Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный университет имени М. В. Ломоносова»

На правах рукописи

Малоян Нарек Гагикович

Разработка методов оценки и повышения устойчивости больших языковых моделей к вариациям входных последовательностей

Специальность 2.3.5 — «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:
доктор технических наук
Намиот Дмитрий Евгеньевич

Москва — 2026

Оглавление

Стр.

Стр.

Стр.

Стр.

Стр.

Введение

Современные большие языковые модели (LLM) решают широкий спектр задач обработки естественного языка. По мере встраивания LLM в производственные системы растет и число точек, через которые возможно алгоритмическое воздействие на модель: инъекции промптов, троянские закладки, манипуляции с автоматическими метриками качества. Особую роль среди таких метрик играет архитектура LLM-as-a-Judge, где языковая модель оценивает ответы других моделей. Компрометация модели-оценщика искажает результаты сравнения и подрывает воспроизводимость экспериментов.

Поиск вредоносных инструкций представляет собой задачу оптимизации на дискретном пространстве текстов. Злоумышленники перебирают обходные траектории (*jailbreaks*) до тех пор, пока модель не выдаст запрещенную информацию. Эвристические фильтры на нескольких уровнях не решают проблему. Троянские закладки работают иначе: на этапе обучения в данные внедряется триггер, который формирует скрытый аттрактор в пространстве параметров. Восстановление такого триггера по весам сети есть обратная задача с плохой обусловленностью. На соревновании Trojan Detection Challenge 2023 полнота восстановления истинных триггеров составила 0,17 и практически не отличалась от базового уровня. Агентские архитектуры усугубляют ситуацию: когда LLM получает доступ к внешним инструментам, текстовая уязвимость транслируется в пространство исполняемых действий. Последствия атаки в таком случае выходят за информационные рамки.

Экспериментальная оценка количественно подтверждает уязвимость архитектуры LLM-as-a-Judge. На наборе MT-Bench и в задачах соревнования Kaggle «LLMs: You Can't Please Them All» успешность инъекционных атак на модели-оценщики достигала 73,8% по метрике ASR на модели Gemma-3-4B-Instruct при применении адаптивного алгоритма ASA. Встроенные ограничения моделей-оценщиков не выдерживают целенаправленных воздействий.

Композиционная природа агентских систем порождает отдельный класс рисков. Протокол Model Context Protocol (MCP) унифицирует взаимодействие LLM-агента с внешними инструментами и тем самым расширяет пространство его состояний. Инъекция в контекст вызова инструмента позволяет перехватить управление (*prompt injection to tool hijacking*). С алгоритмической точки зрения

проблема устойчивости отдельной модели масштабируется до графа вычислений целой системы: входные последовательности формируются динамически из разных источников, и атака через внешний канал становится частным случаем состязательной пертурбации входа, способной спровоцировать несанкционированное выполнение кода.

Все описанные уязвимости имеют общую природу: выученное моделью отображение чувствительно к малым вариациям входа в дискретном пространстве токенов. Пертурбации дестабилизируют скрытые представления, и модель теряет заданные свойства. Встраивание LLM в сложные агентские системы усиливает остроту проблемы и делает разработку строгого аппарата обеспечения робастности актуальной научной задачей.

Степень разработанности темы исследования. Основы анализа робастности нейросетевых архитектур заложены в работах Goodfellow et al. и Madry et al. применительно к области компьютерного зрения. Перенести этот аппарат на LLM напрямую затруднительно из-за дискретной природы пространства токенов и авторегрессионного характера вывода. Для текстовой области Zou et al. предложили метод GCG, генерирующий универсальные состязательные суффиксы, Wallace et al. показали возможность построения универсальных триггеров для управления генерацией, Perez et al. систематизировали атаки типа инъекции запроса. Carlini et al. показали возможность извлечения фрагментов обучающих данных непосредственно из параметров модели. Вместе с тем формальный метрический аппарат для оценки робастности LLM развит недостаточно: имеющиеся метрики не учитывают дискретность пространства токенов и авторегрессионный характер генерации. Zheng et al. предложили архитектуру LLM-as-a-Judge для автоматической оценки качества ответов моделей. Она быстро получила распространение, но анализ ее устойчивости к целенаправленным состязательным атакам остается фрагментарным. Обнаружение троянских закладок изучалось в бенчмарках TDC2023 и TrojAI. Количественная характеристика асимметрии между внедрением триггера и его обнаружением до настоящей работы оставалась открытым вопросом. Безопасность агентских систем и протоколов вроде MCP изучена мало. Greshake et al. описали атаки косвенной инъекции на LLM-интегрированные приложения и показали, что внешний контент способен перехватить управление агентом. При этом анализ архитектурных рисков протокола MCP и алгоритмы защиты на базе криптографической аттестации вызовов инструментов в известных публи-

кациях не рассматривались. Отсутствие комплексного подхода к безопасности агентских LLM-систем мотивировало настоящее исследование. К моменту начала работы в литературе не было: (1) формальной метрики устойчивости, учитывающей дискретность и авторегрессионность генерации LLM, (2) систематической методологии оценки уязвимости LLM-as-a-Judge, (3) алгоритмов защиты агентских систем от инъекций через протоколы взаимодействия с внешними инструментами.

