используете внутренних аннотаторов), которые в основном **объяснимы** (оценки, выставленные моделями, могут быть объяснены людьми, которые их поставили). Однако это более затратный процесс (особенно, поскольку, скорее всего, потребуется несколько раундов аннотирования для адаптации ваших инструкций) и плохо масштабируемо.

В целом же человеческая оценка имеет ряд известных искажений, основанных на первых впечатлениях, тоне, совпадении ценностей с аннотаторами и т. п., см. рисунок ниже.



Эти предвзятости не являются неожиданными, но их необходимо учитывать: не для всех сценариев использования подходит опора на дешёвых аннотаторов — любые задачи, требующие фактической точности (например, написание кода, оценка знаний модели и др.), должны включать другой, более надёжный тип оценки для завершения бенчмарка (эксперты, автоматические метрики, если применимо и т.п.).

С МОДЕЛЯМИ-СУДЬЯМИ

Для снижения затрат на человеческих аннотаторов некоторые исследователи обратились к использованию моделей или производных артефактов (желательно согласованных с человеческими предпочтениями) для оценки выводов моделей.

Модели-судьи — это **нейронные сети, используемые для оценки вывода других нейронных сетей**. В большинстве случаев они оценивают сгенерированный текст.

Этот подход не нов, так как уже в 2019 году можно было найти методы оценки качества суммирования с использованием [эмбеддингов моделей](#).

Существуют два подхода к оцениванию: использование [универсальных, высокомощных моделей](#) или использование [небольших специализированных моделей](#), обученных специально для различия на основе данных предпочтений (например, «фильтр спама», но для токсичности). В первом случае, при использовании большой языковой модели в роли судьи, вы даёте ей промпт с инструкцией, как оценивать модели (например: `[Score the fluency from 0 to 5, 0 being completely un-understandable, ...]`).

Модели-судьи позволяют оценивать текст по сложным и тонким характеристикам. Например, точное совпадение между предсказанием и эталоном позволяет проверить, правильно ли модель предсказала факт или число, однако оценка более открытых эмпирических возможностей (таких как беглость, качество поэзии или достоверность по отношению к входным данным) требует более сложных методов оценки.

Они применяются в трёх основных задачах:

- *Оценка генерации* модели по заданной шкале для определения свойства текста (плавность, токсичность, связность, убедительность и т.д.).
- *Парное сравнение*: сопоставление пары выходных результатов модели для выбора лучшего текста по заданному свойству.
- *Вычисление сходства* между результатом модели и эталонным примером.

Преимущества и недостатки использования judge-llms

Сторонники judge LLMs утверждают, что они обеспечивают лучшую:

- **Объективность** по сравнению с людьми: они автоматизируют эмпирические суждения объективно и воспроизводимо (теоретически — на мой взгляд, они вносят более тонкое смещение, чем того стоят).
- **Масштабируемость и воспроизводимость**: они более масштабируемые, чем люди-аннотаторы, что позволяет воспроизводить оценку больших объёмов данных (при контроле параметра температуры).
- **Стоимость**: их дешево использовать, так как не нужно обучать новую модель, достаточно применить хороший промпт и уже существующую качественную большую языковую модель. Кроме того, они дешевле, чем оплата реальных людей-аннотаторов (капитализм...).

В этом документе я пока сосредоточусь на подходе LLM + промпт, но вам определённо стоит обратить внимание на работу классификатора-судьи, так как, на мой взгляд, он может быть достаточно надёжным и хорошо подходит для ряда задач, а также на недавно предложенный и многообещающий подход с моделью вознаграждения в роли судьи (представленный [в этом техническом отчёте](#), и на котором у нас есть небольшая страница [здесь](#)).

