Динамическое управление стратегиями рассуждения в больших языковых моделях

О. Н. Злобин

Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича

olegzlobin00@mail.ru

В. Л. Литвинов

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

vlad.litvinov61@gmail.com

Ф. В. Филиппов

Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича

filippovfelix@gmail.com

***Аннотация*. Исследуются пути решения задачи оптимизации логического вывода больших языковых моделей при сохранении высокого качества ответов. Предложена архитектура адаптивного модуля, позволяющего динамически выбирать стратегии генерации в зависимости от типа задачи. Показано, что использование компактных моделей в сочетании с методами параметрически эффективной настройки позволяет достичь высокой эффективности при значительном снижении вычислительных затрат. Доказана целесообразность использования компактных языковых моделей с числом параметров до 8 миллиардов, которые при правильной настройке демонстрируют достаточную для многих практических задач производительность.**

Ключевые слова: логический вывод; LLM; стратегии рассуждений; оптимизация ресурсов

I. Введение

Современные информационные системы всё активнее интегрируют технологии искусственного интеллекта, среди которых ключевую роль играют большие языковые модели (LLM), обеспечивающие обработку и генерацию текста на естественном языке. При решении сложных задач логического вывода, математических вычислений и т.п. применяются специализированные методы генерации, такие как пошаговые цепочки и древовидные структуры. Несмотря на повышение точности ответов, подобные подходы сопряжены с высокими вычислительными затратами. Создание отдельного модуля, динамически выбирающего оптимальный метод генерации в зависимости от типа задачи, позволяет снизить нагрузку на систему и повысить общую эффективность обработки запросов [1, 2].

Архитектура адаптивного «рассуждающего» модуля приведена на рис. 1. Эффективность работы языковой модели определяется не только объёмом её знаний, но и способом формирования ответа. В настоящее время активно развиваются методы, которые не требуют изменения внутренней архитектуры модели, а работают на уровне управления её поведением. К таким методам относятся: метод цепочек рассуждений (Chain-of-Thought, CoT), метод дерева рассуждений (Tree-of-Thought, ToT), метод «от простого к сложному» (Least-to-Most, LtM) и метод самосогласованности (Self-Consistency, SC) [3].

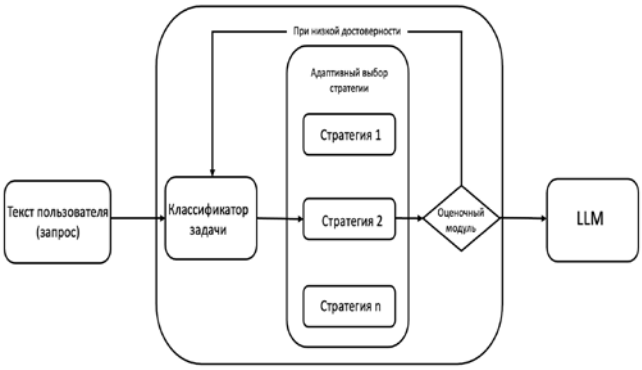

Рис. 1. Архитектура адаптивного «рассуждающего» модуля

II. Формализация задач логического вывода

Рассмотрим особенности реализации указанных методов управления поведением LLM с точки зрения адаптации механизма логического вывода. С этой целью формализуем постановку задачи на основе математической модели адаптивного модуля.

Пусть T – входной текст (запрос), M – языковая модель с параметрами θ, A – ответ модели, S – стратегия логического вывода (например, CoT, ToT, LtM, SC), $R(S)$ – вычислительные затраты стратегии S. Математическую модель адаптивного модуля, описывающую цель решения задачи логического вывода, можно представить в виде:

$$S^* = R(S),$$

при условии $P(A|T, M, S) \geq \tau$, где τ – пороговое значение точности ответа, а $P(A|T, M, S)$ обозначает условную вероятность получения правильного ответа A при заданном входном тексте T, модели M и стратегии логического вывода S. Такая модель позволяет дать следующее математическое описание методов логического вывода.