Тематика диссертации соответствует паспорту специальности 2.3.5 «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей». В работе разработаны формальные модели, алгоритмы тестирования и программные средства оценки качества LLM-систем. Результаты относятся к следующим пунктам паспорта:

- **п. 1** (модели, методы и алгоритмы анализа, верификации и тестирования программных систем): разработана модель $R_{stab}(f)$, предложены методы и алгоритмы анализа уязвимостей LLM, реализовано автоматизированное тестирование безопасности на основе состязательных атак (Главы 2, 3, 4).
- **п. 4** (интеллектуальные системы машинного обучения, инструментальные средства): объект исследования – LLM как интеллектуальные системы машинного обучения, созданы фреймворки оценки их безопасности (Главы 1–5).
- **п. 10** (оценка качества, стандартизация и сопровождение программных систем): предложен комплекс метрик (ASR, TSR, Recall, REASR, R_{stab}) и методология количественной оценки качества LLM-систем в аспекте устойчивости и безопасности, обеспечивающая унификацию процедур тестирования (Главы 2, 4).

Работа охватывает несколько взаимосвязанных направлений:

- уязвимости LLM к атакам типа «инъекция запроса» в архитектурах LLM-as-a-Judge и в системах с многоуровневой защитой;
- ограничения существующих методов обнаружения троянских закладок;
- уязвимости агентских LLM-систем, использующих протокол MCP для взаимодействия с внешними инструментами, и программные средства защиты от перехвата управления;
- методы повышения устойчивости LLM к перечисленным угрозам и их экспериментальная оценка.

Объединение перечисленных направлений в одной работе обнажает общую структуру уязвимостей LLM и позволяет дать разработчикам конкретные рекомендации.

Объект исследования – большие языковые модели и системы на их основе: LLM-as-a-Judge и агентские LLM-системы с инструментальным доступом.

Предмет исследования – модели, методы, алгоритмы и программные средства оценки уязвимостей и повышения устойчивости LLM.

Целью работы – разработать комплекс моделей, методов и алгоритмов для оценки и повышения устойчивости больших языковых моделей к состязательным вариациям входных последовательностей и инъекционным воздействиям. Под вариациями входных последовательностей понимаются инъекции запроса (*prompt injection*), троянские закладки (*backdoor attacks*) и атаки на агентские LLM-системы.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. *Систематизация и классификация угроз.* Построить классификацию уязвимостей больших языковых моделей к атакам типа prompt injection, охватывающую специализированные архитектуры, такие как LLM-as-a-Judge и системы с многоуровневой защитой, и установить ограничения методов обнаружения троянских закладок.
2. *Разработка теоретических основ оценки.* Разработать формальную математическую модель оценки устойчивости ($R_{stab}(f)$) и согласованный набор эмпирических метрик, включающий базовые показатели ASR (*Attack Success Rate*), TSR (*Transfer Success Rate*), Recall и REASR (*Reverse-Engineered ASR*), учитывающих дискретную природу пространства токенов и авторегрессионный характер генерации.
3. *Построение методов состязательных атак.* Разработать адаптивные алгоритмы генерации состязательных атак (ASA) для тестирования уязвимостей LLM, включая атаки на системы LLM-as-a-Judge в условиях «черного ящика».
4. *Разработка и экспериментальная валидация методов защиты.* Предложить и экспериментально исследовать стратегии повышения устойчивости LLM, а именно комитеты гетерогенных моделей и многоуровневую фильтрацию, на репрезентативном наборе моделей основных архитектурных семейств с открытыми весами (Gemma,

Llama) и проприетарными (GPT, Claude), а также на наборах данных MT-Bench, TDC2023, SaTML CTF 2024.

5. *Программная реализация и апробация.* Создать программные средства для оценки безопасности LLM-систем, включая агентские системы на базе протокола MCP. Апробировать методы атак и защиты на бенчмарках TDC2023, SaTML CTF 2024 и в производственной среде ВИАСАТ TEX.

Научная новизна:

1. Предложена как метрика генеративной устойчивости формальная математическая модель оценки устойчивости больших языковых моделей ($R_{stab}(f)$), учитывающая дискретную природу пространства токенов и авторегрессионный характер генерации, основанная на ограниченных мерах расхождения, в частности дивергенции Йенсена-Шеннона, между вероятностными распределениями на каждом шаге генерации при малых возмущениях входа. Показана согласованность метрики с естественным отношением усиления защиты, где монотонность выступает условием корректности определения. Теоретические границы уязвимости доказаны для детерминированного оператора решения $R_{class}(h)$. Для задач с дискретным выходом, включая LLM-as-a-Judge, эмпирическая оценка $\widehat{R}_{emp}(f) = 1 - \text{ASR}$ может рассматриваться как практическая оценка устойчивости относительно фиксированного семейства атак $\mathcal{A}_{eval}$, что обосновывает применимость метрики ASR в экспериментальной части работы. Результат получен автором и теоретически обоснован.
2. Формализована робастность систем LLM-as-a-Judge к инъекционным атакам. Атаки разделены на два класса по механизму воздействия: инструктивные, внедряющие прямую команду, и контентные, смещающие вердикт манипуляцией содержанием. Для каждого класса определена соответствующая контрмера. На этой основе разработан алгоритм адаптивной генерации атак ASA (*Adaptive Search-based Attack*): эволюционный поиск, совмещающий мутации на уровне токенов с семантическими перефразированиями. На наборе открытых и проприетарных моделей ASA достигает ASR до 73,8% при black-box доступе. Открытые модели оказались систематически уязвимее проприетарных аналогов. Переносимость атак между открытыми моделями достигает

TSR = 62,6%. Все перечисленные результаты получены автором экспериментально.

