УДК 004.422+004.9

ИНСТРУМЕНТ ДЛЯ ОПЕРАТИВНОЙ ДИАГНОСТИКИ ПАМЯТИ НЕЙРОСЕТЕВЫХ АРХИТЕКТУР ЯЗЫКОВЫХ МОДЕЛЕЙ

П. А. Гавриков[1] [0009-0009-6175-2883], А. К. Усманов[2] [0009-0006-2535-1052], Д. Реваев[3] [0009-0008-1630-1912], С. Н. Бузыканов[4] [0009-0005-3405-0705]

[1-4]Московский физико-технический институт, г. Долгопрудный, Московская обл., Россия

[1]gavrikov.pa@phystech.edu, [2]usmanov.ak@phystech.edu, [3]revaev.d@phystech.edu, [4]bsn1977@mail.ru

Аннотация

Большие языковые модели (Large Language Models, LLM) прошли путь от простых N-граммных систем до современных универсальных архитектур, однако ключевым ограничением остается квадратичная сложность механизма самовнимания по длине входной последовательности. Это существенно увеличивает потребление памяти и вычислительных ресурсов, а с появлением задач, требующих рекордно длинных контекстов, создает необходимость разработки новых архитектурных решений. Поскольку для исследования предлагаемой архитектуры требуется длительное и дорогостоящее обучение полновесной сети, необходимо разработать инструмент, который позволял бы быстро дать предварительную оценку архитектуре с точки зрения внутренней памяти.

В настоящей работе предложен метод количественной оценки внутренней памяти нейросетевых архитектур на основе синтетических тестов, не требующих больших корпусов данных. Под внутренней памятью понимается объем информации, который модель способна воспроизвести без обращения к исходным входам.

Для верификации подхода разработан программный комплекс, апробированный на архитектурах GPT-2 и Mamba. Использованы задачи копирования, инверсии и извлечения значения по ключу. Проведенное сравнение по точности предсказаний, распределению ошибок и вычислительным

затратам позволяет оперативно оценивать эффективность и перспективность архитектур LLM.

***Ключевые слова**: большие языковые модели, архитектура нейросетей, внутренняя память, долговременное хранение информации, обработка последовательностей, измерение функциональной памяти, сравнение архитектур.*

ВВЕДЕНИЕ

В последние годы искусственный интеллект достиг значительных успехов в области обработки естественного языка. Особенно примечательным стало развитие больших языковых моделей (Large Language Models, LLM), способных не только продолжать текст, но и выполнять сложные когнитивные задачи: от ответа на вопросы и перевода до программирования и логического рассуждения. Эти модели, построенные на нейросетевых архитектурах, стали основой для создания новых поколений интеллектуальных систем, способных взаимодействовать с человеком на естественном языке. Их широкое распространение стало возможным благодаря доступности больших объемов текстовых данных, увеличению вычислительных мощностей и совершенствованию алгоритмических решений в глубоком обучении [1].

Одним из ключевых факторов, определяющих эффективность языковых моделей, является длина контекста, с которым модель способна работать. Под контекстом в данном случае понимается последовательность исходных текстовых единиц (токенов), из которых модель «понимает» смысл сообщения, определяет стиль ответа и извлекает релевантную информацию. Чем длиннее контекст, тем больше информации может быть учтено при генерации следующего выходного элемента. Однако с увеличением длины входа возрастают и вычислительные затраты, что накладывает ограничения на используемые архитектурные решения. В частности, в большинстве современных моделей на практике ограничение длины контекста составляет несколько тысяч текстовых единиц [2]. Это становится серьезным препятствием при решении задач, связанных с анализом длинных документов, поддержанием диалога на

протяжении нескольких обменов репликами или извлечением информации из больших массивов текста [3].

Задача работы с длинными контекстами приобрела актуальность в последние годы в результате совокупного влияния технологических и прикладных факторов. Ранние языковые модели, такие как GPT-2 [4], имели сравнительно небольшие окна контекста (512–1024 токена) и обучались на фрагментах текстов соответствующего размера. Основной задачей было научиться предсказывать следующую единицу текста и улучшить качество генерации. Многочисленные улучшения достигались за счет увеличения числа параметров и размера обучающих выборок. Длина окна внимания при этом оставалась второстепенной характеристикой, поскольку типичные задачи (продолжение текста, классификация, перевод) не требовали обработки больших объемов входной информации.

Ситуация изменилась в 2022–2023 г., когда появились новые классы задач, требующих долгосрочного хранения и использования информации. Среди них – многошаговые диалоги в чат-ботах, анализ длинных юридических и технических документов, понимание и редактирование исходного кода программ, а также цепочки логических рассуждений. Эти задачи подразумевают, что модель должна оперировать не десятками, а тысячами и десятками тысяч единиц текста за один раз [5]. Окна контекста длиной 2048 или 4096 перестали быть достаточными.

На фоне этих требований возникла техническая проблема: архитектура трансформера [6], лежащая в основе большинства современных языковых моделей, в существующем виде оказывается плохо масштабируемой при увеличении длины контекста. Это накладывает серьезные ограничения на использование модели в задачах, требующих анализа длинных входных последовательностей [1]. В результате возникает необходимость в оптимизации существующих архитектур или разработке новых подходов, способных эффективно справляться с большими объемами входных данных.