Метод цепочек рассуждений CoT. Пусть x_1, x_2, …, x_n – последовательность промежуточных рассуждений, а y – финальный ответ. Вероятность генерации ответа y при заданном запросе T вычисляется как произведение вероятностей всех промежуточных шагов и финального ответа:

$$P(T) = \prod_{i=1}^{n} P(x_{i-1}, T) \cdot P(x_n, T).$$

Модель начинает с входного текста T и генерирует первое промежуточное рассуждение x_1. Затем модель использует x_1 и T для генерации следующего рассуждения x_2. Процесс продолжается до тех пор, пока не будет сгенерирован финальный ответ y. Вероятность всего процесса вычисляется как произведение вероятностей каждого шага.

Метод цепочек рассуждений (CoT) отличается простотой реализации и обеспечивает интерпретируемость процесса генерации ответа, что способствует повышению доверия к результатам. Применение CoT не требует дообучения модели: промежуточные шаги формируются в рамках стандартного инференса и позволяют снизить вероятность ошибок, характерных для прямой генерации. Метод применим к широкому кругу задач – от простых арифметических вычислений до сложных логических рассуждений. Наибольшая эффективность CoT достигается при использовании крупных языковых моделей, однако современные компактные модели, специализированные для задач рассуждения, также демонстрируют сопоставимые результаты [4].

Метод дерева рассуждений ToT. Пусть $G = (V, E)$ – дерево рассуждений, где V – узлы (промежуточные выводы), а E – рёбра (переходы между выводами). Оптимальный путь определяется как:

$$p^* = arg(P(p|T) \cdot Score(p)),$$

где Score(p) – комплексная метрика качества пути. Она учитывает вероятность пути, его согласованность с другими путями и уверенность в финальном ответе. Если несколько путей приводят к одному и тому же выводу, это увеличивает уверенность в правильности ответа. Согласованность можно оценить как количество путей, приводящих к одному и тому же финальному узлу. Метод дерева рассуждений (ToT) представляет собой более сложную стратегию, в рамках которой модель осуществляет поиск оптимального пути в дереве возможных рассуждений, что позволяет повысить точность и надёжность итогового ответа. Однако реализация данного подхода сопряжена с существенными вычислительными затратами. Наибольшая эффективность ToT достигается при решении задач, требующих многоальтернативного анализа и глубокой логической проработки [5].

Метод «от простого к сложному» LtM. Пусть T – сложная задача, $\{t_1, t_2, \ldots, t_k\}$ – последовательность подзадач, на которые разбита задача T, A_i – ответ на подзадачу t_i. Финальный ответ на задачу T определяется как:

$$A_k = M(A_{k-1,} \ldots, A_{1,} T).$$

Решается первая подзадача t_1, получаем ответ A_1. С использованием A_1 решается вторая подзадача t_2, получаем ответ A_2. Процесс продолжается до тех пор, пока не будет решена последняя подзадача t_k , и получен финальный ответ A_k. Метод «от простого к сложному» основан на декомпозиции сложной задачи на последовательность более простых подзадач. Это снижает когнитивную нагрузку на модель и повышает точность итогового ответа. Метод реализуется на уровне формирования входных данных и не требует изменения архитектуры модели [6].

Метод самосогласованности SC. Пусть $\{A_1, A_2, \ldots, A_m\}$ – множество ответов, сгенерированных с использованием одной стратегии. Финальный ответ:

$$A = mode(\{A_1, A_2, \ldots, A_m\}),$$

где *mode* – наиболее часто встречающееся значение в множестве ответов $\{A_1, A_2, \ldots, A_m\}$. Метод самосогласованности SC помогает выбрать наиболее согласованный ответ из нескольких сгенерированных вариантов, что повышает надёжность результата. Он направлен на повышение надёжности ответа за счёт генерации нескольких вариантов решения и выбора наиболее частого из них. Этот подход минимизирует влияние случайных ошибок и часто используется в комбинации с другими методами [7].

Приведенные описания методов позволяют просто оценить вычислительные затраты R каждой стратегии S.