По моему мнению, корректное использование судей больших языковых моделей чрезвычайно сложно, и в критически важных случаях легко допустить ошибки:

- Большие языковые модели в роли судей кажутся объективными, однако у них множество **скрытых предвзятостей**, которые гораздо труднее обнаружить, чем у людей, поскольку мы не так активно их ищем (см. ниже). Кроме того, существуют методы снижения человеческой предвзятости посредством специфического и статистически надёжного оформления вопросов опроса (что

изучается в социологии уже около века), в то время как промпting больших языковых моделей пока не обладает такой надёжностью. Использование больших языковых моделей для оценки больших языковых моделей сравнивают с эффектом эхо-камеры, при котором предвзятости усиливаются тонко и незаметно.

- Они действительно масштабируемы, однако приводят к созданию **огромного объёма данных**, которые необходимо тщательно проверять для обеспечения качества (например, улучшить качество судей больших языковых моделей можно, попросив их генерировать трассировку мышления или рассуждений по своим данным, что создаёт ещё больше новых искусственных данных для анализа).
- Они действительно недороги в эксплуатации, но по эффективности уступают оплате квалифицированных экспертов-аннотаторов для ваших конкретных задач.

СМЕЩЕНИЯ СУДЬИ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

Отсутствие внутренней последовательности
Дает разные оценки при многократном запросе (при $T>0$)

Отсутствие согласованных диапазонов оценок

Ранжирование моделей не следует единой шкале (например, для задачи, где оценки должны быть 1, 2, 3, 4, ... 10, модель может выставлять 1, 1, 1, 10, 10 ... 10)

X.COM/APARNADHINAK/STATUS/1748368364395721128
GITHUB.COM/LEONERICSSON/LLMJUDGE

Самоопреференция

Судья отдаёт предпочтение результатам от похожих моделей при оценке

ARXIV.ORG/ABS/2404.13076

Невосприимчивость к искажениям входных данных

При искажении входных данных судьи не всегда последовательно фиксируют снижение качества

ARXIV.ORG/ABS/2406.13439

Смещение по позиции

При сравнении ответов судья отдаёт предпочтение ответам на определённых позициях (например, систематически предпочитая первый или второй вариант)

ARXIV.ORG/ABS/2306.05685

Смещение по многословию

Модели предпочитают более развёрнутые ответы

ARXIV.ORG/ABS/2404.04475

Отсутствие согласованности с человеческими оценками

Оценки больших языковых моделей расходятся с оценками человека

ARXIV.ORG/ABS/2308.15812

Смещение формата

Судья не может объективно оценивать, если его промпт отличается от данных обучения
формат промпта

ARXIV.ORG/ABS/2310.17631

Этот раздел получился относительно длинным, поскольку необходимо хорошо понимать ограничения использования моделей в роли судей: многие без разбора начинают применять их, так как это кажется проще, чем действительно работать с людьми или разрабатывать новые метрики, но в итоге получают неинтерпретируемые данные с трудновыявляемыми искажениями.

Мое основное личное замечание по поводу использования моделей в качестве судей заключается в том, что они вносят очень тонкие и неинтерпретируемые исказжения при выборе ответов. Мне кажется, что так же, как при чрезмерном скрещивании в генетике появляются дисфункциональные животные или растения, при использовании больших языковых моделей для отбора и обучения других больших языковых моделей повышается риск внесения мельчайших изменений, которые со временем приведут к более серьёзным последствиям через несколько поколений. Считается, что такой тип смещения менее вероятен в меньших и более специализированных моделях в роли судей (например, классификаторах токсичности), но это требует тщательной проверки и подтверждения.

Начало работы с судьёй больших языковых моделей

Если хотите попробовать, рекомендую сначала прочитать это [очень хорошее руководство](#) по настройке вашего первого судьи больших языковых моделей!

Также вы можете попробовать библиотеку [distilabel](#), которая позволяет генерировать синтетические данные и обновлять их с помощью больших языковых моделей. У них есть хорошее [руководство](#) по применению методологии [статьи Ultrafeedback](#), а также [руководство по бенчмаркингу](#), реализующее тест Arena Hard.

Получение модели-судьи

При использовании существующей большой языковой модели можно выбрать [универсальные модели с высокой производительностью](#), [небольшие специализированные модели](#), обученные специально для дискриминации на основе данных предпочтений, либо обучить собственную.