Один из наиболее очевидных и формально корректных способов оценки памяти модели – это полноценное обучение на больших объемах реальных данных (например корпусах новостных или разговорных текстов) и последующий анализ поведения модели при варьировании длины входного контекста. Такой

подход, несмотря на свою корректность, имеет целый ряд серьезных недостатков. Обучение современных языковых моделей даже среднего размера требует сотен GPU-часов и масштабной инфраструктуры [1]. В условиях разработки новых архитектур, когда нужно протестировать множество гипотез и модификаций,
такой подход становится крайне неэффективным: стоимость одной неудачной архитектурной итерации становится неоправданно высокой. Кроме того, из-за большого числа параметров (слои, головы (heads), размеры внутренних представлений и т. д.) обучение каждой новой конфигурации становится дорогим по времени и требует длительного инженерного цикла [7]. Таким образом, метод «обучения и тестирования на полном корпусе» тормозит исследовательскую активность и ограничивает возможности итеративной разработки.

Поскольку на сегодняшний день отсутствует инструмент, который бы позволял не только быстро оценить способность модели к запоминанию и воспроизведению длинных контекстов данных, но и не требовал бы при этом огромных корпусов натуральных данных, возникает необходимость создания инструмента для быстрой оценки перспективных архитектур с точки зрения памяти.

ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Существует несколько классов методов, так или иначе позволяющих оценить архитектуру языковой модели на разных этапах ее жизненного цикла: от предварительного анализа архитектурных решений до профилирования при постобучении. Ниже приведен обзор наиболее репрезентативных подходов и средств к обучению, а также анализ их применимости к задаче оценки памяти и способности модели к работе с длинным контекстом.

Одним из наиболее компактных и при этом широко обсуждаемых датасетов в сообществе разработчиков языковых моделей является TinyStories – синтетический корпус, представленный в 2023 г. исследователями из Microsoft Research [8]. Датасет был создан в ответ на потребность в специализированных тренировочных данных для обучения малых языковых моделей объемом от 10 до 100 млн параметров. В отличие от масштабных корпусов, таких как The Pile [9] или

Common Crawl [10], TinyStories ориентирован на минимализм, простоту синтаксиса и ограниченность лексического разнообразия, что делает его пригодным для быстрого обучения и интерпретируемого анализа поведения моделей.

Однако в контексте оценки способности языковой модели к обработке длинных контекстов датасет TinyStories оказывается ограниченно применимым. Поскольку длина каждой истории составляет в среднем менее 300 текстовых единиц, модель, обучающаяся на этом корпусе, практически не сталкивается с необходимостью оперировать удаленными по позиции фрагментами текста. Отсутствие длинных зависимостей и компактная структура входа означают, что модель не развивает способности к долгосрочному контекстному запоминанию и ее механизмы внимания не подвергаются реальному стресс-тесту. Соответственно, TinyStories не позволяет протестировать или сравнить архитектуры по признакам, важным для задач с удлиненным контекстом: устойчивости представлений при увеличении длины входа, способности сохранять релевантную информацию на больших расстояниях, точности адресации при ассоциативном поиске и др.

Одним из наиболее масштабных и широко используемых источников текстовых данных для обучения больших языковых моделей является датасет The Pile [9], разработанный исследовательским коллективом EleutherAI в 2020 г. Этот корпус представляет собой специально составленный набор текстов объемом порядка 825 гигабайт, предназначенный для обучения авторегрессионных языковых моделей.

Объем и разнообразие данных делают The Pile одним из наиболее полных и репрезентативных наборов текстов в открытом доступе. Однако его применение сопряжено с рядом технических ограничений. В первую очередь, обучение на полном корпусе требует значительных вычислительных ресурсов [1]. Даже при использовании оптимизированных программных решений и современных графических ускорителей обучение модели среднего масштаба может занимать несколько недель. Это делает невозможным широкое применение набора The Pile в условиях быстрой прототипизации или исследования множества архитектурных гипотез. Другой проблемой корпуса

является неоднородность качества исходных данных. Некоторые подкорпуса, например, GitHub или Common Crawl (Pile-CC), включают в себя тексты со значительным уровнем шума или нерелевантных фрагментов.

При всей своей ценности как обучающего корпуса, The Pile не подходит для решения задач быстрой оценки архитектурных особенностей моделей. Его масштаб и связанная вычислительная нагрузка делают набор неприменимым в сценариях, где требуется быстрая обратная связь, например при тестировании памяти модели или устойчивости к длинным зависимостям. В подобных случаях более уместны компактные, синтетические задачи, позволяющие получить интерпретируемые показатели в кратчайшие сроки и без необходимости полного обучения.

FlashAttention – это специализированный алгоритм эффективного вычисления механизма внимания, предложенный Трием Фонгом с соавторами в 2022 г. [11]. Его ключевая цель – устранение избыточных обращений к памяти при вычислении операций внимания (attention) в трансформерах. Это достигается путем переработки порядка выполнения операций и управления памятью таким образом, чтобы данные не записывались и не считывались из памяти GPU больше одного раза. Алгоритм FlashAttention позволяет добиться ускорения в 2–4 раза по сравнению с классической реализацией внимания softmax, особенно на больших длинах последовательностей.