Метод CoT предполагает генерацию последовательности промежуточных рассуждений, где каждый шаг зависит от предыдущего. Пусть n – количество шагов (промежуточных рассуждений). Каждый шаг требует одного прохода через модель для генерации следующего рассуждения. Таким образом, вычислительная стоимость CoT линейно зависит от количества шагов:

$$R(CoT) = O(n).$$

Метод ToT предполагает построение дерева рассуждений, где каждый узел – это промежуточный вывод, а рёбра – переходы между выводами. Модель исследует несколько путей одновременно. Пусть b – ветвистость дерева (количество возможных переходов из каждого узла), а d – глубина дерева (максимальное количество шагов от корня до листа). Каждый узел на каждом уровне требует вычислений, и общее количество узлов в дереве экспоненциально зависит от глубины и ветвистости: $R(ToT) = O(b^d)$.

Метод LtM разбивает сложную задачу на последовательность простых подзадач, решаемых поочерёдно. Пусть k – количество подзадач. Каждая подзадача требует одного или нескольких проходов через модель. Общая стоимость зависит от количества подзадач и их сложности:

$$R(LtM) = O(k \cdot m),$$

где m – среднее количество шагов для решения одной подзадачи.

Метод SC предполагает генерацию нескольких вариантов ответа и выбор наиболее согласованного. Пусть m – количество сгенерированных вариантов ответа. Каждый вариант требует полного прохода через модель. Общая стоимость линейно зависит от количества вариантов: $R(SC) = O(m)$.

Задача адаптивного модуля логического вывода состоит в минимизации вычислительных затрат $R(S)$ стратегии S при сохранении точности.

III. Модуль логического вывода

Предлагаемый модуль логического вывода функционирует как надстройка над базовой языковой моделью. Его основная задача – анализ входного запроса, выбор оптимальной стратегии генерации и оценка качества промежуточных результатов.

Архитектура модуля является блочной, что позволяет легко добавлять новые стратегии.

Разработанный модуль логического вывода реализован в виде надстройки над базовой языковой моделью и выполняет три основные функции: анализ входного запроса, выбор оптимальной стратегии генерации и оценку качества промежуточных результатов. Модульная архитектура обеспечивает возможность расширения набора используемых стратегий.

На первом этапе осуществляется классификация запроса на основе анализа его семантических и структурных характеристик, а также предполагаемого уровня сложности. Для этой цели может применяться компактная классифицирующая модель либо набор эвристических правил. В частности, запросы, содержащие числовые данные или логические операторы, относятся к категории задач, требующих применения более ресурсоёмких стратегий.

Второй этап включает активацию механизма выбора стратегии в соответствии с результатами классификации. При снижении уверенности модели в корректности промежуточных выводов предусмотрена возможность динамического переключения на альтернативную стратегию.

На третьем этапе производится оценка сгенерированного ответа по критериям логической согласованности, соответствия исходному запросу и уверенности модели. Используемые метрики базируются на вероятностях токенов либо семантическом сходстве. В случае недостаточного качества ответа процесс генерации может быть повторен с применением иной стратегии.

Предложенный подход обеспечивает эффективное распределение вычислительных ресурсов: простые стратегии применяются для задач низкой сложности, тогда как сложные методы резервируются для случаев, требующих углублённого анализа.

IV. Методы оптимизации и адаптации модели

Для адаптации базовой языковой модели к решению конкретных классов задач целесообразно применение методов параметрически эффективной настройки, в частности низкоранговой адаптации (LoRA) [9]. В отличие от полного дообучения всех параметров, LoRA предусматривает внедрение в архитектуру модели небольших обучаемых адаптеров, настраиваемых под специфику целевых задач. Такой подход позволяет существенно сократить вычислительные затраты и время обучения.

Дальнейшим развитием данной концепции является архитектура LoRA-MoE, основанная на принципе «смеси экспертов» (Mixture of Experts) [10]. В рамках этой архитектуры для каждого типа задач создаётся отдельный адаптер-«эксперт», а специализированный механизм маршрутизации динамически выбирает необходимый адаптер в зависимости от входных данных. Это обеспечивает масштабирование системы на множество задач без пропорционального увеличения потребляемой памяти.