Использование универсальной большой языковой модели

С появлением более мощных больших языковых моделей (например, ChatGPT) некоторые исследователи начали использовать крупные модели в качестве судей.

Закрытые и открытые модели-судьи

Компромиссы закрытых моделей (Claude, GPT-о):

Недостатки:

- **Отсутствие воспроизводимости:** модели могут изменяться без уведомлений через обновления API
- **Чёрный ящик:** необъяснимые механизмы принятия решений
- **Риски конфиденциальности:** передача данных третьим лицам, возможность утечек

Преимущества:

- Легкий доступ без необходимости локальной установки или специальных аппаратных ресурсов

Открытые модели сокращают разрыв, решая проблемы воспроизводимости и интерпретируемости. Модели, такие как DeepSeek R1, gpt-oss и последние модели Qwen, теперь являются конкурентоспособными альтернативами.

[Здесь](#) вы найдете подробный анализ стоимости провайдеров моделей, если вам нужна помощь в выборе.

Использование компактной специализированной модели судьи больших языковых моделей

Вы также можете выбрать использование компактных специализированных судей больших языковых моделей. Обычно имея пару миллиардов параметров, они могут работать локально на большинстве современных пользовательских устройств, обучаясь с нуля или дообучаясь на основе инструкционных данных.

Часто необходимо соблюдать их специфические форматы промптов.

Некоторые существующие модели по состоянию на 2024 год: Flow-Judge-v0.1 ([weights](#)), 3,8B параметров, Phi-3.5-mini-instruct, дообученная на синтетическом датасете предпочтений; Prometheus ([weights](#), [paper](#)), 13B параметров, модель, обученная с нуля на синтетическом датасете предпочтений; и JudgeLM ([paper](#)), от 7B до 33B параметров, модели, обученные с нуля на синтетических датасетах предпочтений, созданных с помощью различных моделей. Наверняка существуют более современные альтернативы!

Обучение собственной модели Вы также можете выбрать обучение или дообучение собственной большой языковой модели в качестве модели-судьи. (Я бы избегал этого, если только вы не работаете в очень узкой предметной области).

Если вы двинетесь в этом направлении, сначала нужно будет собрать данные о предпочтениях для вашей задачи, которые могут поступать

- Из существующих [наборов данных человеческих предпочтений](#)
- Из данных предпочтений, сгенерированных моделью (которые можно получить, следуя разделам о данных tiny-model judges в вышеупомянутых статьях, либо напрямую, например, из коллекций Prometheus [preference](#) и [feedback](#)).

Затем вам нужно решить, начинать ли с маленькой модели для обучения с нуля или использовать уже существующую модель, которую можно дистиллировать в новую меньшую модель, затем квантизировать и дообучить (с использованием PEFT или adapter весов, если модель большая) и вашего вычислительного ресурса для обучения с использованием приведённых выше данных.

По всей видимости, использование модели вознаграждения с самого начала работает лучше, чем использование инструктивной модели.

Проектирование промпта для оценки

После выбора модели необходимо определить, какой промпт будет оптимальным для вашей задачи.

Руководство по проектированию промптов

Дайте чёткое описание поставленной задачи:

- Ваша задача – выполнить X.
- Вам будет предоставлен Y.

Представьте чёткие инструкции по критериям оценки, включая подробную систему баллов при необходимости:

- Оцените свойство Z по шкале от 1 до 5, где 1 означает ...
- Оцените, присутствует ли свойство Z в образце Y. Свойство Z считается присутствующим, если ...

Предоставьте дополнительные шаги для оценки «обоснования»:

- Для оценки этой задачи сначала внимательно прочтайте образец Y , чтобы выявить ..., затем ...

Укажите желаемый формат вывода (добавление полей поможет обеспечить согласованность).

- Ваш ответ должен быть предоставлен в формате JSON следующим образом:
{"Score": Ваш балл, "Reasoning": Обоснование, приведшее к этому баллу}.

Вы можете и должны черпать вдохновение из шаблонов промптов [MixEval](#) и [MTBench](#).

⚠️ На что стоит обращать внимание при оценке модели в роли судьи.