Основное нововведение метода заключается в использовании блочного (tile-based) подхода к вычислению матриц внимания, при котором входная последовательность делится на небольшие блоки. Каждая операция выполняется в пределах локального хранилища, что позволяет держать промежуточные значения в регистровой памяти или кэшах высокого уровня. Это существенно сокращает пропускную способность ввода-вывода и минимизирует объем трафика между GPU DRAM и ядрами.

С практической точки зрения алгоритм FlashAttention особенно полезен при работе с длинными контекстами, где классическая реализация механизма внимания становится узким местом как по времени, так и по памяти. В задачах обработки длинных текстов (1000–64000 токенов и более) FlashAttention обеспечивает как ускорение, так и снижение потребления видеопамяти. Однако

важно понимать, что FlashAttention не является архитектурным улучшением, а представляет собой инженерную оптимизацию одного из компонентов стандартного трансформера. Поэтому его использование не расширяет функциональные возможности модели, а лишь делает существующие механизмы внимания быстрее и дешевле в использовании. Более того, он сохраняет квадратичную сложность по длине контекста, хотя и с меньшим коэффициентом. Это означает, что даже с использованием FlashAttention трансформер остается плохо масштабируемым при экстремально длинных последовательностях (например, 100000 единиц текста и более), что ограничивает его применимость в задачах долговременной памяти.

В контексте задач быстрой оценки новых архитектур механизм FlashAttention может быть полезен в тех случаях, когда исследователь планирует использовать классические блоки внимания в трансформере и ищет способ ускорить обучение на упрощенных корпусах. Тем не менее он по-прежнему требует обучения модели хотя бы на части данных и не решает проблему эффективной диагностики архитектуры без дорогостоящего предобучения. Более того, архитектуры, не основанные на механизме самовнимания (например, Mamba [12], S4 [7]), не могут использовать алгоритм FlashAttention напрямую. Это ограничивает область применения и подчеркивает, что этот механизм является способом повышения эффективности существующего архитектурного шаблона трансформера, нежели универсальным средством для оценки или разработки новых языковых моделей.

Одной из ключевых оптимизаций, применяемых в больших языковых моделях при генерации текста, является использование так называемого KV-кэша (Key-Value Cache) [13]. Этот механизм позволяет значительно сократить вычислительные затраты при автогенерации последовательности, устраняя необходимость многократного пересчета коэффициентов внимания ко всем предыдущим позициям текста на каждом новом шаге вывода. Он особенно важен для трансформеров в режиме авторегрессионной генерации, где текст генерируется по одному токену за раз.

Суть KV-кэширования заключается в размене части времени вычисления ответа обученной модели во время генерации на память: матрицы K и V для

каждого слоя и каждой головы трансформера сохраняются в кэше и переиспользуются, а не пересчитываются на каждом шаге, что позволяет добиться линейного времени генерации относительно длины входной последовательности.

Тем не менее, как и алгоритм FlashAttention, KV-кэш не является архитектурной инновацией. Это инженерная оптимизация, направленная на повышение производительности стандартного механизма самовнимания. Кроме того, он применим только к авторегрессионным моделям с декодером и не используется, например, в энкодерных архитектурах (BERT) или в задачах без автогенерации. В контексте задач оценки новых архитектур по способности к работе с длинными последовательностями KV-кэш не предоставляет полезной информации о внутренней памяти модели. Он не влияет на архитектурную глубину или когнитивные способности, а только ускоряет выполнение уже обученной модели. Это делает его ценным инструментом для развертывания, но ограниченно применимым в задачах архитектурного анализа и ранней диагностики.

ОПИСАНИЕ МЕТОДА

Предлагаемое решение включает в себя количественное упрощение модели, использование конкретных синтетических задач и сбор показателей в процессе и по итогам обучения.

Количественное упрощение модели

Поскольку целью применения описываемого инструмента является оценка именно характеристик памяти, предполагается, что упомянутые характеристики можно оценить без обучения полноразмерной модели. Используются конфигурации сетей, количество слоев, голов внимания и размер внутреннего представления в которых были сокращены до предела, при котором модели еще сохраняют свои свойства выразительности, но уже могут обучаться достаточно быстро для практической применимости метода. Предполагается, что архитектура, показавшая себя лучше других при небольшом числе параметров, покажет себя лучше и при полноразмерной реализации.

Нужно отметить, что описываемый метод не предполагает ни количественной оценки «достаточного» числа параметров или слоев, которое необходимо сохранить для выразительности архитектуры, ни алгоритма получения такой оценки. Поскольку при использовании описываемого метода время тестирования одной архитектуры измеряется в часах, а не в днях и неделях, можно подобрать минимально выразительную конфигурацию эмпирически. Тем не менее это требует дополнительного времени исследователя и остается актуальным вопросом при большом количестве тестируемых архитектур.

Синтетические диагностические задачи

Ключевой составляющей метода является обучение модели на наборе из трех синтетических задач, отобранных по следующим критериям.