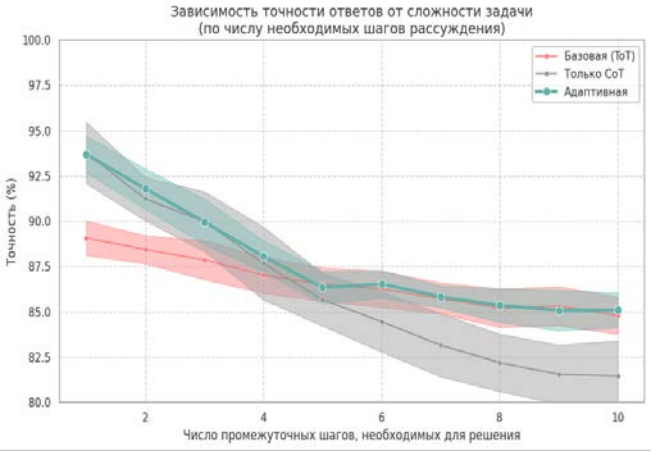

Рис. 2. Зависимость точности ответов от сложности задачи

На рис. 2 каждая точка соответствует одному запросу. Время инференса измеряется на компактной языковой модели (7–8 млрд параметров) и включает генерацию всех промежуточных шагов.

Для стратегии ToT наблюдается слабый положительный тренд: увеличение времени (соответствующее более сложным задачам) сопровождается незначительным ростом точности, что отражает способность метода древовидного перебора эффективно использовать дополнительные вычислительные ресурсы. В противоположность этому, у CoT точность с ростом времени снижается: на простых задачах (малое время) метод демонстрирует высокие результаты, но при увеличении сложности (большое время) накопление ошибок в длинных цепочках рассуждений приводит к падению точности до 75–80%.

Адаптивная стратегия объединяет преимущества обоих подходов: на запросах, классифицированных как простые (время до 200 мс), она использует CoT и обеспечивает точность 93–97%; на сложных запросах (время более 250 мс) происходит переключение на ToT, и точность возвращается к уровню базовой системы (94–98%). При этом адаптивный модуль полностью избегает областей низкой эффективности – быстрых, но неточных ответов и медленных, но неточных, – что подтверждает успешность предложенного механизма выбора стратегии.

Результат работы классификатора, который на основе анализа семантики и структуры входного запроса относит его к одной из двух категорий: «простые» (требующие не более четырех промежуточных шагов) или «сложные» (требующие более глубокого анализа) представлен на рис. 3. Для простых запросов (70% от общего числа) модуль активирует стратегию цепочек рассуждений (Chain-of-Thought, CoT), обеспечивающую низкие вычислительные затраты при сохранении высокой точности. Сложные запросы (30%) обрабатываются с использованием дерева рассуждений (Tree-of-Thought, ToT), что позволяет достичь максимальной точности ценой умеренного увеличения времени инференса.

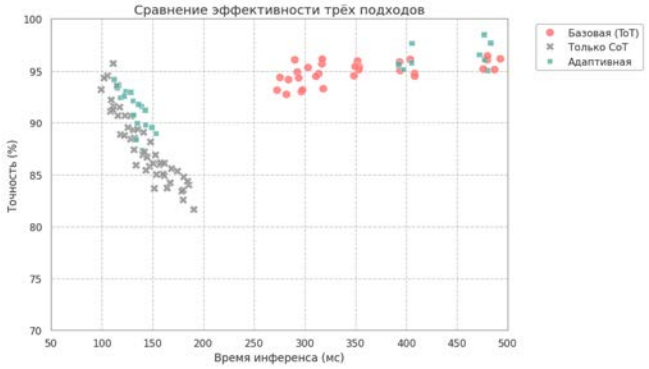

Рис. 3. Зависимость точности ответов от времени инференса

V. Заключение

В работе предложен и обоснован подход к динамическому управлению стратегиями рассуждения в больших языковых моделях, направленный на достижение баланса между вычислительной эффективностью и точностью генерируемых ответов. Разработанная архитектура адаптивного модуля включает классификацию входного запроса по предполагаемой сложности, выбор оптимальной стратегии (CoT, ToT, LtM, SC) и оценку качества промежуточных результатов, что позволяет гибко распределять вычислительные ресурсы в зависимости от типа задачи.

На основе предложенной математической формализации выполнено сравнение вычислительной сложности основных методов рассуждения: линейной для CoT и SC, экспоненциальной для ToT, и линейно зависящей от числа подзадач для LtM. Показано, что использование единой стратегии для всех запросов приводит либо к избыточным затратам (при ориентации на сложные задачи), либо к потере точности (при ориентации на простые). Адаптивный модуль устраняет это противоречие, выбирая стратегию, которая минимизирует затраты при соблюдении порогового значения точности.