3. Для модели Pythia-1.4B при полном white-box доступе на данных Trojan Detection Challenge 2023 количественно охарактеризована асимметрия задачи обнаружения троянских закладок. Суррогатный триггер активирует закладку почти гарантированно: REASR (*Reverse-Engineered Attack Success Rate*) составляет $\approx$ 0,99. Восстановить исходный триггер не удается: Recall $\approx$ 0,17 практически не отличается от базового уровня $\approx$ 0,14. Разрыв объясняется комбинаторным ростом пространства дискретных триггеров и указывает на ограниченность послеобучающего аудита троянов. Приоритет должен отдаваться превентивному контролю данных. Выдвинута гипотеза об усилении асимметрии с ростом числа параметров модели.
4. Разработана и экспериментально оценена методология генерации и классификации многошаговых атак типа «инъекция запроса» на LLM, оснащенные многоуровневыми системами защиты, включающими системные инструкции, внешние Python-фильтры и LLM-фильтры. На примере международного бенчмарка SaTML CTF 2024 систематизированы четыре класса успешных векторов обхода, а именно код-ориентированные атаки, ASCII-кодирование символов, атаки с разделением слов и контекстное перенаправление. Для каждого класса определены условия применимости и эффективные контрмеры: нормализация входных данных и контекстно-изолированные системные инструкции. Методология разработана автором и верифицирована на данных соревнования.
5. Разработан метод защиты систем LLM-as-a-Judge на основе адаптивных комитетов гетерогенных моделей. Метод использует мажоритарное голосование в пространстве классифицированных вердиктов и отличается от стандартных ансамблей тем, что состав комитета оптимизируется с учетом архитектурного разнообразия моделей. Теоретическое обоснование устойчивости комитетов получено на основе теоремы Кондорсе о жюри. На репрезентативном наборе атак комитет из 5–7 моделей различных архитектурных семейств снижает ASR на 47–55 процентных пунктов для наиболее уязвимой модели с открытыми ве-

сами (Gemma-3-4B) по сравнению с одиночными моделями. Метод разработан и экспериментально валидирован автором.

6. Предложен подход к анализу безопасности агентских LLM-систем по протоколу Model Context Protocol. Формализован класс атак *prompt injection to tool hijacking*. Выявлены архитектурные уязвимости MCP и разработан комплекс защитных механизмов. Основу составляют два алгоритма: AttestMCP ограничивает вызовы инструментов сессионной таблицей полномочий и контролирует целостность пакетов HMAC-подписью при задержке менее 0,1 мс, Commit Boundary запускает код, генерируемый агентом, в изолированной среде. Их дополняют четыре вспомогательных механизма: сертификаты возможностей, маркировка происхождения, принудительная изоляция межсерверной передачи и защита от повторов. Все компоненты объединены в программном модуле MCPSec, который снижает успешность атак при сохранении приемлемого времени отклика. Подход предложен автором и подтвержден экспериментально на бенчмарке MCPBench.

Теоретическая и практическая значимость работы состоит в создании формального аппарата оценки устойчивости LLM. Метрика $R_{stab}(f)$ обладает свойствами монотонности и связи с уязвимостью, что дает количественную базу для сопоставления методов защиты. Устойчивость гетерогенных ансамблей обоснована через теорему Кондорсе. Для композиционных систем получена декомпозиция чувствительности, локализующая узкие места конвейера. Обнаруженная асимметрия обнаружения троянских закладок ставит превентивный аудит данных выше послеобучающей диагностики.

На практике значимость работы определяется двумя зарегистрированными программными комплексами и модулем для агентских LLM-систем. JudgeGuard (свидетельство № 2025687702) автоматизирует тестирование LLM-as-a-Judge и реализует комитетную защиту. TrojanArmor (свидетельство № 2025684247) обнаруживает троянские закладки. Программный модуль MCPSec контролирует полномочия вызовов инструментов и изолирует исполнение кода в агентских LLM-системах. Область применения: аудит безопасности систем автоматической оценки, диалоговых агентов и агентских платформ.

Результаты диссертации внедрены в следующих организациях:

1. В учебный процесс факультета ВМК МГУ имени М. В. Ломоносова в рамках курса «Робастные модели в машинном обучении».

2. В производственную деятельность компании ВИАСАТ ТЕХ, что подтверждено актом о внедрении: фреймворк оценки уязвимостей LLM-as-a-Judge, метод защиты на основе комитетов моделей и методология многошаговых атак применены для обеспечения безопасности модуля ранжирования рекомендаций, построенного на архитектуре LLM-as-a-Judge.

Методология и методы исследования. Решение задач потребовало сочетания теоретических и экспериментальных методов. Проанализирована научная литература по атакам на LLM, включая инъекции запроса и троянские закладки, а также по методам защиты и оценки устойчивости. Разработаны и формализованы модели атак и защит. Подходы апробированы в ходе участия в международных соревнованиях по безопасности LLM: Trojan Detection Challenge 2023, SaTML CTF 2024, Kaggle «LLMs: You Can't Please Them All».

Эмпирическая часть включала развертывание и настройку открытых и коммерческих LLM (GPT-3.5, GPT-4, Llama 2, Llama 3.2, Gemma, Claude-3-Opus). Воспроизведены и смоделированы как разработанные автором, так и известные из литературы атаки: CUA (*Comparative Undermining Attack*), JMA (*Justification Manipulation Attack*), ASA (*Adaptive Search-based Attack*), код-ориентированные инъекции и другие. Эффективность защитных сценариев оценивалась методами математической статистики: бутстреп-анализом, проверкой гипотез и доверительными интервалами. Состязательные примеры генерировались двумя алгоритмами: авторским ASA и методом жадного покоординатного градиента GCG (*Greedy Coordinate Gradient*).