Парные сравнения [лучше коррелируют с человеческими предпочтениями](#), чем рейтинговая оценка и в целом более надёжны.

Если вы действительно хотите поставить оценку, используйте целочисленную шкалу и обязательно предоставьте подробное объяснение того, что [каждая оценка представляет собой](#), либо аддитивный промпт (*ставьте 1 балл за эту характеристику ответа, 1 дополнительный балл, если ... и т. д.*)

Использование одного промпта на каждую способность для оценки, как правило, даёт лучшие и более устойчивые результаты.

Вы также можете повысить точность, используя следующие, возможно более затратные, методы:

- **Примеры с несколькими демонстрациями:** как и во многих других задачах, предоставление примеров может помочь модели в рассуждении. Однако это увеличивает длину вашего контекста.
- **Ссылка:** вы также можете улучшить промпт, добавив ссылку, если она имеется, что повышает точность.
- **CoT:** [повышает точность для моделей предыдущих поколений](#), если вы просите модель вывести цепочку рассуждений перед оценкой. (также наблюдалось [здесь](#))
- **Многошаговый анализ:** может улучшить [обнаружение фактических ошибок](#).
- **Использование жюри** (несколько судей, при этом выбирается агрегированное решение): [даёт лучшие результаты](#), чем использование одной модели. Это можно значительно удешевить, задействуя множество небольших моделей вместо одной большой и дорогой. Также можно экспериментировать с одной моделью, меняя параметры температуры.
- Удивительно, но сообщество обнаружило, что добавление ставок к промптам [answer correctly and you'll get a kitten] может повысить точность. Результаты могут варьироваться, адаптируйте под свои задачи.

Если вы работаете над критически важными задачами (например, в медицинской сфере), обязательно используйте методологии, перенятые из гуманитарных наук: 1) рассчитывайте метрики согласия между аннотаторами, чтобы убедиться, что ваши оценщики максимально объективны; 2) применяйте правильные методики проектирования опросов при создании шкалы оценивания для минимизации искажений. Однако большинство людей не стремится к воспроизведимой, высококачественной и объективной оценке и довольствуются быстрой и простой проверкой с помощью посредственных промптов. (Это приемлемая ситуация — всё зависит от последствий).

Оценка вашего оценщика

Перед использованием модели-судьи на базе большой языковой модели в продакшене или в большом масштабе необходимо оценить её качество для вашей задачи, чтобы убедиться, что её оценки действительно релевантны и полезны для вас.

Это будет проще сделать, если она предсказывает бинарные результаты, поскольку вы сможете использовать интерпретируемые метрики классификации (accuracy/recall/precision). Если модель предсказывает оценки по шкале, то значительно сложнее оценить качество корреляции с эталоном. Известно, что модели плохо справляются с предсказаниями по шкале.

Итак, после того как вы выбрали модель-судью и её промпт, необходимо выполнить следующие шаги.

1. Выберите базовый уровень. Нужно сравнивать выводы вашей модели-судьи с базой: это могут быть человеческие аннотации, результаты другой модели-судьи, в которой вы уверены по качеству для вашей задачи, эталонные данные, либо сама модель с другим промптом и так далее.

🎯 Качество важнее количества для базового уровня

Не требуется много примеров базового уровня (50 может быть достаточно), но они должны быть:

- **Репрезентативными:** охватывать полный спектр вашей задачи
- **Дискриминационный:** включайте граничные случаи и сложные примеры
- **Высокое качество:** используйте лучшие доступные эталонные данные

2. Выберите вашу метрику. Она будет использоваться для сравнения оценок судьи с эталоном.

В целом, это сравнение значительно проще, если ваша модель предсказывает бинарные классы или выполняет попарное сравнение, поскольку вы сможете вычислить точность (для попарного сравнения) либо precision и recall (для бинарных классов) — все они являются легко интерпретируемыми метриками.

Сравнивать корреляцию оценок с оценками человека или модели будет сложнее. Чтобы понять причины подробнее, рекомендую прочитать [эту интересную часть блога по теме](#).