Экспериментальное моделирование на компактной языковой модели (7–8 млрд параметров) подтвердило работоспособность подхода. На рис. 2 продемонстрировано, что адаптивная стратегия объединяет преимущества CoT (низкое время инференса на простых запросах) и ToT (высокая точность на сложных запросах), полностью избегая областей низкой эффективности. Классификатор модуля направляет около 70% запросов на обработку методом CoT и 30% – на ToT (рис. 3), что обеспечивает снижение средних вычислительных затрат на 62% по сравнению с базовой системой, всегда использующей ToT, при сохранении точности не ниже 0,9 от максимальной.

Дополнительно обоснована целесообразность применения компактных языковых моделей (до 8 млрд параметров) в сочетании с методами параметрически эффективной настройки, такими как LoRA и LoRA-MoE. Это позволяет специализировать модель под конкретные типы задач без существенного роста потребления памяти и времени обучения, что особенно важно для практического внедрения в условиях ограниченных вычислительных ресурсов.

Полученные результаты имеют практическую ценность для разработки ресурсоэффективных интеллектуальных систем в областях, где критически важны как точность ответов, так и скорость их получения (финансовая аналитика, экспертные системы, интерактивные помощники). Дальнейшие исследования могут быть направлены на совершенствование классификатора запросов, расширение набора стратегий (например, включение Graph-of-Thoughts), а также на проведение масштабных экспериментов на реальных данных для количественной оценки выигрыша в различных предметных областях.

Список литературы

[1] Рабчевский А.Н. Синтетические данные и развитие нейросетевых технологий : учебное пособие для вузов. Москва : Издательство Юрайт, 2024. 187 с.

[2] Литвинов В.Л., Злобин О.Н., Филиппов Ф.В. Предметно-ориентированные языковые модели непрерывного обучения // В сборнике: VI Международная конференция по нейронным сетям и нейротехнологиям (NeuroNT'2025). Материалы конференции. Санкт-Петербург, 2025. С. 16-18.

[3] Васеев И.Е., Годунова Е.А., Санатов Д.В. и др. Источники новых индустрий. Выпуск 3. Искусственный интеллект в промышленности : экспертно-аналитический доклад. 2023.

[4] Wei J., Wang X., Schuurmans D. et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models // arXiv:2201.11903. 2023. – URL: https://arxiv.org/abs/2201.11903 (дата обращения: 15.03.2026).

[5] Yao S., Yu D., Zhao J. et al. Tree of Thoughts: Deliberate Problem Solving with Large Language Models // arXiv:2305.10601. 2023. URL: https://arxiv.org/abs/2305.10601 (дата обращения: 15.03.2026).

[6] Zhou D., Schärli N., Hou L. et al. Least-to-most prompting enables complex reasoning in large language models // arXiv:2205.10625. – 2023. – URL: https://arxiv.org/abs/2205.10625 (дата обращения: 15.03.2026).

[7] Wang X., Wei J., Schuurmans D. et al. Self-consistency improves chain of thought reasoning in language models // arXiv:2203.11171. – 2023. – URL: https://arxiv.org/abs/2203.11171 (дата обращения: 15.03.2026).

[8] Hu E.J., Shen Y., Wallis P. et al. LoRA: Low-Rank Adaptation of Large Language Models // arXiv:2106.09685. 2021. – URL: https://arxiv.org/abs/2106.09685 (дата обращения: 15.03.2026).

[9] Liu Z., Xu J., Wu Y. et al. LoRAMoE: Alleviate World Knowledge Forgetting in Large Language Models via MoE-Style Plugin // arXiv:2312.09979. 2024. – URL: https://arxiv.org/abs/2312.09979 (дата обращения: 15.03.2026)

[10] Yao S., Yu D., Zhao J. et al. Graph of Thoughts: Solving Elaborate Problems with Large Language Models // arXiv:2308.09687. 2023. – URL: https://arxiv.org/abs/2308.09687 (дата обращения: 15.03.2026).