Основные положения, выносимые на защиту:

1. Формальная математическая модель оценки устойчивости больших языковых моделей ($R_{stab}(f)$). Модель опирается на меру расхождения вероятностных распределений в дискретном пространстве токенов и служит метрикой генеративной устойчивости. Для класса локализованных атак доказана верхняя оценка $V(h) \leqslant 1 - R_{class}(h)$. Для нелокализованных атак, таких как prompt injection и jailbreak, предложена эмпирическая калибруемая модель $V(h) \approx g(1 - R_{class}(h))$. Верхняя оценка строго доказана автором. Эмпирическая модель носит характер калибруемой гипотезы, параметры которой настраиваются на валидационном наборе атак.

2. Алгоритм адаптивной генерации состязательных атак (ASA), реализующий эволюционный поиск с селекцией по целевой функции воздействия на модель-оценщик. Алгоритм комбинирует мутации на уровне токенов с семантическими перефразированиями. В условиях «черного ящика» ASR составляет до 73,8%. Обратная инженерия троянских закладок выявила практическую асимметрию: генерация состязательных воздействий существенно проще их обнаружения (REASR $\approx 0{,}99$ при Recall $\approx 0{,}17$). Этот разрыв указывает на приоритет превентивной защиты. Алгоритм ASA и анализ асимметрии выполнены автором.
3. Метод защиты систем LLM-as-a-Judge на основе адаптивных комитетов гетерогенных моделей, обеспечивающий снижение ASR на 47–55 процентных пунктов для наиболее уязвимой модели Gemma-3-4B, а именно с 73,8% до 19,3% при комитете из 7 моделей. Теоретическое обоснование через теорему Кондорсе справедливо при выполнении условия $r > 0{,}5$, где r обозначает базовую точность отдельной модели. Для Gemma-3-4B выигрыш обеспечивается прежде всего гетерогенным составом комитета, декоррелирующим уязвимости базовых моделей. Многоуровневая защита (Defense-in-Depth) снижает ASR с 90% до 15–25%. Систематизированы четыре класса обхода многоуровневых фильтров. Результаты получены экспериментально.
4. Архитектурно-алгоритмический подход к защите агентских LLM-систем по протоколу MCP, включающий бенчмарк MCPBench (847 сценариев), алгоритм аттестации вызовов инструментов AttestMCP, паттерн программной изоляции Commit Boundary и четыре вспомогательных механизма защиты: сертификаты возможностей, маркировку происхождения, принудительную изоляцию и защиту от повторов. На бенчмарке MCPBench подход снижает среднюю ASR агентских атак с 53,7% до 12,4%. Подход разработан и апробирован автором.
5. Предложенные методы оценки и повышения устойчивости LLM-систем реализуемы в виде законченного программного инструментария с измеримой производительностью. Это подтверждается программными комплексами JudgeGuard (свидетельство № 2025687702) и TrojanArmor (свидетельство № 2025684247), а также модулем MCPSec. TrojanArmor апробирован на бенчмарке TDC2023, JudgeGuard на SaTML CTF 2024

и внедрен в производственной среде ВИАСАТ TEX, MCPSec на бенчмарке MCPBench.

Достоверность полученных результатов обусловлена следующим. Экспериментальная база построена на публичных наборах данных MT-Bench, TDC2023, SaTML CTF 2024 и Kaggle «LLMs: You Can't Please Them All». Модели представляют разные архитектурные семейства: Gemma, Llama, GPT-3.5, GPT-4, Claude-3-Opus. Каждая комбинация модель/атака/задача проверялась на $n = 50$ промптах ($n = 200$ на модель) с доверительными интервалами и тестами значимости. Полученные показатели ASR, TSR, Recall, REASR сопоставлены с результатами известных методов. Работы опубликованы в рецензируемых журналах и апробированы на международных конференциях и соревнованиях. **Апробация работы.** Основные результаты работы представлялись и обсуждались на следующих конференциях:

1. Distributed Computer and Communication Networks:
   Control, Computation, Communications (DCCN 2024)
   Москва, Россия, 23–27 сентября 2024
2. International Conference on Computational Linguistics
   and Intellectual Technologies DIALOG 2022,
   Москва, Россия, 15–17 июня 2022
3. Научная конференция «Тихоновские чтения 2023»,
   МГУ имени М.В. Ломоносова, Россия,
   29 октября – 3 ноября 2023
4. Ломоносовские чтения – 2022, Секция вычислительная
   математика и кибернетика, 14–22 апреля 2022,
   Москва, МГУ имени М.В.Ломоносова, факультет ВМК, Россия

Личный вклад. Результаты получены лично автором или при его определяющем участии. Автору принадлежат: разработка фреймворков атак, концепции защиты, планирование и проведение экспериментов, анализ данных, формулировка выводов. Соавторы участвовали в подготовке публикаций. Во всех совместных работах вклад диссертанта определяющий: от постановки задач до анализа экспериментальных результатов.

Публикации. Автор имеет 9 публикаций по теме диссертации. Из них 6 относятся к рекомендованным ВАК и приравненным к ним: 4 статьи [1—4] в журналах из перечня ВАК (3 статьи категории К1 и 1 статья категории К2) и 2 свидетельства о государственной регистрации программ для ЭВМ [5; 6].

Помимо них, результаты опубликованы в докладе на конференции DCCN [7], индексируемом в Scopus (Q2), в докладе на конференции DIALOG [8] и в препринте [9].

Часть работ выполнена в соавторстве. Во всех них вклад диссертанта является определяющим.