1. Простота генерации: каждая задача должна быть легко воспроизводиться, не должна требовать хранения больших корпусов данных или знания конкретного естественного языка, предметной области или какой-либо характеристики словаря. Такое требование позволяет оценивать модель безотносительно корпусов предобучения и сконцентрироваться на характеристиках памяти.
2. Диагностическая специфичность: каждая задача должна оценивать конкретный аспект работы внутренней памяти модели, позволяя делать интерпретируемые выводы.
3. Сложность решения: задачи должны последовательно увеличивать нагрузку на память, от базового воспроизведения до извлечения структурированной информации, позволяя начинать с оценки базовой способности воспроизвести информацию и заканчивая пониманием структуры входа и взаимосвязи частей этой структуры.

В соответствии с этими критериями были выбраны три синтетические задачи, представленные ранее в работе [15]:

1) задача копирования (Copy Task) – модель должна воспроизвести входную последовательность без изменений. Эта задача является базовой и позволяет оценить способность модели к удержанию линейной последовательности на всем протяжении контекста. Она особенно

чувствительна к затуханию градиентов и забыванию на больших расстояниях;

2) задача переворота (Reverse Task) – модель должна вывести ту же последовательность, но в обратном порядке. Это требует от модели не только хранения, но и перестройки последовательности во внутреннем представлении, что отражает способность к манипулированию содержимым памяти;
3) задача ассоциативного извлечения (Associative Retrieval Task) – во входную последовательность включаются пары «ключ – значение», а также отдельный запрос по ключу. Модель должна вернуть соответствующее значение. Эта задача проверяет наличие структурного представления во внутреннем состоянии модели, близкого к ассоциативной памяти.

Порядок задач соответствует увеличению когнитивной сложности. Если в первом случае достаточно механической передачи информации, то в третьей задаче модель должна интерпретировать структуру входа и избирательно реагировать на заданный шаблон.

Метрики и статистики

Для всесторонней оценки качества модели и диагностики ее поведения применялся набор аналитических показателей, получаемых в процессе обучения и непосредственного применения сети. Все они рассчитаны на анализ в пределах одной порции обрабатываемых данных, что позволяет осуществлять интерактивный контроль за ходом эксперимента.

Собирались следующие статистики.

1. Распределение точности предсказания по номеру токена внутри одного примера входа. Этот показатель отражает, какие части последовательности воспроизводятся хуже, и позволяет выявить позиционную деградацию памяти (например, хуже воспроизводится вторая половина текста).
2. Объем используемой оперативной памяти во время обучения. Этот показатель позволяет сопоставить объем памяти, необходимой для

вычисления ответа, с параметрами модели и задачей, давая косвенную оценку аппаратной стоимости архитектуры.

3. Время обучения одной эпохи. Этот параметр является важным показателем инженерной эффективности, он позволяет сравнивать архитектуры по затратам на обучение при прочих равных условиях.

4. Распределение вклада токенов в функцию потерь. Показывает, какие участки входа дали наибольший вклад в ошибку модели и, следовательно, были плохо предсказаны. Он дополняет показатель точности и позволяет глубже интерпретировать поведение сети.

Основным критерием оценки является максимальная длина последовательности, при которой модель сохраняет приемлемую среднюю точность ответа (Accuracy). Порог допустимого падения точности задавался эмпирически. При снижении качества ниже установленного порога считалось, что модель утратила способность к удержанию информации на данной длине контекста. Таким образом, определение «внутренней памяти» сводится к выявлению границы длины, при которой архитектура сохраняет функциональность.

ЭКСПЕРИМЕНТЫ

Механика подачи данных

Поскольку некоторые модели имеют строго ограниченную длину входа, в течение всех экспериментов данные подавались единым образом – посредством скользящего окна. Это позволяет проводить эксперименты независимо от архитектуры сети. В таком случае в каждом эксперименте появляются две интересующие нас переменные – длина окна, подававшегося на каждом шаге эксперимента на вход, и текущая длина последовательности, которая может превышать длину окна контекста или быть меньше этой длины.

Архитектуры

В качестве исследуемых моделей были выбраны три архитектуры: GPT-2, Mamba и GPT-X. Первые две хорошо изучены.

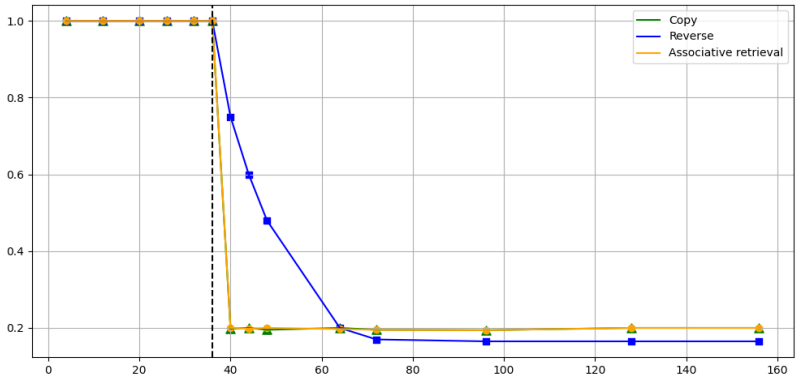
GPT-2 представляет собой классический трансформер с авторегрессионным вниманием.

