Потерянные в середине: как языковые модели используют длинные контексты (Lost in the Middle: How Language Models Use Long Contexts)

Nelson F. Liu[1*] Kevin Lin[2] John Hewitt[1] Ashwin Paranjape[3]
Michele Bevilacqua[3] Fabio Petroni[3] Percy Liang[1]
[1]Stanford University [2]University of California, Berkeley [3]Samaya AI
nfliu@cs.stanford.edu

Abstract

Хотя современные языковые модели могут принимать длинные контексты в качестве входных данных, относительно мало известно о том, насколько хорошо они используют более длинные контексты. Мы анализируем производительность языковых моделей в двух задачах, требующих идентификации релевантной информации в их входных контекстах: многодокументный вопросно-ответный анализ и извлечение ключевых значений. Мы обнаруживаем, что производительность может значительно ухудшаться при изменении позиции релевантной информации, что указывает на то, что текущие языковые модели не могут надежно использовать информацию в длинных входных контекстах. В частности, мы наблюдаем, что производительность часто максимальна, когда релевантная информация находится в начале или в конце входного контекста, и значительно ухудшается, когда модели должны получать доступ к релевантной информации в середине длинных контекстов, даже для моделей с явно длинным контекстом. Наш анализ дает лучшее понимание того, как языковые модели используют свой входной контекст, и предлагает новые протоколы оценки для будущих моделей с длинным контекстом.

1 Введение

Языковые модели стали важным и гибким строительным блоком в различных языковых технологиях, ориентированных на пользователя, включая разговорные интерфейсы, поиск и суммаризацию, а также совместное написание (Shuster et al., 2022; Thoppilan et al., 2022; Lee et al., 2022, inter alia). Эти модели выполняют задачи нижнего уровня в основном через подсказки: вся релевантная спецификация задачи и данные для обработки форматируются как текстовый входной контекст, и модель возвращает сгенерированное текстовое завершение. Эти входные контексты могут содержать тысячи токенов, особенно когда языковые модели используются для обработки длинных документов (например, юридических или научных документов, истории разговоров и т. д.) или

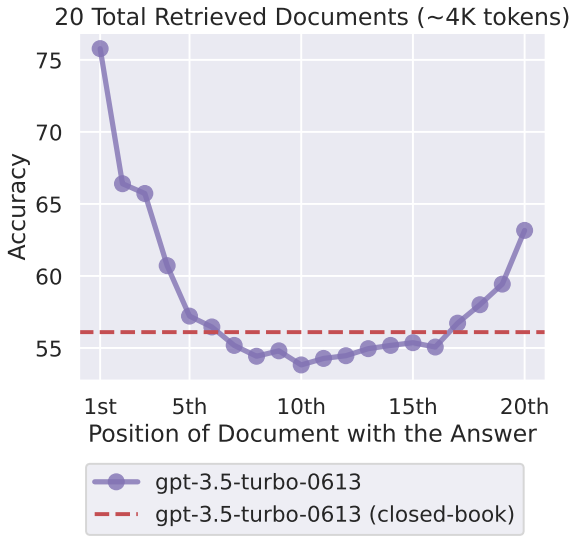

Рис. 1: Изменение местоположения релевантной информации (в данном случае, позиции отрывка, который отвечает на входной вопрос) в контексте входных данных языковой модели приводит к U-образной кривой производительности — модели лучше используют релевантную информацию, которая находится в самом начале (эффект первичности) или в конце её входного контекста (эффект недавности), а производительность значительно ухудшается, когда модели должны получать доступ и использовать информацию, расположенную в середине её входного контекста.

*Работа частично выполнена в качестве стажера в Samaya AI.

когда языковые модели дополняются внешней информацией (например, релевантными документами из поисковой системы, результатами запросов к базе данных и т. д.; Petroni et al., 2020; Ram et al., 2023; Shi et al., 2023; Mallen et al., 2023; Schick et al., 2023, inter alia).

Обработка этих случаев использования требует, чтобы языковые модели успешно работали с длинными последовательностями. Существующие языковые модели обычно реализуются с помощью трансформеров (Vaswani et al., 2017), которые требуют памяти и вычислений, увеличивающихся квадратично в зависимости от длины последовательности. В результате трансформерные языковые модели часто обучались с относительно небольшими оконными контекстами (от 512 до 2048 токенов). Недавние улучшения в аппаратном обеспечении (например, более быстрые графические процессоры с большим объемом памяти) и алгоритмах (Dai et al., 2019; Dao et al., 2022; Poli et al., 2023; Rubin and Berant, 2023, inter alia) привели к появлению языковых моделей с большими оконными контекстами (например, 4096, 32K и даже 100K токенов), но остается неясным, как эти модели с расширенным контекстом используют свои входные контексты при выполнении задач нижнего уровня.

Мы эмпирически исследуем этот вопрос с помощью контролируемых экспериментов с различными современными открытыми (MPT-30B-Instruct, LongChat-13B (16K)) и закрытыми (OpenAI's GPT-3.5-Turbo и Anthropic's Claude-1.3) языковыми моделями в условиях, требующих доступа и использования информации в пределах входного контекста. В частности, в наших экспериментах вносятся контролируемые изменения в размер входного контекста и положение релевантной информации в пределах входного контекста, и изучаются их эффекты на производительность языковой модели. Если языковые модели могут надежно использовать информацию в пределах длинных входных контекстов, то их производительность должна быть минимально подвержена влиянию положения релевантной информации в контексте входных данных.

Сначала мы экспериментируем с многодокументным вопросно-ответным анализом, который требует от моделей анализа предоставленных документов для нахождения релевантной информации и использования её для ответа на заданный вопрос; эта задача имитирует настройку генерации с дополнением поиска, лежащую в основе многих коммерческих приложений генеративного поиска и вопросно-ответного анализа (например, Bing Chat). В этом контексте мы контролируем (i) длину входного контекста, изменяя количество документов в контексте входных данных (аналогично извлечению большего или меньшего количества документов в генерации с дополнением поиска), и (ii) контролируем положение релевантной информации в пределах входного контекста, изменяя порядок документов, чтобы разместить релевантный документ в начале, середине или конце контекста.

Мы обнаруживаем, что изменение положения релевантной информации в контексте входных данных может существенно повлиять на производительность модели, что указывает на то, что текущие языковые модели не могут надежно получать доступ и использовать информацию в длинных входных контекстах. Более того, мы наблюдаем характерную U-образную кривую производительности (Рисунок 1); производительность языковой модели наивысшая, когда релевантная информация находится в самом начале (эффект первичности) или в конце её входного контекста (эффект недавности), и производительность значительно ухудшается, когда модели должны получать доступ и использовать информацию в середине своих входных контекстов (§2.3). Например, когда релевантная информация размещена в середине её входного контекста, производительность GPT-3.5-Turbo на задаче многодокументного вопросно-ответного анализа ниже, чем её производительность при прогнозировании без каких-либо документов (т. е. в закрытой книге; 56.1%). Кроме того, мы обнаруживаем, что модели часто имеют идентичную производительность с их аналогами с расширенным контекстом, что указывает на то, что модели с расширенным контекстом не обязательно лучше используют свой входной контекст (§2.3).

Учитывая, что языковые модели испытывают трудности с извлечением и использованием релевантной информации в задаче многодокументного вопросно-ответного анализа, в какой степени языковые модели вообще могут извлекать из своих входных контекстов? Мы изучаем этот вопрос с помощью синтетической задачи извлечения ключевых значений, которая предназначена для минимального тестирования базовой способности извлекать совпадающие токены из входного контекста. В этой задаче моделям предоставляется коллекция пар ключ-значение в формате JSON, и они должны вернуть значение, связанное с определенным ключом. Подобно задаче многодокументного вопросно-ответного анализа, задача извлечения ключевых значений допускает контролируемые изменения длины входного контекста (добавление большего количества пар ключ-значение) и положения релевантной информации. Хотя некоторые модели выполняют синтетическую задачу извлечения ключевых значений идеально, другие модели испытывают трудности даже с простым извлечением совпадающих токенов, которые встречаются в середине их входного контекста, и продолжают демонстрировать U-образную кривую производительности.

Чтобы лучше понять, почему языковые модели испытывают трудности с надежным доступом и использованием информации в своих входных контекстах, мы изучаем роль архитектуры модели (только декодер против кодер-декодер), контекстуализации с учетом запроса и тонкой настройки инструкций (§4). Мы обнаруживаем, что:

- Кодер-декодер модели относительно устойчивы к изменениям положения релевантной информации в их входном контексте, но только при оценке последовательностей в пределах их максимальной длины последовательности на этапе обучения. При оценке последовательностей, превышающих те, что были видны во время обучения, мы наблюдаем U-образную кривую производительности (§4.1).
- Контекстуализация с учетом запроса (размещение запроса перед и после документов или пар ключ-значение) обеспечивает почти идеальную производительность в синтетической задаче извлечения ключевых значений, но минимально изменяет тенденции в многодокументном вопросно-ответном анализе (§4.2).
- Даже базовые языковые модели (т. е. без тонкой настройки инструкций) демонстрируют U-образную кривую производительности при изменении положения релевантной информации в контексте входных данных.