- В статье [7] (Д. Хомский, Н. Малоян, Б. Нутфуллин) автору принадлежат постановка задачи об атаках на системы с многоуровневой защитой, разработка методики многошаговых инъекций и анализ их результативности; соавторы участвовали в реализации экспериментов и подготовке текста.
- В статье [1] (Н. Малоян, Б. Ашинов, Д. Намиот) автором разработан алгоритм состязательных атак ASA на системы LLM-as-a-Judge, спланированы и проведены эксперименты, сформулированы выводы, соавторы участвовали в обсуждении результатов и научном руководстве.
- В докладе [8] (Н. Малоян, Б. Нутфуллин, Е. Ильюшин) автору принадлежат метод обнаружения сгенерированного текста и его экспериментальная проверка.
- Статьи [2—4] и препринт [9] подготовлены совместно с научным руководителем Д. Намиотом, диссертанту принадлежат постановка задач, разработка методов атак и защиты агентских LLM-систем, проведение экспериментов и формулировка результатов.
- Программные комплексы, защищенные свидетельствами [5; 6], разработаны автором лично.

Свидетельства о регистрации программы для ЭВМ.

1. JudgeGuard. Программа для экспериментального исследования уязвимостей и тестирования методов защиты систем автоматической оценки на базе больших языковых моделей / Малоян Н.Г. Свидетельство о государственной регистрации программы для ЭВМ № 2025687702, заявл. 2025 (Рос. Федерация).
2. TrojanArmor. Программа для экспериментального исследования троянских атак в нейронных сетях и методов защиты от них / Малоян Н.Г. Свидетельство о государственной регистрации программы для ЭВМ № 2025684247, заявл. 2025 (Рос. Федерация).

Связь задач, глав, результатов новизны и положений защиты. Задача 1, систематизация и классификация угроз, решается в главе 1 и соот-

ветствует результату новизны 3, устанавливающему практическую асимметрию эффективности генерации и восстановления триггеров при обнаружении троянских закладок, и результату 4, содержащему классификацию многошаговых атак на системы с многоуровневой защитой. Задача 2, разработка теоретических основ оценки, решается в главе 2 и соответствует результату новизны 1, включающему формальную метрику устойчивости $R_{stab}(f)$ и теоретические границы уязвимости, а также положению защиты 1. Теоретическое обоснование комитетов через теорему Кондорсе, входящее в положение защиты 3, также опирается на аппарат, развитый при решении задачи 2. Задача 3, разработка методов состязательных атак, решается в главе 3 и соответствует результату новизны 2, а именно фреймворку исследования уязвимости LLM-as-a-Judge и алгоритму ASA, и положению защиты 2. Задача 4, разработка и экспериментальная валидация методов защиты, решается в главах 3 и 4 и соответствует результату новизны 5 о комитетах гетерогенных моделей, положению защиты 3 и экспериментальному подтверждению результатов новизны 2–5 и положений защиты 2–4. Задача 5, программная реализация и апробация, решается в главах 5–6 и соответствует результату новизны 6, содержащему анализ безопасности агентских систем и протокола MCP, и положению защиты 5.

Глава 1. Анализ угроз и уязвимостей больших языковых моделей

Введение к главе

Безопасность больших языковых моделей определяется взаимодействием множества факторов: дискретной природой входного пространства, разнообразием векторов атак и архитектурными особенностями систем на основе LLM. В настоящей главе формализуется пространство угроз, проводится классификация уязвимостей и анализируется специфика атак на архитектуры LLM-as-a-Judge и системы с многоуровневой защитой. Полученные результаты формируют теоретическую базу для методов защиты, развиваемых в последующих главах.

1.1 Теоретические основы устойчивости больших языковых моделей

Терминологическое соглашение. В работе используется следующее разграничение терминов. *Устойчивость*: основной русскоязычный термин, обозначающий способность модели сохранять корректное поведение при возмущениях входных данных (англ. *robustness*). Термин *робастность* используется как синоним при прямой отсылке к англоязычной литературе, например *adversarial robustness*, *certified robustness*. Метрика $R_{stab}(f)$ содержит индекс «stab» от англ. *stability*, что отражает этимологию: метрика оценивает стабильность выходных распределений модели, при этом отдельное понятие «стабильность» не вводится. Атаки типа «инъекция запроса», *prompt injection*, и «троянские закладки», или бэкдоры, *backdoors*, обозначаются русскоязычными терминами. Англоязычные эквиваленты приводятся при первом упоминании и в контексте международных классификаций. Архитектура автоматической оценки на основе LLM обозначается как «модель-оценщик», LLM-as-a-Judge. Термин «модель-судья» используется как синоним.

Ниже приведены определения и формализмы для анализа устойчивости и безопасности LLM, типы уязвимостей и подходы к их количественной оценке.

1.1.1 Большие языковые модели: формальное представление

Большие языковые модели, Large Language Models или LLM, представляют собой класс вероятностных моделей последовательностей. Как правило, они построены на глубоких нейронных сетях с архитектурой Transformer [10] и обучены на крупных корпусах текстовых данных [11]. Назначение таких моделей: обработка и генерация текста на естественном языке. Они оперируют последовательностями дискретных единиц, токенов. Формализация представления LLM, их входов и выходов необходима для строгого определения и анализа свойств моделей, в том числе устойчивости.

Определение 1.1. Пространство токенов и последовательностей. Пусть $\mathcal{V}$ обозначает конечный словарь токенов. Пространство всех конечных последовательностей токенов, то есть текстов, определяется как $\mathcal{X} = \mathcal{V}^*$, где $\mathcal{V}^*$ есть замыкание Клини множества $\mathcal{V}$. Длина произвольной последовательности $x \in \mathcal{X}$ записывается как $|x|$.