Mamba представляет собой линейно масштабируемую по памяти и времени архитектуру, основанную на идеях рекуррентной обработки входа. В отличие от трансформеров, она не использует механизм самовнимания и сохраняет внутреннее состояние между шагами, что делает ее перспективной для задач с длинными зависимостями. Однако из-за последовательной природы вычислений Mamba не позволяет эффективно использовать параллелизм на видеокартах, что делает ее обучение медленным и ресурсоемким по сравнению с моделями, основанными на механизме внимания.

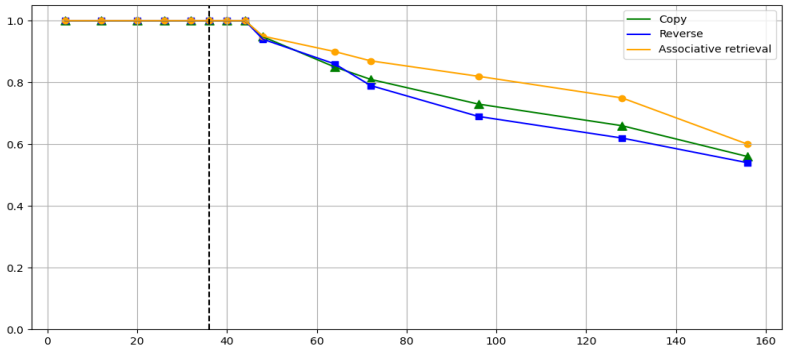
Модель GPT-X включена в исследование как объект с заранее неизвестным поведением, что делает ее тестирование особенно показательным. Результаты ее тестирования позволяют оценить пригодность предложенного инструмента для реального применения.

Результаты экспериментов

На рис. 1 видно, что с достижением предела окна контекста модель GPT-2 резко теряет в качестве предсказания, в то время как Mamba продолжает отвечать на сравнительно высоком уровне точности. Как известно, классическая архитектура трансформера не предполагает внутренних структур памяти, из-за чего такая модель не имеет возможности сохранять где-либо во внутреннем представлении участки входа, которые она видела ранее вне текущего окна. Поэтому при выходе первого токена за пределы окна, как при длине 36, модель не может его воспроизвести, как и последующие участки входа при дальнейшем сдвиге окна контекста. Аналогичная ситуация наблюдается и в двух других примерах. В табл. 1 можно проследить предметно, что точность GPT-2 практически не изменяется с увеличением длины последовательности свыше 72 токенов.



(a)



(b)

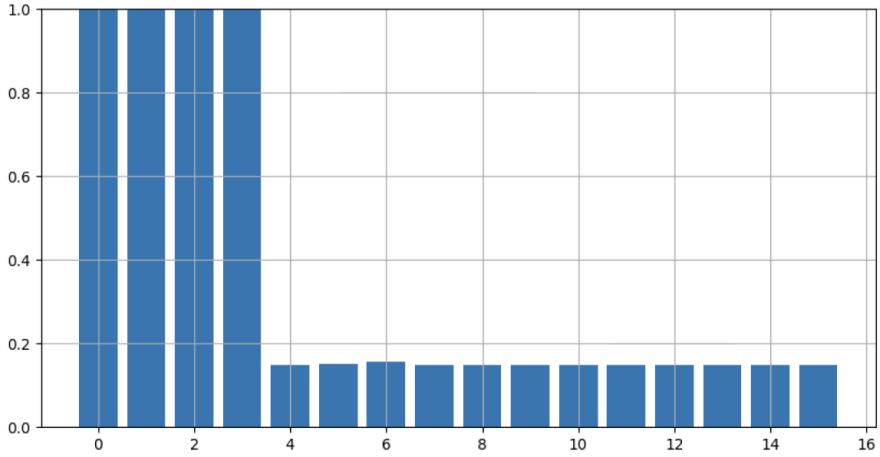
Рис. 1. График зависимости точности от длины последовательности для моделей GPT-2 (a) и Mamba (b) на задачах копирования, переворота и ассоциативного извлечения при размере скользящего окна в 36 токенов.

Табл. 1. Результаты моделей GPT-2 и Mamba на задачах копирования (copy), перевота (reverse) и ассоциативного извлечения (AR)

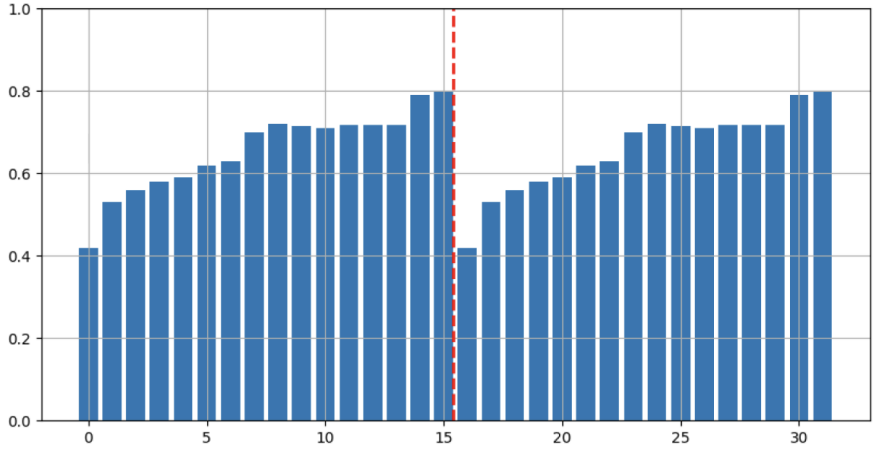
Длина	4	12	26	32	36	40	44	48	56	72	96	128	156
GPT-2, Copy	1	1	1	1	0.20	0.20	0.20	0.20	0.20	0.19	0.20	0.20	0.20
Mamba, Copy	1	1	1	1	1	1	1	0.95	0.85	0.81	0.73	0.66	0.56
GPT-2, Reverse	1	1	1	1	1	0.75	0.60	0.48	0.20	0.17	0.17	0.17	0.17
Mamba, Reverse	1	1	1	1	1	1	1	0.94	0.86	0.79	0.69	0.62	0.54