Наши результаты показывают, что предоставление языковым моделям более длинных входных контекстов — это компромисс: предоставление языковой модели большего объема информации может помочь ей выполнить задачу нижнего уровня, но также увеличивает объем контента, который модель должна анализировать, что может снизить точность. Чтобы лучше понять этот компромисс на практике, мы проводим тематическое исследование с моделями извлечения-читателя на открытом вопросно-ответном анализе (§5). В отличие от нашей контролируемой задачи многодокументного вопросно-ответного анализа, где контекст всегда содержит ровно один документ, который отвечает на вопрос, ни один или многие из топ k документов могут не содержать ответа в настройке открытого вопросно-ответного анализа. Когда мы извлекаем из Википедии, чтобы ответить на запросы из NaturalQuestions-Open, мы обнаруживаем, что производительность модели насыщается задолго до насыщения извлечения, что указывает на то, что текущие модели не могут эффективно использовать дополнительные извлеченные документы — использование 50 документов вместо 20 извлеченных документов лишь незначительно улучшает производительность ($\sim$1.5% для GPT-3.5-Turbo и $\sim$1% для claude-1.3).

Наш анализ дает лучшее понимание того, как языковые модели используют свой входной контекст, и вводит новые протоколы оценки для будущих моделей с длинным контекстом; чтобы утверждать, что языковая модель может надежно использовать информацию в пределах длинных входных контекстов, необходимо показать, что её производительность минимально подвержена влиянию положения релевантной информации в

контексте входных данных (например, минимальная разница в наилучшей и наихудшей производительности). Чтобы способствовать дальнейшей работе по пониманию и улучшению того, как языковые модели используют свой входной контекст, мы выпускаем наш код и данные оценки.[1]

2 Многодокументный вопросно-ответный анализ

Наша цель — лучше понять, как языковые модели используют свой входной контекст. Для этого мы анализируем производительность моделей в многодокументном вопросно-ответном анализе, который требует от моделей нахождения релевантной информации в пределах входного контекста и использования её для ответа на вопрос. В частности, мы вносим контролируемые изменения в длину входного контекста и положение релевантной информации и измеряем изменения в производительности задачи.

2.1 Экспериментальная установка

В задаче многодокументного вопросно-ответного анализа входные данные модели включают (i) вопрос, на который нужно ответить, и (ii) k документов (например, отрывки из Википедии), где ровно один из документов содержит ответ на вопрос, а $k-1$ «отвлекающих» документов не содержат. Эта задача требует от модели доступа к документу, содержащему ответ, в пределах её входного контекста и использования его для ответа на вопрос. Рисунок 2 представляет пример.

Мы реализуем эту задачу с данными из NaturalQuestions-Open (Lee et al., 2019; Kwiatkowski et al., 2019), которые содержат исторические запросы, отправленные в поисковую систему Google, в сочетании с аннотированными людьми ответами, извлеченными из Википедии. В частности, мы берем 2655 запросов, где аннотированный длинный ответ является абзацем (в отличие от списка или таблицы). Мы используем отрывки (кусочки не более 100 токенов) из Википедии в качестве документов в пределах наших входных контекстов. Для каждого из запросов нам нужен документ, содержащий ответ, и $k-1$ отвлекающих документов, которые не содержат ответа. Чтобы получить документ, который отвечает на вопрос, мы используем абзац Википедии, содержащий ответ из аннотаций NaturalQuestions.

Чтобы собрать $k-1$ отвлекающих документов, которые не содержат ответа, мы используем систему извлечения (Contriever, дообученную на MS-MARCO; Izacard et al., 2021) для извлечения $k-1$ отрывков из Википедии, которые наиболее релевантны запросу и не содержат ни одного из аннотированных ответов NaturalQuestions.[2,3] В входном контексте отвлекающие документы представлены в порядке убывания релевантности.[4]

Чтобы модулировать положение релевантной информации в пределах входного контекста, мы изменяем порядок документов, чтобы изменить положение документа, содержащего ответ (Рисунок 3). Чтобы модулировать длину входного контекста в этой задаче, мы увеличиваем или уменьшаем количество извлеченных документов, не содержащих ответа (Рисунок 4).

Следуя Kandpal et al. (2022) и Mallen et al. (2023), мы используем точность в качестве нашего основного показателя оценки, оценивая, появляется ли какой-либо из правильных ответов (как взято из аннотаций NaturalQuestions) в предсказанном выводе.

Наша экспериментальная установка аналогична эксперим

ентам с иголкой в стоге сена Ivgi et al. (2023), которые сравнивают производительность вопросно-ответного анализа, когда релевантный абзац размещен (i) в начале входного контекста или (ii) в случайной позиции в пределах входного контекста. Они обнару-

[1] nelsonliu.me/papers/lost-in-the-middle

[2] Неопределенность в NaturalQuestions-Open означает, что небольшое количество отвлекающих отрывков может содержать разумный ответ. Мы также проводим эксперименты на подмножестве однозначных вопросов, находя аналогичные результаты и выводы; см. Приложение A.

[3] Мы также изучали использование случайных документов в качестве отвлекающих, см. Приложение B для получения дополнительной информации.

[4] Поскольку может быть предварительное предположение о том, что «результаты поиска» появляются в ранжированном порядке, мы изучили случайное упорядочивание $k-1$ отвлекающих документов и упоминание о том, что документы случайно упорядочены в описании задачи, но обнаружили те же тенденции. См. Приложение C для получения дополнительной информации.

Input Context

```
Write a high-quality answer for the given question using only the provided search
results (some of which might be irrelevant).

Document [1](Title: Asian Americans in science and technology) Prize in physics for
discovery of the subatomic particle J/ψ. Subrahmanyan Chandrasekhar shared...
Document [2](Title: List of Nobel laureates in Physics) The first Nobel Prize in
Physics was awarded in 1901 to Wilhelm Conrad Röntgen, of Germany, who received...
Document [3](Title: Scientist) and pursued through a unique method, was essentially
in place. Ramón y Cajal won the Nobel Prize in 1906 for his remarkable...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Рис. 2: Пример задачи многодокументного вопросно-ответного анализа с входным контекстом и желаемым ответом модели. Документ, содержащий ответ, выделен в входном контексте здесь для ясности.

Input Context

```
Write a high-quality answer for the given question
using only the provided search results (some of
which might be irrelevant).

Document [1](Title: List of Nobel laureates in
Physics) ...
Document [2](Title: Asian Americans in science and
technology) ...
Document [3](Title: Scientist) ...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Рис. 3: Модуляция положения релевантной информации в пределах входного контекста для примера многодокументного вопросно-ответного анализа, представленного на Рисунке 2. Изменение порядка документов во входном контексте не влияет на желаемый вывод.