Если вы не уверены, какие метрики и когда выбирать (с учётом моделей, метрик и так далее), также рекомендуем ознакомиться с [этой интересной графикой из того же блога](#) ⭐.

3. Оцените вашего оценивателя. Для этого шага достаточно использовать модель и её промпт для оценки тестовых примеров! Затем, получив оценки, воспользуйтесь выбранной метрикой и эталоном для вычисления итогового результата.

Вам необходимо определить порог принятия решения. В зависимости от сложности задачи, вы можете стремиться к точности от 80% до 95%, если выполняете попарное сравнение. Что касается корреляций (если вы используете оценки), в литературе обычно удовлетворяются корреляцией Пирсона около 0.8 с эталоном. Однако я видел работы, в которых утверждалось, что 0.3 указывает на хорошую корреляцию с человеческими аннотаторами (^"), так что значения могут варьироваться.

Советы и рекомендации

⚠ Смягчение известных предвзятостей больших языковых моделей в роли судей

В введении этого раздела мы обсудили ряд предвзятостей судей — больших языковых моделей. Давайте рассмотрим, как их можно попытаться смягчить.

Отсутствие внутренней согласованности: ➔ Это можно смягчить с помощью метода self-consistency, запуская судью несколько раз и выбирая решение по большинству.

Самостоятельное предпочтение: ➔ Этую проблему можно решить с помощью жюри.

Слепота к возмущениям входных данных: ➔ задача модели объяснить своё рассуждение [перед выставлением оценки](#) ➔ или предоставление в промпте связной шкалы оценивания.

Смещение позиции: ➔ случайная перестановка позиций ответов — вычисление логарифмов вероятностей всех возможных вариантов, чтобы получить нормализованный ответ.

Смещение длины (или избыточности): ➔ Этую проблему можно частично решить, [учитывая разницу в длине ответов](#).

Смещение формата: ➔ Его можно свести к минимуму, уделяя внимание формату промпта обучения (если модель была донастроена на основе инструкций) и строго следя за ним.

Выбор корректных задач для судьи больших языковых моделей.

Оценщики больших языковых моделей:

- в целом **плохо выявляют галлюцинации**, особенно так называемые частичные галлюцинации (похожие на эталонные данные, но всё же немного отличающиеся) (см. [это](#) и [это](#)).
- демонстрируют низкую или среднюю корреляцию с оценками людей по задачам [summarization](#) ([там тоже](#)), [faithfulness](#), и не постоянно коррелируют с человеческой оценкой в более широком контексте [набор задач](#).

Что насчёт моделей награждения?

Модели награждения обучаются предсказывать оценку на основе человеческих аннотаций для заданных пар промпт/заполнение. Итоговая цель — чтобы их предсказания соответствовали человеческим предпочтениям. После обучения эти модели могут использоваться для улучшения других моделей, выполняя роль функции вознаграждения, являющейся прокси для человеческой оценки.

Наиболее распространённым типом модели награждения является модель Брэдли-Терри, которая выдаёт один **парный счёт** согласно формуле:

$$p(\text{заполнение } b \text{ предпочтительнее заполнения } a) = \text{sigmoid(score}_b - \text{score}_a)$$

Эта модель обучается исключительно на парных сравнениях заполнений, которые проще собрать, чем оценки, но при этом может сравнивать несколько заполнений только для одного промпта, а не между разными промптами.

Другие модели расширили этот подход, чтобы прогнозировать более тонкую вероятность того, что одно заполнение лучше другого ([example](#)).

Это позволяет (теоретически) оценивать тонкие различия между заполнениями, но с ограничениями — становится сложно сохранять и сравнивать множество разных оценок между промптами в одном тестовом наборе. Кроме того, длина контекста и ограничения по памяти могут стать проблемой при сравнении слишком длинных заполнений.

Некоторые модели вознаграждения, такие как [SteerLM](#), выдают **абсолютные оценки**, которые можно использовать для прямой оценки сгенерированных ответов без необходимости парных сравнений. Эти модели могут быть проще в использовании для оценки, но для них также сложнее собирать данные, поскольку абсолютные оценки менее стабильны по сравнению с парными оценками в предпочтениях людей.