GPT-2, Associative retrieval	1	1	1	1	0.20	0.19	0.19	0.19	0.20	0.20	0.20	0.20	0.20
Mamba, Associative retrieval	1	1	1	1	1	1	0.95	0.93	0.90	0.87	0.82	0.75	0.60

Модель Mamba за счет своей рекуррентной архитектуры потенциально способна обрабатывать бесконечно длинные зависимости и сохранять их в своем внутреннем представлении, благодаря чему точность ее предсказаний почти не падает с ростом длины.



(a)



(b)

Рис. 2. График зависимости точности от номера токена в последовательности для модели GPT-2 (a) и GPT-X (b) на задаче переворота при размере скользящего окна в 8 токенов (a) и 32 токена (b).

На рис. 2 представлены графики распределения точности предсказания в задаче переворота для двух моделей: классической трансформерной архитектуры GPT-2 и экспериментальной модели GPT-X неизвестной структуры. Как и ожидалось, модель GPT-2 продемонстрировала характерное поведение: при выходе токенов за пределы окна внимания точность предсказаний снижается до 15%, в соответствии с табл. 2. Это подтверждает ограниченность памяти трансформера фиксированной длиной окна контекста.

В случае модели GPT-X ситуация иная: точность остается высокой на всем протяжении последовательности, включая те токены, которые по идее должны выходить за пределы обозреваемого окна. На графике красной пунктирной линией отмечена граница, после которой поведение модели изменяется – это может указывать на наличие в ее архитектуре механизма обработки входа блоками с сохранением промежуточного состояния. Такой эффект может быть достигнут за счет встроенной памяти или оптимизации скрытых представлений, как это реализовано, например, в некоторых вариантах рекуррентных трансформеров [14] или state-space моделей.

Поведение GPT-X указывает на наличие у нее внутренней памяти, превышающей по емкости окно стандартного самовнимания. Это делает модель особенно интересной для дальнейшего анализа: требуется исследовать характер хранения информации, ее объем, механизм извлечения и устойчивость к увеличению длины входа. Такие особенности могут иметь важное значение при разработке архитектур с расширенными возможностями долговременного контекста.

Табл. 2. Результаты моделей GPT-2 и GPT-X на задаче переворота (reverse) при длине контекстного окна в 8 и 32 токена соответственно.

Позиция токена во входной посл-	Точность, GPT-X	Точность, GPT-2	Позиция токена во входной посл-ти	Точность, GPT-X

ти				
0	0.42	1	16	0.42
1	0.53	1	17	0.53
2	0.56	1	18	0.56
3	0.58	1	19	0.58
4	0.59	0.15	20	0.59
5	0.62	0.151	21	0.62
6	0.63	0.157	22	0.63
7	0.7	0.15	23	0.7
8	0.72	0.15	24	0.72
9	0.715	0.15	25	0.715
10	0.71	0.15	26	0.71
11	0.717	0.15	27	0.717
12	0.717	0.15	28	0.717
13	0.718	0.15	29	0.718
14	0.79	0.15	30	0.79
15	0.8	0.15	31	0.8

ЗАКЛЮЧЕНИЕ

Представлен инструмент для быстрой диагностики архитектур больших языковых моделей в аспекте их способности обрабатывать длинные контексты. Актуальность задачи связана с тем, что эффективность современных больших языковых моделей определяется не только качеством генерации, но и глубиной внутренней памяти. Полноценное обучение каждой новой архитектуры требует

значительных ресурсов, поэтому необходимы методы раннего и дешевого тестирования.

Предложен диагностический инструмент на основе синтетических задач – задачи копирования, задачи переворота и задачи ассоциативного извлечения. Эти тесты позволяют оценить объем внутренней памяти модели и проследить, на какой длине последовательности сохраняется приемлемая точность. Каждая задача вводит свой уровень сложности: копирование проверяет удержание последовательности в памяти, переворот – способность к перестановке участков входа, а ассоциативное извлечение – выборочное обращение к элементам памяти.

Проведено сравнение архитектур GPT-2, Mamba и GPT-X в сопоставимых условиях. Mamba продемонстрировала наилучшие результаты на всех задачах, устойчиво работая с последовательностями, длина которых превышает размер контекстного окна. GPT-2 показала ожидаемые результаты, быстро теряя точность при увеличении длины входа. GPT-X заняла промежуточное положение, что может быть связано с внутренней блочной обработкой данных в этой сети.