Input Context

```
Write a high-quality answer for the given question
using only the provided search results (some of
which might be irrelevant).

Document [1](Title: Asian Americans in science and
technology) ...
Document [2](Title: List of Nobel laureates in
Physics) ...
Document [3](Title: Scientist) ...
Document [4](Title: Norwegian Americans) ...
Document [5](Title: Maria Goeppert Mayer) ...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Рис. 4: Модуляция длины входного контекста примера многодокументного вопросно-ответного анализа, представленного на Рисунке 2. Добавление документов, не содержащих ответа, увеличивает длину входного контекста, но не влияет на желаемый вывод.

живают, что кодер-декодер модели имеют значительно более высокую производительность, когда релевантная информация размещена в начале входного контекста. В отличие от этого, мы изучаем более тонкие изменения в положении релевантной информации.

2.2 Модели

Мы анализируем несколько современных открытых и закрытых языковых моделей. Мы используем жадное декодирование при генерации выводов и оставляем изучение других методов декодирования для будущей работы. Мы используем стандартный набор подсказок для каждой модели (Рисунок 2).

Открытые модели. Мы экспериментируем с MPT-30B-Instruct, который имеет максимальную длину контекста 8192 токена. Модель была первоначально предварительно обучена на 1 триллион токенов с использованием последовательностей длиной 2048 токенов, после чего последовала дополнительная фаза предварительного обучения с адаптацией длины последовательности на 50 миллиардов токенов с использованием последовательностей длиной 8192 токена. MPT-30B-Instruct использует ALiBi (Press et al., 2022) для представления позиционной информации. Мы также оцениваем LongChat-13B (16K) (Li et al., 2023), который расширяет окно контекста LLaMA-13B (Touvron et al., 2023a) с 2048 до 16384 токенов, используя сжатые вращающиеся позиционные

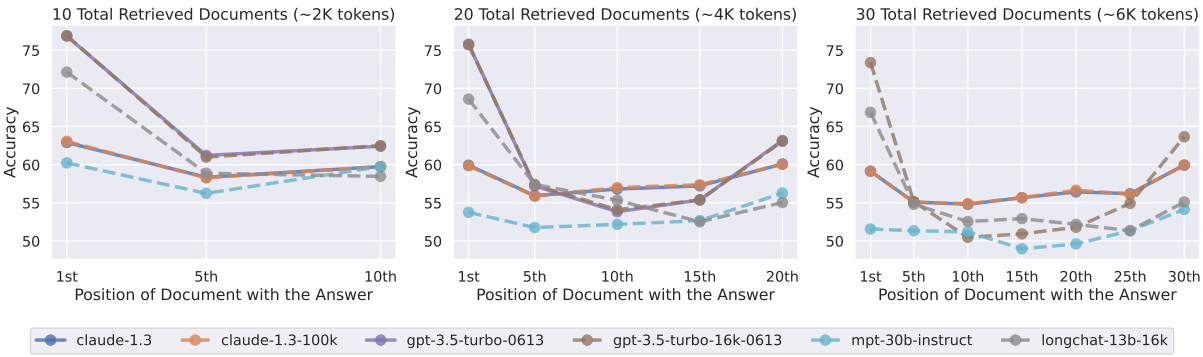

Рис. 5: Влияние изменения положения релевантной информации (документа, содержащего ответ) на производительность многодокументного вопросно-ответного анализа. Более низкие позиции ближе к началу входного контекста. Производительность наивысшая, когда релевантная информация находится в самом начале или в конце контекста, и быстро ухудшается, когда модели должны анализировать информацию в середине своего входного контекста.

встраивания перед дообучением с последовательностями длиной 16384 токена.

Закрытые модели. Мы используем API OpenAI для экспериментов с GPT-3.5-Turbo и GPT-3.5-Turbo (16K).[5] GPT-3.5-Turbo имеет максимальную длину контекста 4K токенов, а GPT-3.5-Turbo (16K) — версия с расширенной максимальной длиной контекста 16K токенов. Мы оцениваем Claude-1.3 и Claude-1.3 (100K) с помощью API Anthropic; Claude-1.3 имеет максимальную длину контекста 8K токенов, а Claude-1.3 (100K) имеет расширенную длину контекста 100K токенов.[6]

2.3 Результаты и обсуждение

Мы экспериментируем с входными контекстами, содержащими 10, 20 и 30 всего документов. Рисунок 5 представляет производительность многодокументного вопросно-ответного анализа при изменении положения релевантной информации в пределах входного контекста. Чтобы контекстуализировать производительность модели, мы также оцениваем в закрытой книге и оракульских настройках (Таблица 1). В закрытой книге модели не получают никаких документов в их входном контексте и должны полагаться на свою параметрическую память для генерации правильного ответа. С другой стороны, в оракульской настройке языковым моделям предоставляется единственный документ, содержащий ответ, и они должны использовать его для ответа на вопрос.

Модель	Закрытая книга	Оракул
LongChat-13B (16K)	35.0%	83.4%
MPT-30B-Instruct	31.5%	81.9%
GPT-3.5-Turbo	56.1%	88.3%
GPT-3.5-Turbo (16K)	56.0%	88.6%
Claude-1.3	48.3%	76.1%
Claude-1.3 (100K)	48.2%	76.4%

Таблица 1: Точность языковых моделей в закрытой книге и оракульской настройке на задаче многодокументного вопросно-ответного анализа.

Производительность модели наивысшая, когда релевантная информация находится в начале или в конце её входного контекста. Как показано на Рисунке 5, изменение положения релевантной информации в контексте входных данных приводит к значительному снижению производительности модели. В частности, мы видим характерную U-образную кривую производительности — модели часто лучше используют релевантную информацию, которая находится в самом начале (эффект первичности) и в самом конце контекста (эффект недавности), и испытывают ухудшение производительности, когда

[5] Мы используем версии модели OpenAI 0613.

[6] Мы также оцениваем GPT-4 (8K) на подмножестве экспериментов с многодокументным вопросно-ответным анализом, находя аналогичные результаты и тенденции, как и у других моделей (хотя GPT-4 имеет более высокую абсолютную производительность). Оценка GPT-4 на полных экспериментах с многодокументным вопросно-ответным анализом и извлечением ключевых значений обойдется в более чем $6000. См. Приложение D для результатов и обсуждения GPT-4.

вынуждены использовать информацию в середине своего входного контекста. Например, производительность GPT-3.5-Turbo на многодокументном вопросно-ответном анализе может упасть более чем на 20% — в худшем случае производительность в условиях с 20 и 30 документами ниже, чем производительность без каких-либо входных документов (т. е. производительность в закрытой книге; 56.1%). Эти результаты указывают на то, что текущие модели не могут эффективно анализировать весь свой оконный контекст при подсказках для задач нижнего уровня.

Модели с расширенным контекстом не обязательно лучше используют входной контекст. Когда входной контекст помещается в оконный контекст как модели, так и её аналога с расширенным контекстом, мы видим, что производительность между ними почти идентична. Например, условия с 10 и 20 документами помещаются в оконный контекст GPT-3.5-Turbo и GPT-3.5-Turbo (16K), и мы наблюдаем, что их производительность в зависимости от положения относительной информации почти совпадает (сплошная фиолетовая и пунктирная коричневая серии на Рисунке 5). Эти результаты указывают на то, что модели с расширенным контекстом не обязательно лучше своих нерасширенных аналогов в использовании своего входного контекста.

3 Насколько хорошо языковые модели могут извлекать из входных контекстов?

Учитывая, что языковые модели испытывают трудности с извлечением и использованием информации из середины их входных контекстов в задаче многодокументного вопросно-ответного анализа, в какой степени они могут просто извлекать из входных контекстов? Мы изучаем этот вопрос с помощью синтетической задачи извлечения ключевых значений, которая предназначена для предоставления минимального теста базовой способности извлекать совпадающие токены из входного контекста.

3.1 Экспериментальная установка

В нашей синтетической задаче извлечения ключевых значений входные данные включают (i) строковый сериализованный объект JSON с k парами ключ-значение, где каждый из ключей и значений является уникальными, случайно сгенерированными UUID, и (ii) ключ в вышеупомянутом объекте JSON. Цель — вернуть значение, связанное с указанным ключом. Таким образом, каждый объект JSON содержит одну релевантную пару ключ-значение (где значение должно быть возвращено) и $k-1$ нерелевантных «отвлекающих» пар ключ-значение. Рисунок 6 предоставляет пример входного контекста и его соответствующего желаемого вывода. Мы снова измеряем точность, оценивая, появляется ли правильное значение в предсказанном выводе.

Наша синтетическая задача извлечения ключевых значений имеет аналогичные цели с тестом малого извлечения Papailiopoulos et al. (2023) и задачей извлечения строк Li et al. (2023), но мы явно стремимся дистиллировать и упростить задачу, удаляя как можно больше семантики естественного языка (используя случайные UUID вместо этого), поскольку языковые особенности могут представлять потенциальные искажения. Например, трансформерные языковые модели могут иметь различную чувствительность к различным лингвистическим особенностям в их входных данных (O'Connor and Andreas, 2021).

Чтобы модулировать положение релевантной информации в пределах входного контекста, мы изменяем положение ключа для извлечения в сериализованном объекте JSON. Чтобы модулировать длину входного контекста, мы изменяем количество входных пар ключ-значение JSON k, добавляя или удаляя случайные ключи, изменяя количество отвлекающих пар ключ-значение.

3.2 Результаты и обсуждение

Мы экспериментируем с входными контекстами, содержащими 75, 140 и 300 пар ключ-значение (500 примеров каждый). Мы используем тот же набор моделей, что и в экспериментах с многодокументным вопросно-ответным анализом, см. §2.2 для получения дополнительной информации.

Рисунок 7 представляет производительность извлечения ключевых значений. Claude-1.3 и Claude-1.3 (100K) почти идеаль-

Input Context

```
Extract the value corresponding to the specified key in the JSON object below.