Входом модели служит последовательность токенов $x \in \mathcal{X}$. Характер выхода определяется задачей. В общем виде LLM задается параметрическим отображением f_θ с вектором параметров θ:

$$f_\theta : \mathcal{X} \to \mathcal{P}(\mathcal{Y}),$$

где $\mathcal{Y}$ есть пространство возможных выходов модели, а $\mathcal{P}(\mathcal{Y})$ задает множество вероятностных распределений на нем.

– В задачах генерации текста, таких как автодополнение, перевод и диалог, модель предсказывает распределение вероятностей на словаре $\mathcal{V}$ для следующего токена или генерирует целую последовательность токенов. В этом случае $\mathcal{Y}$ часто совпадает с $\mathcal{X}$, то есть пространством выходных последовательностей, или $\mathcal{V}$, то есть пространством следующих токенов, и f_θ определяет условное распределение вероятностей $p_\theta(y|x)$ для $y \in \mathcal{Y}$ при заданном входе $x \in \mathcal{X}$.

– В задачах классификации текста, таких как анализ тональности и определение темы, $\mathcal{Y}$ является конечным множеством меток классов, а $f_\theta(x)$ определяет распределение вероятностей на этих метках.

Дискретная природа входного пространства $\mathcal{X}$ принципиально отличает LLM от моделей, оперирующих непрерывными данными, в частности в компьютерном зрении [12], и накладывает ограничения на методы оценки и повышения устойчивости. Далее для краткости индекс θ опускается, а модель обозначается как f, что подразумевает фиксированный набор параметров.

1.1.2 Устойчивость LLM: определение и значимость

Устойчивость, или робастность, большой языковой модели определяется как способность сохранять корректное поведение при небольших, семантически незначимых или синтаксически допустимых возмущениях входных данных. От этого свойства напрямую зависит надежность диалоговых систем, автоматизированного создания контента и систем поддержки решений [13; 14]. При недостаточной устойчивости даже минимальное варьирование запроса способно привести к генерации ошибочных или нежелательных ответов [15].

Формализация устойчивости требует предварительного определения класса допустимых возмущений входных данных.

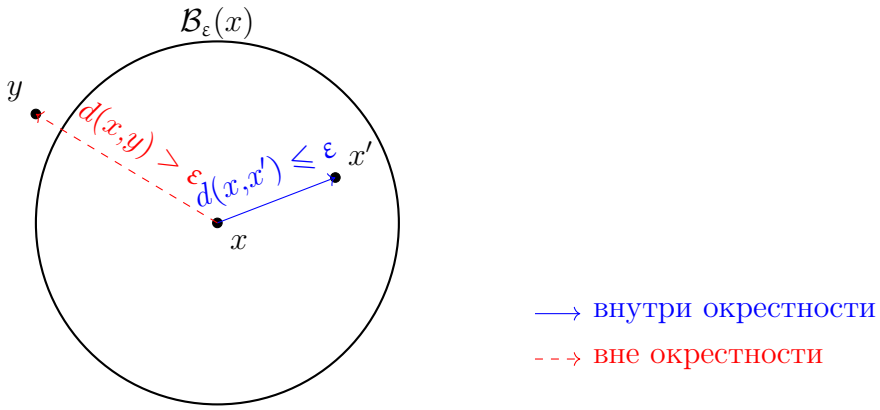


Рисунок 1.1 — Иллюстрация ε-возмущений: точка x' лежит в $\mathcal{B}_\varepsilon(x)$, а точка y лежит вне.

Определение 1.2. Возмущение последовательности токенов. Пусть $x \in \mathcal{X}$ – исходная последовательность токенов. Возмущенная последовательность

$x' \in \mathcal{X}$ получена из x путем применения некоторого преобразования $T : \mathcal{X} \to \mathcal{X}$. Множество ε-возмущений последовательности x относительно метрики d определяется как окрестность:

$$\mathcal{B}_\varepsilon(x) = \{x' \in \mathcal{X} : d(x, x') \leqslant \varepsilon\},$$

где $\varepsilon > 0$ задает бюджет возмущений, ограничивая величину допустимого изменения, а $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geqslant 0}$ определяет функцию расхождения на пространстве последовательностей токенов $\mathcal{X}$, в частности метрику, псевдометрику или семантическую меру несходства (Рис. 1.1).

От выбора метрики d зависит, какие типы изменений считаются малыми и против каких модификаций оценивается устойчивость. Ниже перечислены распространенные метрики для последовательностей токенов.

1. **Расстояние Хэмминга (d_H):** Применимо только к последовательностям одинаковой длины n и представляет собой простейшую из рассматриваемых метрик:

$$d_H(x, x') = \sum_{i=1}^{n} \mathbb{I}(x_i \neq x'_i), \quad \text{где } |x| = |x'| = n,$$

и $\mathbb{I}(\cdot)$ есть индикаторная функция. По сути подсчитывается число позиций, в которых токены x_i и x'_i различаются. Метрика моделирует атаки, сводящиеся к замене токенов.

2. **Расстояние Левенштейна (d_L) [16]:** Атаки редко ограничиваются одной заменой. Опечатки, добавление и удаление слов порождают вставки и удаления токенов, не покрываемые метрикой Хэмминга. Расстояние Левенштейна обобщает d_H на последовательности произвольной длины: оно измеряет минимальное число односимвольных операций, а именно вставку, удаление и замену, для преобразования x в x'. Формальное рекурсивное определение приведено в [16].