Предложенный подход продемонстрировал эффективность при сравнении архитектур по их способности к долговременному хранению информации и может быть использован как инструмент предварительной оценки и выбора наиболее перспективных из них.

СПИСОК ЛИТЕРАТУРЫ

1. *Kaplan J., McCandlish S., Henighan T.*, et al. Scaling Laws for Neural Language Models // arXiv preprint arXiv:2001.08361. 2020. https://doi.org/10.48550/arXiv.2001.08361

2. *Brown T., Mann B., Ryder N.*, et al. Language Models are Few- Shot Learners // Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 1877-1901. https://doi.org/10.5555/3495724.3495883

3. *Beltagy I., Peters M. E., Cohan A.* Longformer: The Long- Document Transformer // arXiv preprint arXiv:2004.05150. 2020. https://doi.org/10.48550/arXiv.2004.05150

4. *Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I.* Language Models are Unsupervised Multitask Learners // OpenAI. 2019.

5. Common Crawl Foundation. Common Crawl dataset. https://commoncrawl.org

6. *Gu A., Goel K., Ré C.* Efficiently Modeling Long Sequences with Structured State Spaces // International Conference on Learning Representations (ICLR). 2022.

7. *Gao L., Biderman S., Black S.*, et al. The Pile: An 800 GB Dataset of Diverse Text for Language Modeling // arXiv preprint arXiv:2101.00027. 2020. https://doi.org/10.48550/arXiv.2101.00027

8. *Eldan R., Li Y.* TinyStories: How Small Can Language Models Be and Still Speak Coherent English? // arXiv preprint arXiv:2305.07759. 2023. https://doi.org/10.48550/arXiv.2305.07759

9. *Dao T.* FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness // Advances in Neural Information Processing Systems (NeurIPS). 2022. Vol. 35. P. 16344-16359. https://doi.org/10.48550/arXiv.2205.14135

10. *Gu A., Goel K., Dao T.*, et al. Mamba: Linear-Time Sequence Modeling with Selective State Spaces // International Conference on Learning Representations (ICLR). 2024.

11. *Kwon W., Lee S., Li S., Zaharia M., Zhang H., Stoica I., Sheng Y., Crichton W., Xie S., Gonzalez J.* Efficient Memory Management for Large Language Model Inference with KV-Caching // arXiv preprint arXiv:2309.06180. 2023. https://doi.org/10.48550/arXiv.2309.06180

12. *Vaswani A., Shazeer N., Parmar N.*, et al. Attention Is All You Need // Advances in Neural Information Processing Systems (NIPS). 2017. Vol. 30. P. 5998–6008.
https://doi.org/10.5555/3295222.3295349

13. *Tay Y., Bahri D., Metzler D.*, et al. Long Range Arena: A Benchmark for Efficient Transformers // arXiv preprint arXiv:2011.04006. 2020. https://doi.org/10.48550/arXiv.2011.04006

14. *Bulatov A., Kuratov Y., Burtsev M.* Recurrent Memory Transformer // Advances in Neural Information Processing Systems. 2022. Vol. 35. P. 11079-11091. https://doi.org/10.48550/arXiv.2207.06881

A TOOL FOR RAPID DIAGNOSTICS OF MEMORY IN NEURAL NETWORK ARCHITECTURES OF LANGUAGE MODELS

P. A. Gavrikov[1] [0009-0009-6175-2883], A. K. Usmanov[2] [0009-0006-2535-1052], D. Revaev[3] [0009-0008-1630-1912], S. N. Buzykanov[4] [0009-0005-3405-0705]

[1-4]*Moscow Institute of Physics and Technology, Moscow, Russia*

[1]gavrikov.pa@phystech.edu, [2]usmanov.ak@phystech.edu, [3]revaev.d@phystech.edu, [4]bsn1977@mail.ru

Abstract

Large Language Models (LLMs) have evolved from simple n-gram systems to modern universal architectures; however, a key limitation remains the quadratic complexity of the self-attention mechanism with respect to input sequence length. This significantly increases memory consumption and computational costs, and with the emergence of tasks requiring extremely long contexts, creates the need for new architectural solutions. Since evaluating a proposed architecture typically requires long and expensive full-scale training, it is necessary to develop a tool that allows for a rapid preliminary assessment of a model's internal memory capacity.

This paper presents a method for quantitative evaluation of the internal memory of neural network architectures based on synthetic tests that do not require large data corpora. Internal memory is defined as the amount of information a model can reproduce without direct access to its original inputs.

To validate the approach, a software framework was developed and tested on the GPT-2 and Mamba architectures. The experiments employed copy, inversion, and associative retrieval tasks. Comparison of prediction accuracy, error distribution, and computational cost enables a fast assessment of the efficiency and potential of various LLM architectures.

Keywords: *large language models; neural network architecture; internal memory; long-term information retention; sequence processing; functional memory measurement; architecture comparison.*

REFERENCES

1. *Kaplan J., McCandlish S., Henighan T.*, et al. Scaling Laws for Neural Language Models // arXiv preprint arXiv:2001.08361. 2020. https://doi.org/10.48550/arXiv.2001.08361

2. *Brown T., Mann B., Ryder N.*, et al. Language Models are Few- Shot Learners // Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 1877-1901. https://doi.org/10.5555/3495724.3495883

3. *Beltagy I., Peters M. E., Cohan A.* Longformer: The Long- Document Transformer // arXiv preprint arXiv:2004.05150. 2020. https://doi.org/10.48550/arXiv.2004.05150

4. *Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I.* Language Models are Unsupervised Multitask Learners // OpenAI. 2019.