JSON data:
{"2a8d601d-1d69-4e64-9f90-8ad825a74195": "bb3ba2a5-7de8-434b-a86e-a88bb9fa7289",
 "a54e2eed-e625-4570-9f74-3624e77d6684": "d1ff29be-4e2a-4208-a182-0cea716be3d4",
 "9f4a92b9-5f69-4725-ba1e-403f08dea695": "703a7ce5-f17f-4e6d-b895-5836ba5ec71c",
 "52a9c80c-da51-4fc9-bf70-4a4901bc2ac3": "b2f8ea3d-4b1b-49e0-a141-b9823991ebeb",
 "f4eb1c53-af0a-4dc4-a3a5-c2d50851a178": "d733b0d2-6af3-44e1-8592-e5637fdb76fb"}

Key: "9f4a92b9-5f69-4725-ba1e-403f08dea695"
Corresponding value:
```

Desired Output

```
703a7ce5-f17f-4e6d-b895-5836ba5ec71c
```

Рис. 6: Пример задачи извлечения ключевых значений с входным контекстом и желаемым выводом модели. Учитывая ключ, цель — вернуть связанное значение. Все ключи и значения — это UUID длиной 128 бит. Релевантная пара ключ-значение для ответа на запрос выделена здесь в пределах входного контекста для ясности.

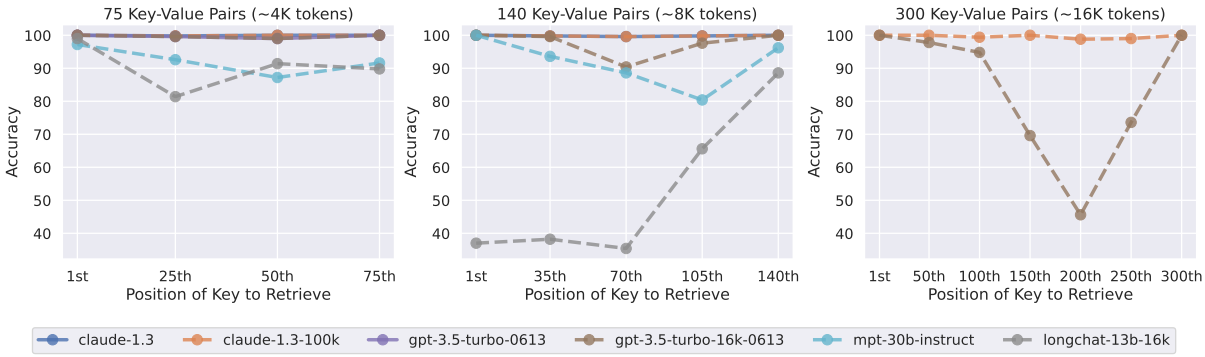

Рис. 7: Влияние изменения длины входного контекста и положения релевантной информации на производительность извлечения ключевых значений. Более низкие позиции ближе к началу входного контекста. Хотя некоторые модели показывают идеальную точность в этой синтетической задаче (например, Claude-1.3 и Claude-1.3 (100K)), мы снова видим, что производительность часто наивысшая, когда релевантная информация находится в самом начале или в конце контекста, и быстро ухудшается, когда модели должны извлекать из середины входного контекста.

но выполняют все оцененные длины входных контекстов, но другие модели испытывают трудности, особенно когда контексты содержат 140 или 300 пар ключ-значение — хотя синтетическая задача извлечения ключевых значений требует только идентификации точного совпадения в пределах входного контекста, не все модели достигают высокой производительности.

Подобно нашим результатам многодокументного вопросно-ответного анализа, GPT-3.5-Turbo, GPT-3.5-Turbo (16K) и MPT-30B-Instruct имеют наименьшую производительность, когда они должны получать доступ к парам ключ-значение в середине их входного контекста. LongChat-13B (16K) демонстрирует другую тенденцию в настройке с 140 парами ключ-значение; мы качественно наблюдаем, что когда релевантная информация размещена в начале входного контекста, LongChat-13B (16K) склонен генерировать код для извлечения ключа, а не выводить значение напрямую.

4 Почему языковые модели не устойчивы к изменениям положения релевантной информации?

Наши результаты многодокументного вопросно-ответного анализа и извлечения ключевых значений показывают, что языковые модели испытывают трудности с надежным доступом и использованием информации в длинных входных контекстах,

поскольку производительность значительно ухудшается при изменении положения релевантной информации. Чтобы лучше понять, почему, мы проводим предварительные исследования роли архитектуры модели (только декодер против кодер-декодер), контекстуализации с учетом запроса и тонкой настройки инструкций.

4.1 Влияние архитектуры модели

Открытые модели, которые мы оценивали, все являются моделями только декодера — на каждом временном шаге они могут только обращаться к предыдущим токенам. Чтобы лучше понять потенциальные эффекты архитектуры модели на то, как языковые модели используют контекст, мы сравниваем модели только декодера и кодер-декодер.

Мы экспериментируем с Flan-T5-XXL (Raffel et al., 2020; Chung et al., 2022) и Flan-UL2 (Tay et al., 2023). Flan-T5-XXL обучен с последовательностями длиной 512 токенов (кодер и декодер). Flan-UL2 изначально обучен с последовательностями длиной 512 токенов (кодер и декодер), но затем предварительно обучен ещё на 100K шагов с 1024 токенами (кодер и декодер) перед тонкой настройкой инструкций на последовательностях с 2048 токенами в кодере и 512 токенами в декодере. Однако, поскольку эти модели используют относительные позиционные встраивания, они могут (в принципе) экстраполировать за пределы этих максимальных длин контекста; Shaham et al. (2023) обнаруживают, что обе модели могут хорошо работать с последовательностями длиной до 8K токенов.

Рисунок 8 сравнивает производительность моделей только декодера и кодер-декодер. Когда Flan-UL2 оценивается на последовательностях в пределах её 2048-токенного оконного контекста на этапе обучения (Рисунок 8; левый подграфик), её производительность относительно устойчива к изменениям положения релевантной информации в пределах входного контекста (1.9% абсолютная разница между наилучшей и наихудшей производительностью). Когда оценивается на настройках с последовательностями длиннее 2048 токенов (Рисунок 8; центральный и правый), производительность Flan-UL2 начинает ухудшаться, когда релевантная информация размещена в середине. Flan-T5-XXL показывает аналогичную тенденцию, где более длинные входные контексты приводят к большему ухудшению производительности при размещении релевантной информации в середине входного контекста. Мы предполагаем, что кодер-декодер модели могут лучше использовать свои оконные контексты, потому что их двунаправленный кодер позволяет обрабатывать каждый документ в контексте будущих документов, что потенциально улучшает оценку относительной важности между документами.

4.2 Влияние контекстуализации с учетом запроса

Наши эксперименты с многодокументным вопросно-ответным анализом и извлечением ключевых значений размещают запрос (т. е. вопрос для ответа или ключ для извлечения) после данных для обработки (т. е. документов или пар ключ-значение). В результате модели только декодера не могут обращаться к токенам запроса при контекстуализации документов или пар ключ-значение, поскольку запрос появляется только в конце подсказки, и модели только декодера могут обращаться только к предыдущим токенам на каждом временном шаге. В отличие от этого, кодер-декодер модели (которые кажутся более устойчивыми к изменениям положения релевантной информации; §4.1) используют двунаправленный кодер для контекстуализации входных контекстов — можем ли мы использовать это наблюдение для улучшения моделей только декодера, размещая запрос перед и после данных, позволяя контекстуализацию документов (или пар ключ-значение) с учетом запроса?

Мы обнаруживаем, что контекстуализация с учетом запроса значительно улучшает производительность в задаче извлечения ключевых значений — все модели достигают почти идеальной производительности в настройках с 75, 140 и 300 парами ключ-значение. Например, GPT-3.5-Turbo (16K) с контекстуализацией с учетом запроса достигает идеальной производительности при оценке с 300 парами ключ-значение.

В отличие от значительного влияния на производительность извлечения ключевых значений, контекстуализация с учетом запро-

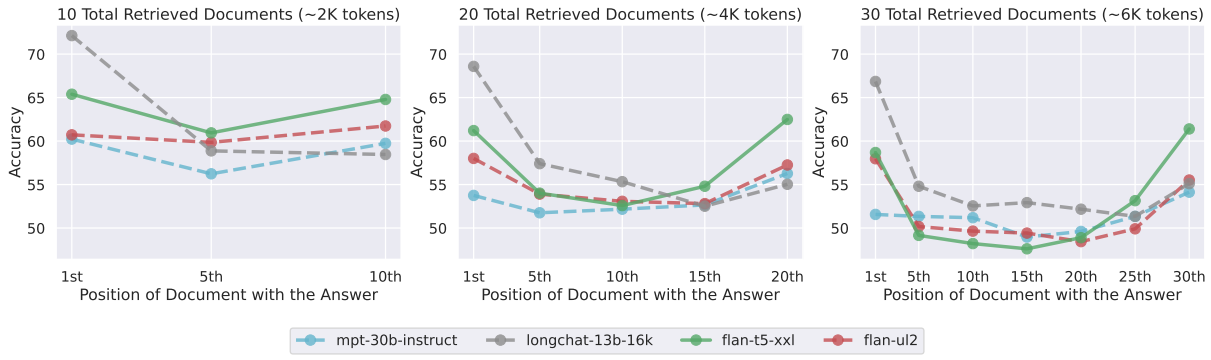

Рис. 8: Когда кодер-декодер модели (Flan-UL2 и Flan-T5-XXL) оцениваются на последовательностях, которые короче их максимальной длины последовательности на этапе обучения кодера (2048 и 512 токенов соответственно), они относительно устойчивы к изменениям положения релевантной информации в их входном контексте (левый подграфик). В отличие от этого, когда эти модели оцениваются на последовательностях длиннее тех, что были видны во время обучения (центральный и правый подграфики), мы наблюдаем U-образную кривую производительности — производительность выше, когда релевантная информация находится в начале или в конце входного контекста, в отличие от середины входного контекста.