В последнее время были предложены модели, выдающие как абсолютные, так и относительные оценки, такие как [HelpSteer2-Preference](#) и [ArmoRM](#).

Как использовать модель вознаграждения для оценки?

Имея набор промптов, мы можем сгенерировать ответы с помощью языковой модели и попросить модель вознаграждения их оценить.

Для моделей, выдающих абсолютные оценки, полученные результаты можно усреднить, чтобы получить адекватную итоговую оценку.

Однако в более распространённом случае относительных оценок среднее вознаграждение может быть смещено выбросами (несколькими очень хорошими или очень плохими завершениями), поскольку разные промпты по своей природе могут иметь разные шкалы вознаграждений (некоторые промпты намного сложнее или проще других).

Вместо этого можно использовать

- **win rates:** взять эталонный набор ответов и вычислить процент ответов от модели, которые ранжируются выше эталонных. Это обеспечивает чуть более подробный анализ.
- **win probabilities:** средняя вероятность того, что ответы лучше эталонных, что даёт более тонкий и плавно изменяющийся сигнал.

Для относительных оценок не стоит просто усреднять сырье значения вознаграждения — выбросы и различающиеся шкалы сложности промптов будут искажать результаты. Вместо этого используйте долю побед или вероятность победы относительно некоторого эталона.

Преимущества и недостатки моделей вознаграждения

Модели вознаграждения, как правило, обладают следующими свойствами:

- **Очень быстрые:** получение оценки сводится к однократному прямому проходу относительно небольшой модели (поскольку мы получаем только оценку, а не длинный текст, в отличие от judge-LLMs).
- **Детерминированные:** при том же прямом проходе всегда воспроизводятся одинаковые оценки.
- **Вряд ли подвержены позиционному смещению:** поскольку большинство моделей обрабатывают только одно завершение, на них не влияет порядок. Для парных моделей позиционное смещение обычно также минимально, при условии, что данные обучения были сбалансированы относительно случаев, когда наилучший ответ занимал как первую, так и вторую позицию.
- **Не требуют настройки промптов:** модель просто выдаёт оценку на основе одного или двух завершений, в зависимости от данных предпочтений, на которых она была обучена.

С другой стороны, они:

- **Требуется специфическое дообучение:** это может быть достаточно затратным этапом, и хотя такие модели наследуют многие возможности базовой модели, они всё же могут плохо справляться с задачами, выходящими за рамки распределения обучения.
- **Потеря эффективности при использовании как в обучении с подкреплением, так и в оценке** (или при применении алгоритмов прямого выравнивания на наборах данных, схожих с обучающими данными модели награды), поскольку языковая модель может переобучаться под предпочтения модели награды.

Дальнейшие шаги

- Хорошее место для поиска моделей с высокой производительностью — это [RewardBench Leaderboard](#).
- Вы можете ознакомиться с применением моделей награды в статье [Nemotron](#).
- Для моделей награды, оценивающих отдельные промпты и ответы, можно кэшировать оценки многих эталонных моделей и легко видеть, как показывает себя новая модель.
- Отслеживание коэффициентов выигрыша или вероятностей во время обучения, например, [в этой статье](#), позволяет выявлять деградацию модели и выбирать оптимальные контрольные точки.

ОГРАНИЧЕНИЕ ВЫХОДНЫХ ДАННЫХ МОДЕЛИ

В ряде случаев нам может понадобиться, чтобы модель выдавала предсказание в строго определённом формате для упрощения оценки.

Использование промпта

Самый простой способ сделать это — добавить промпт задачи, который содержит очень конкретные инструкции о том, как модель должна отвечать

(`[Provide numerical answers in digits.]`, `[Use no abbreviation.]`, и т. д.).

Это не обязательно будет работать всегда, но должно быть достаточно эффективно для моделей с высокой способностью. Это тот подход, который мы использовали, например, в статье [GAIA](#).