5. Common Crawl Foundation. Common Crawl dataset. https://commoncrawl.org

6. *Gu A., Goel K., Ré C.* Efficiently Modeling Long Sequences with Structured State Spaces // International Conference on Learning Representations (ICLR). 2022.

7. *Gao L., Biderman S., Black S.*, et al. The Pile: An 800 GB Dataset of Diverse Text for Language Modeling // arXiv preprint arXiv:2101.00027. 2020. https://doi.org/10.48550/arXiv.2101.00027

8. *Eldan R., Li Y.* TinyStories: How Small Can Language Models Be and Still Speak Coherent English? // arXiv preprint arXiv:2305.07759. 2023. https://doi.org/10.48550/arXiv.2305.07759

9. *Dao T.* FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness // Advances in Neural Information Processing Systems (NeurIPS). 2022. Vol. 35. P. 16344-16359. https://doi.org/10.48550/arXiv.2205.14135

10. *Gu A., Goel K., Dao T.*, et al. Mamba: Linear-Time Sequence Modeling with Selective State Spaces // International Conference on Learning Representations (ICLR). 2024.

11. *Kwon W., Lee S., Li S., Zaharia M., Zhang H., Stoica I., Sheng Y., Crichton W., Xie S., Gonzalez J.* Efficient Memory Management for Large Language Model Inference with KV-Caching // arXiv preprint arXiv:2309.06180. 2023. https://doi.org/10.48550/arXiv.2309.06180

12. *Vaswani A., Shazeer N., Parmar N.*, et al. Attention Is All You Need // Advances in Neural Information Processing Systems (NIPS). 2017. Vol. 30. P. 5998–6008.
https://doi.org/10.5555/3295222.3295349

13. *Tay Y., Bahri D., Metzler D.*, et al. Long Range Arena: A Benchmark for Efficient Transformers // arXiv preprint arXiv:2011.04006. 2020.
https://doi.org/10.48550/arXiv.2011.04006

14. *Bulatov A., Kuratov Y., Burtsev M.* Recurrent Memory Transformer // Advances in Neural Information Processing Systems. 2022. Vol. 35. P. 11079-11091.
https://doi.org/10.48550/arXiv.2207.06881

СВЕДЕНИЯ ОБ АВТОРАХ

ГАВРИКОВ Павел Андреевич – студент 1-го курса магистратуры ФРКТ МФТИ. Область научных интересов: Большие языковые модели, генеративные нейронные сети, теория компиляторов. Число научных публикаций: 2.

Pavel Andreevich GAVRIKOV – First-year Master's student, Department of Radio Engineering and Computer Technology (DREC), Moscow Institute of Physics and Technology (MIPT). Research interests: large language models, generative neural networks, compiler theory. Number of scientific publications: 2.

email: gavrikov.pa@phystech.edu

ORCID: 0009-0009-6175-2883

УСМАНОВ Азамат Комилжон угли – студент 1-го курса магистратуры ФПМИ МФТИ. Область научных интересов: обработка естественного языка, авторегрессионные модели, оптимизация больших языковых моделей. Число научных публикаций: 1.

Azamat Komiljon ugli USMANOV – First-year Master's student, Department of Applied Mathematics and Computer Science (DAPCS), Moscow Institute of Physics and Technology (MIPT). Research interests: natural language processing, autoregressive models, optimization of large language models. Number of scientific publications: 1.

email: usmanov.ak@phystech.edu

ORCID: 0009-0006-2535-1052

РЕВАЕВ Дмитрий аспирант 1-го года, Кафедра Интеллектуальных Систем ФПМИ МФТИ. Область научных интересов: большие языковые модели, обучение с подкреплением, теория информации.

Dmitriy REVAYEV – First-year PhD student, The Intelligent Systems Department, The Phystech School of Applied Mathematics and Informatics, Moscow Institute of Physics and Technology (MIPT). Research Interests: Large Language Models, Reinforcement Learning, Information Theory.

email: revaev.d@phystech.edu

ORCID: 0009-0008-1630-1912

БУЗЫКАНОВ Сергей Николаевич – д. т. н., профессор кафедры радиотехнических систем НИУ «МЭИ», доцент кафедры МТТК МФТИ, ведущий инженер Huawei. Область научных интересов: цифровая обработка сигналов, системы обработки информации, системы искусственного интеллекта и большие языковые модели. Число научных публикаций – более 70.

Sergey Nikolaevich BUZYKANOV – Doctor of Engineering, Professor of the Radio Engineering Systems Department at National Research University "MPEI," Associate Professor of the MTTC Department at MIPT, and a Huawei Lead Engineer. Research interests: digital signal processing, data processing systems, artificial intelligence systems and large language models. Over 70 scientific publications.

email: bsn1977@mail.ru

ORCID: 0009-0005-3405-0705

Материал поступил в редакцию 9 ноября 2025 года