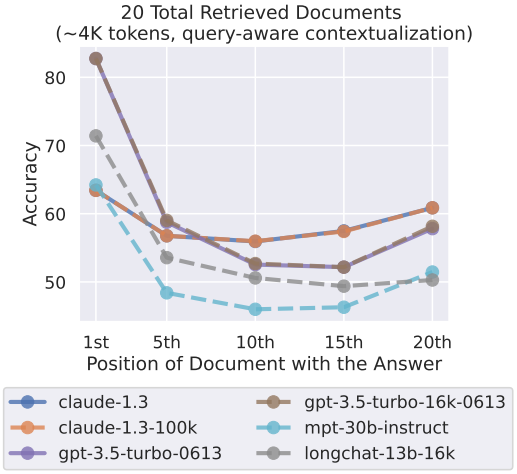

Рис. 9: Контекстуализация с учетом запроса (размещение запроса перед и после документов) не существенно улучшает устойчивость языковых моделей к изменению положения релевантной информации в многодокументном вопросно-ответном анализе; производительность слегка увеличивается, когда релевантная информация находится в самом начале, но в остальном слегка уменьшается.

са минимально влияет на тенденции производительности в задаче многодокументного вопросно-ответного анализа (Рисунок 9); она слегка улучшает производительность, когда релевантная информация находится в самом начале входного контекста, но слегка уменьшает производительность в других настройках.

4.3 Влияние тонкой настройки инструкций

Модели, которые мы оценивали, все прошли тонкую настройку инструкций — после их начального предварительного обучения они проходят супервизорное дообучение на наборе данных инструкций и ответов. Спецификация задачи и/или инструкция обычно размещаются в начале входного контекста в данных супервизорного дообучения инструкций, что может привести к тому, что языковые модели, прошедшие тонкую настройку инструкций, будут уделять больше внимания началу входного контекста. Чтобы лучше понять потенциальные эффекты тонкой настройки инструкций на то, как языковые модели используют длинные входные контексты, мы сравниваем производительность многодокументного вопросно-ответного анализа MPT-30B-Instruct с её базовой моделью (т. е. до тонкой настройки инструкций) MPT-30B. Мы используем ту же экспериментальную установку, что и в §2.

Рисунок 10 сравнивает производительность многодокументного вопросно-ответного анализа MPT-30B и MPT-30B-Instruct в зависимости от положения релевантной информации в входном контексте. Удивительно, но мы видим, что и MPT-30B, и MPT-30B-Instruct демонстрируют U-образную кривую производительности, где производительность наивысшая, когда релевантная информация находится в самом

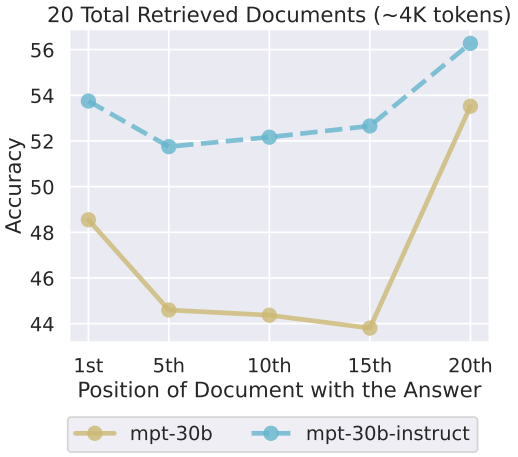

Рис. 10: Производительность многодокументного вопросно-ответного анализа MPT-30B-Instruct по сравнению с её базовой моделью (т. е. до тонкой настройки инструкций) MPT-30B. Обе модели имеют U-образную кривую производительности, где производительность значительно выше, когда релевантная информация находится в начале или в конце входного контекста, что указывает на то, что сам процесс тонкой настройки инструкций не обязательно ответственен за эти тенденции производительности.

начале или в самом конце контекста. Хотя абсолютная производительность MPT-30B-Instruct равномерно выше, чем у MPT-30B, их общие тенденции производительности схожи. Мы также наблюдаем, что тонкая настройка инструкций слегка уменьшает разницу в наихудшей производительности с почти 10% между наилучшей и наихудшей производительностью базовой модели до около 4%.

Эти наблюдения дополняют предыдущие работы, которые обнаружили, что не прошедшие тонкую настройку инструкций языковые модели склонны к недавним токенам (т. е. к концу входного контекста; Khandelwal et al., 2018; Press et al., 2021). Эта склонность к недавности была замечена в предыдущих работах при оценке моделей на предсказание следующего слова в непрерывном тексте, настройке, где языковые модели минимально выигрывают от долгосрочной информации (Sun et al., 2021). В отличие от этого, наши результаты показывают, что языковые модели способны использовать более долгосрочную информацию (т. е. начало входного контекста), когда они подсказаны данными в формате инструкций. Мы предполагаем, что не прошедшие тонкую настройку инструкций языковые модели учатся использовать эти длинные контексты из аналогично форматированных данных, которые могут встречаться в текстах Интернета, виденных во время предварительного обучения, например, вопросы и ответы на StackOverflow.

Чтобы лучше понять влияние дополнительного дообучения и масштаба модели, мы также экспериментировали с моделями Llama-2 различных размеров (7B, 13B и 70B) с и без дополнительного супервизорного дообучения и обучения с подкреплением от человеческой обратной связи (Приложение E). Мы обнаруживаем, что U-образная кривая производительности появляется только в достаточно больших языковых моделях (с или без дополнительного дообучения) — модели Llama-2 7B исключительно склонны к недавности, в то время как модели 13B и 70B демонстрируют U-образную кривую производительности. Кроме того, мы видим, что процедура супервизорного дообучения и обучения с подкреплением от человеческой обратной связи Llama-2 слегка смягчает позиционное искажение в меньших моделях (13B, аналогично тенденциям, показанным при сравнении MPT-30B и MPT-30B-Instruct), но минимально влияет на тенденции в более крупных моделях (70B).

5 Всегда ли больше контекста лучше? Тематическое исследование с открытым вопросно-ответным анализом

Наши результаты показывают, что предоставление языковым моделям более длинных входных контекстов — это компромисс: предоставление языковой модели большего объема информации может помочь ей выполнить задачу нижнего уровня, но также увеличивает объем контента, который модель должна анализировать, что может снизить точность. Даже если языковая модель может принимать 16K токенов, действительно ли полезно предоставлять 16K токенов контекста? Ответ на этот вопрос в конечном итоге зависит от конкретной задачи нижнего уровня, поскольку он зависит от маржинальной ценности добавленного контекста и способности модели эффективно ис-

пользовать длинные входные контексты, но мы проводим тематическое исследование с открытым вопросно-ответным анализом на NaturalQuestions-Open, чтобы лучше понять этот компромисс в существующих языковых моделях.

Мы используем языковые модели в стандартной настройке извлечения-читателя. Система извлечения (Contriever, дообученная на MS-MARCO) принимает входной запрос из NaturalQuestions-Open и возвращает k документов из Википедии с наивысшими оценками релевантности. Чтобы условно использовать языковые модели на этих извлеченных документах, мы просто включаем их в подсказку. Мы оцениваем извлечение и точность читателя (появляется ли какой-либо из аннотированных ответов в предсказанном выводе) в зависимости от количества извлеченных документов k. Мы используем подмножество NaturalQuestions-Open, где длинный ответ является абзацем (в отличие от таблицы или списка).

Рисунок 11 представляет результаты извлечения и открытого вопросно-ответного анализа. Мы видим, что производительность модели читателя насыщается задолго до насыщения извлечения, что указывает на то, что читатели неэффективно используют дополнительный контекст. Использование более чем 20 извлеченных документов лишь незначительно улучшает производительность читателя (∼1.5% для GPT-3.5-Turbo и ∼1% для Claude-1.3), в то время как значительно увеличивает длину входного контекста (а значит, задержку и стоимость). Эти результаты, в сочетании с наблюдением, что модели часто лучше извлекают и используют информацию в начале или в конце входных контекстов, предполагают, что эффективная переоценка извлеченных документов (перемещение релевантной информации ближе к началу входного контекста) или усечение ранжированного списка (извлечение меньшего количества документов, когда это уместно; Arampatzis et al., 2009) могут быть перспективными направлениями для улучшения того, как читатели на основе языковых моделей используют извлеченный контекст.