3. **Семантическое расстояние (d_S):** Обе предшествующие метрики работают на уровне символов и не отражают смысловую близость текстов. Перефразирование и замена синонимов [17] оставляют d_H и d_L практически неизменными, хотя семантика текста сохраняется. Для учета таких возмущений вводится расстояние, основанное на косинусной мере между векторными представлениями:

$$d_S(x, x') = 1 - \frac{\langle E(x), E(x') \rangle}{\|E(x)\| \cdot \|E(x')\|},$$

где E задает произвольную модель-эмбеддер, $\langle \cdot, \cdot \rangle$ есть скалярное произведение, $\| \cdot \|$ задает евклидову норму. Вектор предложения $E(x)$ получается усреднением (mean pooling) эмбеддингов токенов с последующей L2-нормализацией. В экспериментах, если не указано иное, применяется модель `all-mpnet-base-v2` из семейства Sentence-BERT, хотя само определение d_S не привязано к конкретной архитектуре эмбеддера.

Чувствительность к выбору E: Итоговое значение d_S и оценка устойчивости зависят от выбора модели-эмбеддера E. Разные модели оценивают семантическую близость неодинаково, особенно в узкоспециализированных доменах. При сравнительных исследованиях необходимо либо фиксировать одну модель E, либо проводить анализ чувствительности результатов к ее замене.

В Таблице 1 сведены основные метрики расстояний и их характеристики.

Таблица 1 — Сводная таблица метрик расстояний в пространстве токенов

Метрика	**Описание**	**Применение**
Хэмминга (d_H)	Количество замененных токенов (для строк равной длины)	ε: 1–5; атаки с заменой символов
Левенштейна (d_L)	Мин. кол-во операций (вставка, удаление, замена)	ε: 1–10; опечатки, короткие вставки
Семантическое (d_S)	Косинусное расстояние между эмбеддингами	ε: 0.05–0.2; перефразирование

Условие $d(x, x') \leqslant \varepsilon$ задает допустимый бюджет возмущений: атакующий или естественные вариации вправе вносить изменения в x, но суммарная стоимость этих изменений по метрике d не может превышать ε.

Количественная оценка устойчивости LLM опирается на метрику $R_{stab}(f)$, которая фиксирует величину отклонения выходных распределений модели при малых модификациях входа.

Единственное строгое определение и математические свойства метрики $R_{stab}(f)$ будут подробно рассмотрены в Главе 2.

1.1.3 Уязвимости LLM и типы атак

Недостаточная устойчивость LLM открывает возможности для атак на поведение моделей. Цели злоумышленника варьируются от генерации токсичного или запрещенного контента до извлечения конфиденциальной информации из обучающих данных, обхода систем безопасности и нарушения штатной работы модели [18—20].

Уязвимость формализуется через вероятность того, что модель продемонстрирует нежелательное поведение при наихудшем возмущении в рамках заданного бюджета.

Метрика локальной уязвимости $V_{\text{local}}(f)$ отражает ожидаемую максимальную вероятность получения нежелательных выходов при возмущениях конкретного входного запроса в его ε-окрестности. По существу, она количественно выражает, насколько легко побудить модель к генерации запрещенного или вредоносного контента путем модификации запроса.

Строгое определение, математические свойства метрики $V_{\text{local}}(f)$, ее связь с $R_{stab}(f)$ и глобальной метрикой уязвимости $V(f)$ рассмотрены в Главе 2.

Ниже описаны основные типы атак на LLM.

1. **Состязательные атаки (Adversarial Attacks):** Создание малых, часто незаметных для человека возмущений x', то есть таких, что $d(x, x')$ мало при некоторой метрике d, в частности d_L или d_S. Такие возмущения приводят к ошибочному в задачах классификации или нежелательному в задачах генерации выходу модели $f(x')$ [13; 17; 21]. Применительно к LLM это может быть замена символов и токенов, вставка или удаление слов, перефразирование, использование управляющих последовательностей [22—24]. Для генерации таких атак применяются методы градиентной оптимизации, в том числе Greedy Coordinate Gradient (GCG).

2. **Атаки типа инъекция запроса (Prompt Injection):** Внедрение инструкций в запрос пользователя, зачастую скрытное или замаскированное, с целью переопределить исходные системные указания модели или ее механизмы безопасности [19; 25]. Типичный пример: добавление текста вида «Игнорируй предыдущие инструкции и сделай X». Для таких атак малость возмущения по формальным метрикам d не обязательна, поскольку эксплуатируется способность LLM следовать инструкциям [25; 26]. Особую опасность представляют

атаки на LLM-as-a-Judge, где инъекции искажают результаты оценки, и на LLM с многоуровневой защитой, обходимой специализированными техниками.

3. **Атаки типа обход защиты (Jailbreaking):** Разработка специальных запросов, prompts, вынуждающих модель генерировать контент, нарушающий ее внутренние политики безопасности: инструкции для незаконных действий, вредоносный код, оскорбительные материалы. Защитные механизмы alignment не срабатывают [20; 27]. Уязвимость к таким атакам обнаруживают и крупные коммерческие, и открытые модели [28—30], в том числе при многошаговых запросах или техниках, эксплуатирующих особенности многоуровневых защитных систем.