Few shots и обучение в контексте

Следующий способ — ограничить модель с помощью так называемого «обучения в контексте». Путём предоставления примеров в промпте (что называется *few-shot prompting*), модель косвенно склоняется следовать повторяющейся структуре промпта для текущего примера.

Это метод, который в целом хорошо работал до конца 2023 года!

Однако широкое распространение методов *instruction-tuning* и добавление инструкционных данных на поздних этапах предварительного обучения моделей (непрерывное предварительное обучение) сместило более новые модели в сторону специфических форматов вывода (что [здесь](#) называется *Training on the test task*, или, как я бы сказал, *переобучение формату промпта*). Модели рассуждения также демонстрируют слабые результаты при использовании примеров с малым количеством выстрелов из-за следа рассуждений.

Этот метод также может быть ограничен для более старых моделей с меньшим размером контекста, так как некоторые примеры с малым количеством выстрелов не помещаются в окно контекста.

Структурированная генерация текста

Структурированная генерация текста ограничивает выходные данные, заставляя их следовать заданному пути, определённому грамматикой или регулярными выражениями, например. Библиотека `[outlines]` реализует это с помощью конечных автоматов, что весьма элегантно.

(Существуют и другие подходы, такие как чередующаяся генерация для создания json, но конечные автоматы — мои любимые).

Чтобы лучше понять, что происходит при использовании структурированной генерации, вы можете ознакомиться с [блогом](#), который мы написали вместе: структурированная генерация снижает вариативность промптов при оценке и делает результаты и ранжирование более стабильными. Вы также можете ознакомиться с общим [блогом](#) `[outlines]` для интересных реализаций и наблюдений, связанных с структурированной генерацией.

Однако некоторые недавние [research](#) показывают, что структурированная генерация может снижать производительность модели на некоторых задачах (напри-

мер, рассуждении), смещаая априорное вероятностное распределение слишком далеко от ожидаемого.

Далее

- ★ [Понимание работы конечного автомата при использовании структурированной генерации](#) на основе Outlines. Очень понятное руководство о том, как работает их метод!
- [Статья по методу Outlines](#) — более академическое объяснение упомянутого выше
- [Перемежающаяся генерация](#) — ещё один метод ограничения генераций для некоторых специфических форматов вывода

Забытые дети оценки

СТАТИСТИЧЕСКАЯ ЗНАЧИМОСТЬ

При публикации результатов оценки крайне важно приводить **доверительные интервалы** вместе с точечными оценками.

Доверительные интервалы, построенные на сырых результатах, можно получить из стандартных отклонений оценок или с помощью [бутстрэпинга](#) – для автоматических метрик это относительно просто, а для моделей-судей [последнее исследование](#) рекомендовало коррекцию смещения с использованием специальных оценивателей. Для оценки с участием людей необходимо сообщать показатели согласия.

Также эти показатели можно вычислять по вариациям промптов: задавая одни и те же вопросы немного по-разному или повторно используя одни и те же образцы с различными форматами промптов.

СТОИМОСТЬ И ЭФФЕКТИВНОСТЬ

При разработке и публикации результатов оценки следует начинать коллективно учитывать затраты на запуск модели! Модель рассуждения, требующая 10 минут размышлений и 10 000 токенов для ответа на 10 + 1 вопросов (потому что она решает сделать полный отступ о бинарной и десятичной арифметике), существенно менее эффективна, чем небольшая модель, отвечающая на 30 вопросов всего за несколько токенов.



Мы предлагаем вам предоставить следующую информацию:

- **Потребление токенов:** укажите общее количество выходных токенов, использованных во время оценки. Это особенно важно для оценки **эффективности** и влияет на стоимость проведения оценок модели в роли арбитра. Количество токенов напрямую влияет на **денежные затраты** и помогает другим оценить вычислительные ресурсы. Денежные затраты также могут служить хорошим приближением эффективности.
- **Время:** зафиксируйте время вывода, необходимое модели для завершения оценки. Это включает как фактическое время вывода, так и любые дополнительные задержки из-за ограничений API. Это особенно важно для приложений, чувствительных ко времени (например, некоторых инструментов-агентов, как в GAIA2).