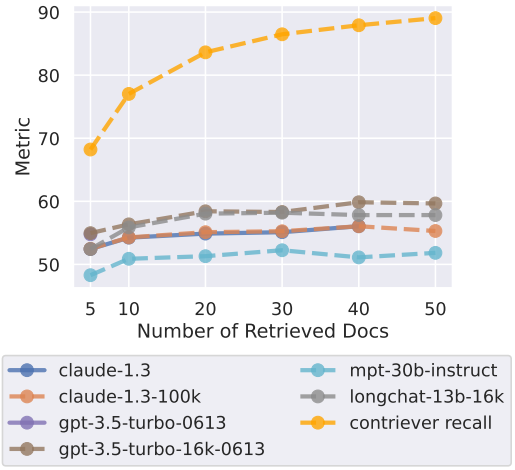

Рис. 11: Извлечение и производительность модели в зависимости от количества извлеченных документов. Производительность модели насыщается задолго до извлечения, что указывает на то, что модели испытывают трудности с использованием дополнительных извлеченных документов.

6 Связанные работы

6.1 Языковые модели с длинным контекстом

Существует много предыдущих работ по проектированию производительных языковых моделей с более дешевым масштабированием, чем у трансформеров в длине контекста. Многие направления работы преследуют варианты трансформеров с модификациями внимания, такими как рекуррентность (Dai et al., 2019), факторизация внимания в вычислительно менее интенсивные аппроксимации (Beltagy et al., 2020; Zaheer et al., 2020), или аппроксимации низкого ранга (Wang et al., 2020; Peng et al., 2021). Dao et al. (2022) вместо этого предоставляют более быстрое точное внимание с помощью тщательно разработанного CUDA-ядра, учитывающего ввод-вывод. Отдельно существуют попытки полностью отказаться от внимания, чтобы устранить квадратичную сложность длины последовательности, часто через свертки и/или линейные RNN, например, в RWKV (Peng, 2023), S4 (Gu et al., 2022), или Hyena (Poli et al., 2023). Многие предыдущие усилия оценивают перплексию на разнообразном веб-корпусе как прокси для способности обрабатывать длинные контексты; эта работа показывает, что точный

доступ к знаниям на длинных контекстах может быть дополнительной задачей.

6.2 Как языковые модели используют контекст?

Пионерская работа Khandelwal et al. (2018) показала, что малые LSTM языковые модели делают все более грубое использование более долгосрочного контекста; Sankar et al. (2019) обнаружили аналогичные результаты в моделях диалога. В аналогичном ключе, Daniluk et al. (2017) обнаружили, что внимательные LSTM языковые модели склонны в основном использовать недавнюю историю. Petroni et al. (2020) были одними из первых, кто продемонстрировал потенциал сочетания контекста из системы извлечения информации с предварительно обученными языковыми моделями для неуправляемого вопросно-ответного анализа. O'Connor and Andreas (2021) обнаружили, что многие операции, разрушающие информацию, мало влияют на предсказания трансформерных языковых моделей. Krishna et al. (2022) обнаружили, что генерация с длинным контекстом в умеренно больших трансформерных языковых моделях вырождается, потому что модели не могут правильно учитывать длинный контекст. Наконец, изучая модели с длинным контекстом, Sun et al. (2021) обнаружили, что более длинные контексты улучшают предсказание только нескольких токенов, эмпирическое наблюдение, согласующееся с теорией Sharan et al. (2018), которые показали, что распределения последовательностей с ограниченной взаимной информацией обязательно приводят к маргинальным средним преимуществам предсказания от все более длинного контекста. Qin et al. (2023) анализируют, как эффективные трансформеры выполняют различные задачи NLP с длинным контекстом, обнаруживая, что трансформеры с длинным контекстом склонны к недавности и неэффективно используют долгосрочный контекст.

6.3 Эффект серийной позиции

U-образная кривая, которую мы наблюдаем в этой работе, имеет связь в психологии, известную как эффект серийной позиции (Ebbinghaus, 1913; Murdock Jr, 1962), который утверждает, что при свободном воспроизведении элементов из списка люди склонны лучше запоминать первые и последние элементы списка. Эффект серийной позиции играет роль в понимании того, как люди развивают кратковременную и долговременную память. Наблюдение эффекта, подобного серийной позиции, в языковых моделях может быть удивительным, поскольку механизмы самовнимания, лежащие в основе трансформерных языковых моделей, технически одинаково способны извлекать любой токен из их контекстов.

7 Заключение

Мы эмпирически изучаем, как языковые модели используют длинные входные контексты через серию контролируемых экспериментов. Мы показываем, что производительность языковых моделей значительно ухудшается при изменении положения релевантной информации, что указывает на то, что модели испытывают трудности с надежным доступом и использованием информации в длинных входных контекстах. В частности, производительность часто наименьшая, когда модели должны использовать информацию в середине длинных входных контекстов. Мы проводим предварительное исследование роли (i) архитектуры модели, (ii) контекстуализации с учетом запроса и (iii) тонкой настройки инструкций, чтобы лучше понять, как они влияют на то, как языковые модели используют контекст. Наконец, мы завершаем практическим тематическим исследованием открытого вопросно-ответного анализа, обнаруживая, что производительность читателей на основе языковых моделей насыщается задолго до извлечения. Наши результаты и анализ дают лучшее понимание того, как языковые модели используют свой входной контекст, и предлагают новые протоколы оценки для будущих моделей с длинным контекстом.

Благодарности

Мы хотели бы поблагодарить Люка Цеттлемойера, который был нашим редактором действий TACL, и анонимных рецензентов за их комментарии и отзывы. Мы также благодарим Клаудиу Леовяну-Кондрея, Меган Лещински, Дмитрия Охонко, Майтру Рагху,

Эрика Уоллеса и Санг Майкла Си за отзывы и обсуждения, которые помогли улучшить эту работу. Кроме того, мы благодарны Севон Мин за её помощь с набором данных AmbigQA. Эта работа была поддержана Центром исследований на основе моделей Стэнфордского университета (CRFM), OpenAI через грант на кредиты API для CRFM Стэнфордского университета и Anthropic через программу академического доступа Claude.

References

Avi Arampatzis, Jaap Kamps, and Stephen Robertson. 2009. Where to stop reading a ranked list? threshold optimization using truncated score distributions. In Proc. of SIGIR.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. ArXiv:2004.05150.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. ArXiv:2210.11416.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In Proc. of ACL.

Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. In Proc. of ICLR.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. ArXiv:2205.14135.

Hermann Ebbinghaus. 1913. Memory: A contribution to experimental psychology. H. A. Ruger & C. E. Bussenius, Trans.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. In Proc. of ICLR.

Maor Ivgi, Uri Shaham, and Jonathan Berant. 2023. Efficient long-text understanding with short-text models. Transactions of the Association for Computational Linguistics, 11:284–299.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. ArXiv:2112.09118.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In Proc. of EACL.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2022. Large language models struggle to learn long-tail knowledge. ArXiv:2211.08411.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In Proc. of ACL.

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. 2022. RankGen: Improving text generation with large ranking models. In Proc. of EMNLP.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In Proc. of ACL.

Mina Lee, Percy Liang, and Qian Yang. 2022. CoAuthor: Designing a human-AI collaborative writing dataset for exploring language model capabilities. In Proc. of CHI.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, , and Hao Zhang. 2023. How long can open-source LLMs truly promise on context length?

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Proc. of ACL.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In Proc. of EMNLP.

Bennet B. Murdock Jr. 1962. The serial position effect of free recall. Journal of experimental psychology, 64(5):482.

Joe O'Connor and Jacob Andreas. 2021. What context features can Transformer language models use? In Proc. of ACL.

Dimitris Papailiopoulos, Kangwook Lee, and Jy-yong Sohn. 2023. A little retrieval test for large language models. https://github.com/anadim/the-little-retrieval-test.

Bo Peng. 2023. RWKV-LM. https://github.com/BlinkDL/RWKV-LM.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2021. Random feature attention. In Proc. of ICLR.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In Proc. of AKBC.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. In Proc. of ICML.

Ofir Press, Noah A. Smith, and Mike Lewis. 2021. Shortformer: Better language modeling using shorter inputs. In Proc. of ACL.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In Proc. of ICLR.

Guanghui Qin, Yukun Feng, and Benjamin Van Durme. 2023. The NLP task effectiveness of long-range transformers. In Proc. of EACL.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text Transformer. Journal of Machine Learning Research, 21(140):1–67.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. ArXiv:2302.00083.

Ohad Rubin and Jonathan Berant. 2023. Long-range language modeling with self-retrieval. ArXiv:2306.13421.

Chinnadhurai Sankar, Sandeep Subramanian, Chris Pal, Sarath Chandar, and Yoshua Bengio. 2019. Do neural dialog systems use the conversation history effectively? an empirical study. In Proc. of ACL.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. ZeroSCROLLS: A zero-shot benchmark for long text understanding. ArXiv:2305.14196.

Vatsal Sharan, Sham Kakade, Percy Liang, and Gregory Valiant. 2018. Prediction with a short memory. In Proc. of STOC.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. REPLUG: Retrieval-augmented black-box language models. ArXiv:2301.12652.

Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, Morteza Behrooz, William Ngan, Spencer Poff, Naman Goyal, Arthur Szlam, Y-Lan Boureau, Melanie Kambadur, and Jason Weston. 2022. BlenderBot 3: a deployed conversational agent that continually learns to responsibly engage. ArXiv:2208.03188.

Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. 2021. Do long-range language models actually use long-range context? In Proc. of EMNLP.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. UL2: Unifying language learning paradigms. ArXiv:2205.05131.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. LaMDA: Language models for dialog applications. ArXiv:2201.08239.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and efficient foundation language models. ArXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. ArXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proc. of NeurIPS.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer:

Self-attention with linear complexity. ArXiv:2006.04768.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for longer sequences. In Proc. of NeurIPS.

A Неопределенность в отвлекающих документах многодокументного вопросно-ответного анализа

Следуя предыдущим работам на NaturalQuestions-Open (Izacard et al., 2021; Izacard and Grave, 2021, inter alia), мы используем дамп Википедии с конца 2018 года в качестве нашего корпуса извлечения. Однако этот стандартный дамп Википедии имеет небольшое временное несоответствие с аннотациями NaturalQuestions.

Например, рассмотрим вопрос «в какой команде НФЛ играет Роберт Гриффин III». Аннотированный ответ NaturalQuestions — «в настоящее время свободный агент». Однако корпус извлечения Википедии содержит информацию о том, что он играет за «Балтимор Рэйвенс», так как он был освобожден из команды между временной меткой дампа Википедии и процессом аннотирования NaturalQuestions.

Мы используем аннотации неопределенности Min et al. (2020), чтобы создать подмножество однозначных вопросов. Эксперименты на этом однозначном подмножестве данных показывают аналогичные результаты и выводы, как и эксперименты на полном наборе вопросов (Рисунок 12).

B Случайные отвлекающие в многодокументном вопросно-ответном анализе

Мы также проводим эксперименты с многодокументным вопросно-ответным анализом с случайными документами из Википедии в качестве отвлекающих, что позволяет нам абстрагировать влияние извлеченных отвлекающих (жестких отрицательных). Обратите внимание, что в этой настройке документ, содержащий ответ, часто можно идентифицировать с помощью простых эвристик (например, лексическое совпадение с запросом).

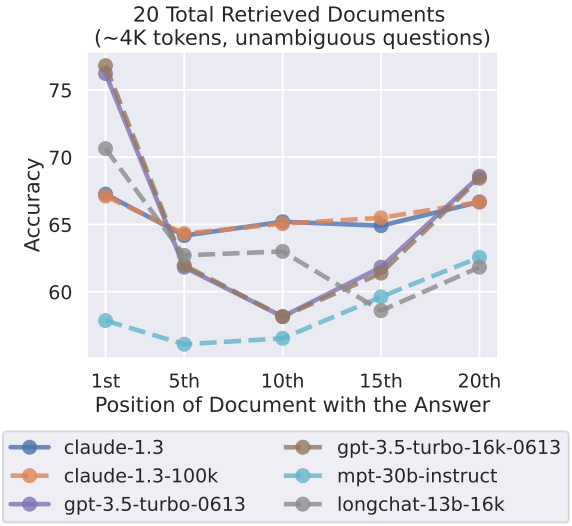

Рис. 12: Производительность языковой модели на однозначном подмножестве вопросов.

Рисунок 13 представляет результаты этого эксперимента. Хотя все модели имеют более высокую абсолютную точность в этой настройке, они удивительно все еще испытывают трудности с анализом всего своего входного контекста, что указывает на то, что их ухудшение производительности не связано исключительно с неспособностью идентифицировать релевантные документы.

C Случайное упорядочивание отвлекающих в многодокументном вопросно-ответном анализе

Наша подсказка инструктирует языковую модель использовать предоставленные результаты поиска для ответа на вопрос. Возможно, в данных предварительного обучения или тонкой настройки инструкций существует предположение, что результаты поиска отсортированы по убыванию релевантности (т. е. документы ближе к началу входного контекста более вероятно полезны, чем те, что в конце). Чтобы убедиться, что наши выводы не являются просто следствием этого искажения, мы проводим эксперименты с измененной инструкцией «Напишите качественный ответ на заданный вопрос, используя только предоставленные результаты поиска (некоторые из которых могут быть нерелевантными). Результаты поиска упорядочены случайным образом.» Кроме того, мы случайно перемешиваем $k - 1$ отвлекаю-

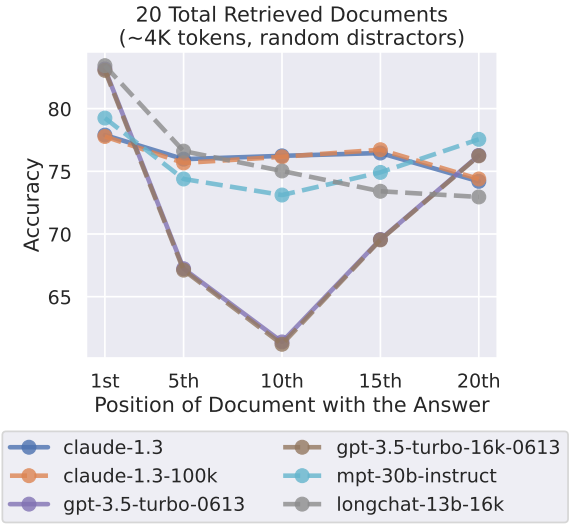

Рис. 13: Производительность языковой модели на многодокументном вопросно-ответном анализе при использовании случайных отвлекающих, а не извлеченных отвлекающих.

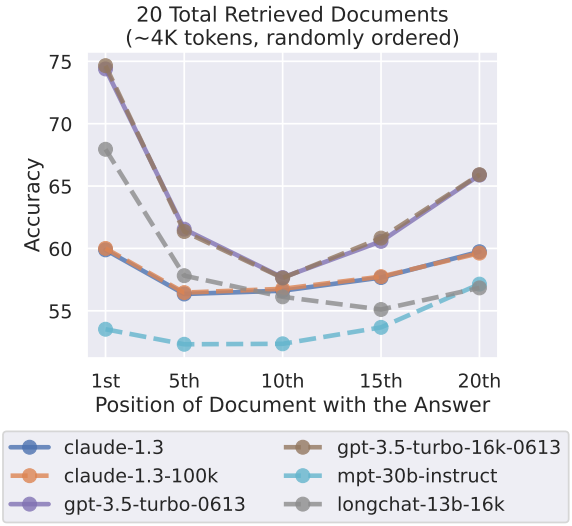

Рис. 14: Производительность языковой модели при случайном упорядочивании отвлекающих (вместо представления их в порядке убывания релевантности) и упоминании об этом в подсказке.

щих документов.

Рисунок 14 представляет результаты этого эксперимента. Мы продолжаем видеть U-образную кривую производительности, с ухудшением производительности, когда языковые модели должны использовать информацию в середине своих входных контекстов. Сравнивая результаты в §2.3 с теми, когда случайно упорядочиваем отвлекающие и упоминаем об этом в подсказке, мы видим, что рандомизация слегка снижает производительность, когда релевантная информация находится в самом начале контекста, и слегка увеличивает производительность, когда используется информация в середине и конце контекста.

D Производительность GPT-4