Наконец, отчёт об экологическом следе используемых моделей становится всё более важным в свете общего состояния доступных на Земле ресурсов. Сюда входят выбросы углерода при обучении и потребление энергии при выводе, которые зависят от размера модели, аппаратного обеспечения (если оно известно) и количества сгенерированных токенов. Некоторые меньшие или квантизированные модели достигают очень интересного соотношения производительности и энергопотребления.

Эти метрики стоимости также могут иметь решающее значение при сравнении методов оценки. Например, использование мощной LLM в роли судьи может давать более качественный сигнал, чем автоматические метрики, но увеличение стоимости в 100 раз может быть неоправданным для некоторых сценариев. Аналогично, метрики, основанные на семплировании (`pass@k, maj@n`), увеличивают затраты пропорционально числу сэмплов, и это повышение стоимости нужно соотносить с улучшением качества сигнала, которое они дают.

Заключение

Оценка – это и искусство, и наука. Мы рассмотрели ландшафт оценки больших языковых моделей в 2025 году – от понимания причин оценки моделей и базовых механизмов токенизации и вывода до ориентирования в постоянно развивающейся экосистеме бенчмарков, а также создания собственных процедур оценки для ваших задач.

Ключевые моменты, которые, я надеюсь, вы запомните:

Критически оценивайте то, что вы измеряете. Оценки являются приближением к возможностям модели, поэтому высокий результат по бенчмарку не гарантирует реальную производительность. Разные подходы к оценке (автоматические метрики, судьи-человеки или модель-судьи) имеют свои предубеждения, ограничения и компромиссы.

Соотносите оценку с вашей целью. Проводите ли вы абляции во время обучения? Используйте быстрые, надёжные бенчмарки с сильным сигналом даже на небольших моделях. Сравниваете итоговые модели для выбора? Сосредоточьтесь на более сложных, неиспорченных наборах данных, проверяющих комплексные способности. Создаёте модель для конкретного варианта использования? Создавайте пользовательские оценки, которые отражают ваши задачи и данные.

Воспроизводимость требует внимательности к деталям. Незначительные различия в промптах, токенизации, нормализации, шаблонах или случайных сидах могут привести к изменению результатов на несколько пунктов. При публикации результатов будьте прозрачны в отношении вашей методологии. При попытках воспроизвести результаты ожидайте, что точное повторение будет чрезвычайно сложным, даже если вы пытаетесь контролировать все переменные.

Отдавайте предпочтение интерпретируемым методам оценки. По возможности выбирайте функциональное тестирование и проверяющие на основе правил вместо моделей-судей. Оценки, которые можно понять и отладить, обеспечивают более ясные и действенные выводы... чем более интерпретируема ваша оценка, тем лучше вы сможете улучшать свои модели!

Оценка — это процесс, который никогда не заканчивается. По мере улучшения моделей бенчмарки достигают насыщения. По мере роста обучающих данных вероятность их загрязнения увеличивается. С развитием вариантов использования необходимо измерять новые возможности. Оценка — это постоянная борьба!

В заключение: качество создаваемых нами моделей зависит от нашей способности измерять то, что важно. Спасибо за внимание!

БЛАГОДАРНОСТИ

Выражаем искреннюю благодарность всем, кто прямо или косвенно способствовал созданию этого документа, в особенности Хинек Кидличеку, Лубне Бен Аллал, Сандеру Лэнду и Натану Хабибу.

Для указания источника в академических публикациях, пожалуйста, цитируйте данную работу следующим образом:

Clémentine Fourrier, Thibaud Frere, Guilherme Penedo, Thomas Wolf (2025). «The LLM Evaluation Guidebook».

Цитирование в формате BibTeX

```
@misc{fourrier2025_the_llm_evaluation_guidebook,
    title={The LLM Evaluation Guidebook},
    author={Clémentine Fourrier and Thibaud Frere and
        Guilherme Penedo and Thomas Wolf},
    year={2025},
}
```