Мы оцениваем GPT-4 (8K) на подмножестве 500 случайных примеров многодокументного вопросно-ответного анализа с 20 всего документами в каждом входном контексте (Рисунок 15). GPT-4 достигает более высокой абсолютной производительности, чем любая другая языковая модель, но все еще показывает U-образную кривую производительности — её производительность наивысшая, когда релевантная информация находится в самом начале или в конце контекста, и производительность ухудшается, когда она должна использовать информацию в середине своего входного контекста.

E Производительность Llama-2

Мы оцениваем Llama-2 (Touvron et al., 2023b) на многодокументном вопросно-ответном анализе с 20 всего документами в каждом входном контексте. Токенизатор Llama производит более длинные последовательности, чем токенизаторы для наших ранее изученных моделей, поэтому мы отбрасываем 20 примеров (из 2655), которые превышают максимальную длину контекста Llama-2 в 4096 токенов. Мы экспериментируем с моделями различных размеров (7B, 13B и 70B параметров), с и без дополнительного супервизорного дообучения и обучения с подкреплением от человеческой обратной связи (модели «-chat-»). Результаты представлены на Рисунке 16.

Сравнивая модели Llama-2 различных размеров, мы обнаруживаем, что только более крупные модели (13B и 70B) демонстрируют U-образную кривую производительности (т. е. как эффект первичности, так и эффект недавности) — самые маленькие модели Llama-2 (7B) исключительно склонны к недавности. Учитывая эти результаты, мы предполагаем, что предыдущие работы (например, Khandelwal et al., 2018; Sun et al., 2021) не наблюдали никакого эффекта пер-

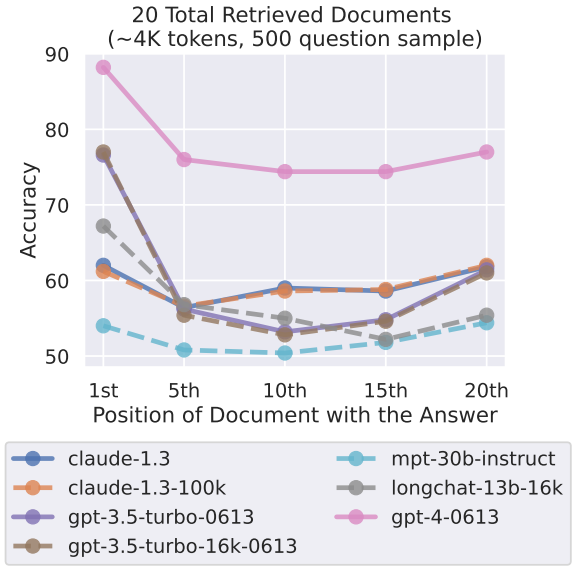

Рис. 15: Хотя GPT-4 имеет более высокую абсолютную производительность, чем другие модели, её производительность все еще ухудшается, когда релевантная информация находится в середине входного контекста.

вичности в языковых моделях, потому что изучаемые ими модели были слишком малы (менее 1B параметров).

Сравнивая между моделями Llama-2 с и без дополнительного супервизорного дообучения и обучения с подкреплением от человеческой обратной связи, мы видим, что дополнительное дообучение значительно улучшает производительность в задаче многодокументного вопросно-ответного анализа. Модели 7B с и без дополнительного дообучения показывают минимальный эффект первичности и в основном склонны к недавности. Базовая модель 13B имеет значительный эффект первичности и недавности — существует 20-процентное расхождение в точности между наилучшей и наихудшей производительностью. Применение дополнительного дообучения к модели 13B, кажется, слегка уменьшает это искажение (10-процентное ухудшение в худшем случае), но искажение остается значительным. Однако модели 70B с и без дополнительного дообучения имеют в основном схожие тенденции (демонстрируя как эффект первичности, так и эффект недавности), и дополнительное дообучение минимально влияет на степень позиционного искажения.

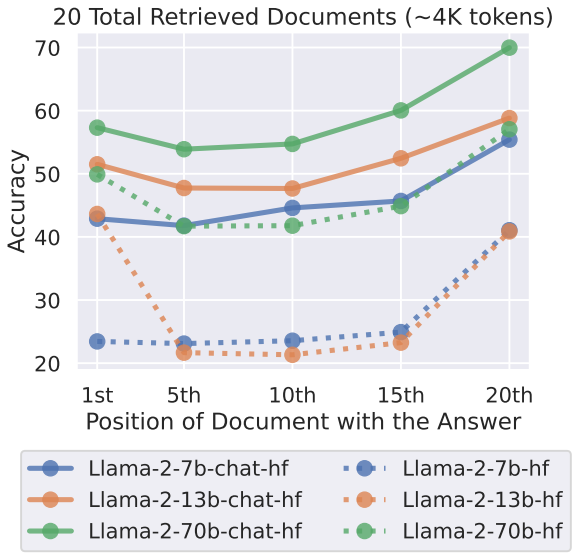

Рис. 16: Производительность многодокументного вопросно-ответного анализа (20 всего документов) моделей Llama-2 различных размеров (7B, 13B, 70B параметров), с и без дополнительного супервизорного дообучения и обучения с подкреплением от человеческой обратной связи (модели «-chat-»).

F Подсчет токенов

Таблица 2, Таблица 3 и Таблица 4 представляют среднее и максимальное количество токенов в каждом из входных контекстов для всех экспериментальных настроек. Обратите внимание, что MPT-30B и MPT-30B-Instruct используют один и тот же токенизатор, GPT-3.5-Turbo и GPT-3.5-Turbo (16K) используют один и тот же токенизатор, а Claude-1.3 и Claude-1.3 (100K) используют один и тот же токенизатор. Более того, токенизатор Claude-1.3 такой же, как и токенизатор GPT-3.5-Turbo, за исключением некоторых дополнительных специальных токенов, которые не появляются в наших данных. В результате количество токенов для этих двух семейств моделей одинаково в наших экспериментальных настройках.

	Closed-Book		Oracle	
	avg ± stdev	max	avg ± stdev	max
LongChat-13B (16K)	55.6 ± 2.7	70	219.7 ± 48.5	588
MPT-30B	43.5 ± 2.2	58	187.9 ± 41.8	482
GPT-3.5-Turbo	15.3 ± 2.2	29	156.0 ± 41.8	449
Claude-1.3	15.3 ± 2.2	29	156.0 ± 41.8	449

Таблица 2: Token count statistics for each of the evaluated models on the closed-book and oracle multi-document question answering settings.

	10 docs		20 docs		30 docs	
	avg ± stdev	max	avg ± stdev	max	avg ± stdev	max
LongChat-13B (16K)	1749.9 ± 112.4	2511	3464.6 ± 202.3	4955	5181.9 ± 294.7	7729
MPT-30B	1499.7 ± 88.5	1907	2962.4 ± 158.4	3730	4426.9 ± 230.5	5475
GPT-3.5-Turbo	1475.6 ± 86.5	1960	2946.2 ± 155.1	3920	4419.2 ± 226.5	6101
Claude-1.3	1475.6 ± 86.5	1960	2946.2 ± 155.1	3920	4419.2 ± 226.5	6101

Таблица 3: Token count statistics for each of the evaluated models on each of the document question answering settings.

	75 KV pairs		140 KV pairs		300 KV pairs	
	avg ± stdev	max	avg ± stdev	max	avg ± stdev	max
LongChat-13B (16K)	5444.5 ± 19.1	5500	10072.4 ± 24.1	10139	21467.3 ± 35.9	21582
MPT-30B	4110.5 ± 23.8	4187	7600.9 ± 31.1	7687	16192.4 ± 46.6	16319
GPT-3.5-Turbo	3768.7 ± 25.6	3844	6992.8 ± 34.1	7088	14929.4 ± 50.7	15048
Claude-1.3	3768.7 ± 25.6	3844	6992.8 ± 34.1	7088	14929.4 ± 50.7	15048

Таблица 4: Token count statistics for each of the evaluated models on each of the key-value (KV) retrieval settings.

G Полные результаты многодокументного вопросно-ответного анализа

Этот раздел содержит таблицы производительности моделей при оценке на задаче многодокументного вопросно-ответного анализа с различным количеством документов (Рисунок 5). «Индекс n» указывает производительность, когда документ с ответом находится на позиции $n + 1$, где более низкие индексы ближе к началу входного контекста. Например, индекс 0 относится к производительности, когда документ с ответом размещен в самом начале контекста (т. е. первым среди всех документов).

G.1 10 всего извлеченных документов

Model	Index 0	Index 4	Index 9
Claude-1.3	62.9%	58.3%	59.7%
Claude-1.3 (100K)	63.1%	58.3%	59.7%
GPT-3.5-Turbo	76.8%	61.2%	62.4%
GPT-3.5-Turbo (16K)	76.9%	61.0%	62.5%
MPT-30B-Instruct	60.2%	56.2%	59.7%
LongChat-13B (16K)	72.1%	58.9%	58.5%

Таблица 5: Производительность модели при оценке задачи вопросов и ответов с использованием 10 извлечённых документов.

G.2 20 всего извлеченных документов

Model	Index 0	Index 4	Index 9	Index 14	Index 19
Claude-1.3	59.9%	55.9%	56.8%	57.2%	60.1%
Claude-1.3 (100K)	59.8%	55.9%	57.0%	57.4%	60.0%
GPT-3.5-Turbo	75.8%	57.2%	53.8%	55.4%	63.2%
GPT-3.5-Turbo (16K)	75.7%	57.3%	54.1%	55.4%	63.1%
MPT-30B-Instruct	53.7%	51.8%	52.2%	52.7%	56.3%
LongChat-13B (16K)	68.6%	57.4%	55.3%	52.5%	55.0%

Таблица 6: Производительность модели при оценке задачи вопросов и ответов с использованием 20 извлечённых документов.

G.3 30 всего извлеченных документов

Model	Index 0	Index 4	Index 9	Index 14	Index 19	Index 24	Index 29
Claude-1.3	59.1%	55.1%	54.8%	55.7%	56.4%	56.2%	59.9%
Claude-1.3 (100K)	59.1%	55.1%	54.9%	55.7%	56.6%	56.1%	60.0%
GPT-3.5-Turbo (16K)	73.4%	55.1%	50.5%	50.9%	51.8%	54.9%	63.7%
MPT-30B-Instruct	51.6%	51.3%	51.2%	49.0%	49.6%	51.3%	54.1%
LongChat-13B (16K)	66.9%	54.8%	52.5%	52.9%	52.2%	51.3%	55.1%

Таблица 7: Производительность модели при оценке задачи вопросов и ответов с использованием 30 извлечённых документов.