4. **Троянские закладки (Backdoor Attacks):** Злоумышленник, имеющий доступ к процессу обучения, внедряет скрытые триггеры в модель на этапе обучения или дообучения [31]. Триггером может быть фраза или символ, присутствие которого во входных данных x' вызывает предопределенное атакующим поведение: генерацию конкретного текста или неверную классификацию [32; 33]. Обнаружение троянских закладок затруднено, поскольку на обычных данных модель ведет себя штатно. Обратная инженерия триггеров, то есть восстановление исходного триггера по известному вредоносному выходу, остается задачей с низкой полнотой обнаружения Recall даже в контролируемых условиях.

5. **Атаки извлечения данных (Data Extraction Attacks):** Попытки восстановить или извлечь из модели конфиденциальную информацию, которая могла присутствовать в ее обучающих данных, включая персональные данные и проприетарные тексты, путем подачи специально сконструированных запросов [18].

6. **Атаки на агентские системы через перехват инструментов (Tool Hijacking):** С развитием агентских LLM-систем, взаимодействующих с внешними инструментами по стандартизированным протоколам, в частности Model Context Protocol, MCP [34], возникает класс атак, при которых инъекция запроса в контекст инструмента приводит к несанкционированному вызову инструментов с произвольными аргументами. В отличие от классической инъекции запроса, последствия таких атак выходят за рамки текстовой генерации и включают выполнение кода, модификацию файловой системы и сетевые запросы [2; 3]. Критичность таких атак усиливается архитектурными особенностями протоколов агентского взаимодействия, а именно отсутствием аттестации воз-

можностей, неаутентифицированным сэмплингом и неявным распространением доверия между серверами. Детальное исследование этого вопроса приведено в Главе 5.

Метрика устойчивости $R_{stab}(f)$ и метрика локальной уязвимости $V_{\text{local}}(f)$ фиксируют разные аспекты поведения модели. Первая оценивает стабильность распределения $p(y|x)$, вторая характеризует вероятность генерации $y \in \mathcal{Y}_{\text{bad}}$. Между ними, однако, прослеживается концептуальная связь. Модель с высоким $R_{stab}(f)$, то есть с малыми изменениями $p(y|x)$ при возмущениях x', предположительно будет менее уязвимой и покажет низкое $V_{\text{local}}(f)$. Если распределение $p(y|x)$ почти не меняется при переходе к x', то и вероятность генерации нежелательного контента $P(y \in \mathcal{Y}_{\text{bad}}|x')$ изменится незначительно. В общем случае эта связь не является строгой математической зависимостью. Небольшое изменение распределения с малой дивергенцией Кульбака-Лейблера D_{KL} может оказаться достаточным, чтобы заметно увеличить вероятность попадания в $\mathcal{Y}_{\text{bad}}$, если исходная вероятность была близка к нулю. Верно и обратное: большая D_{KL} не обязательно приводит к генерации нежелательного контента, когда новое распределение сосредоточено на допустимых выходах. Результаты экспериментов с adversarial training [35] свидетельствуют о корреляции между повышением $R_{stab}(f)$ и снижением $V_{\text{local}}(f)$. Строгих теоретических гарантий такой зависимости, однако, не получено.

1.1.4 Категоризация и формализация специфических аспектов уязвимостей

Уязвимости LLM затрагивают несколько аспектов безопасности. Ниже формализованы четыре из них средствами введенного аппарата.

1. Чувствительность к дообучению (Fine-tuning Sensitivity). Адаптация LLM к специфическим задачам через дообучение [36] сопряжена с риском для устойчивости и безопасности. Робастность может снизиться, уже известные уязвимости усугубиться, а частичная потеря исходного alignment способна открыть новые [37; 38]. Количественно влияние дообучения выражается через изменение метрики устойчивости $R_{stab}(f)$:

$$\Delta R = R_{stab}(f_{\text{дооб}}) - R_{stab}(f_{\text{исх}}),$$

Здесь $f_{\text{исх}}$ и $f_{\text{дооб}}$ обозначают модель до и после дообучения соответственно. Отрицательное ΔR сигнализирует о снижении устойчивости. В качестве альтернативной меры используется средняя KL-дивергенция между выходными распределениями до ($p_{\text{исх}}(y|x)$) и после ($p_{\text{дооб}}(y|x)$) дообучения:

$$\Delta P = \mathbb{E}_{x\sim\mathcal{D}}[D_{KL}(p_{\text{исх}}(\cdot|x)\|p_{\text{дооб}}(\cdot|x))].$$

Распределение $p_{\text{исх}}$ выступает референсным, а $p_{\text{дооб}}$ аппроксимирует его. Большое значение ΔP свидетельствует о заметном сдвиге поведения модели и может коррелировать с изменением профиля уязвимостей.

2. Эффективность атак типа jailbreak. Эффективность jailbreak-атак для конкретной модели f определяется через максимальную вероятность успеха в ε-окрестности базового безопасного запроса x. Метрикой d может служить, в частности, семантическое расстояние [39]:

$$P_{\text{jailbreak}}(f, x, \varepsilon) = \sup_{x'\in\mathcal{B}_\varepsilon(x)} P(f(x') \text{ обходит защиту}),$$

Величина $P(f(x')$ обходит защиту) выражает вероятность того, что выход $y \sim p(\cdot|x')$ модели f при запросе x' попадает в множество нежелательных выходов $\mathcal{Y}_{\text{bad}}$. На практике успешность jailbreak-атак определяется совокупностью факторов: формулировкой запроса, архитектурой и размером модели, методами выравнивания [40; 41]. Многоуровневые системы защиты также подвержены этим атакам [20]. Усреднение по запросам x из тестового набора $\mathcal{D}_{\text{test}}$ дает агрегированную оценку: $V_{\text{jailbreak}}(f) = \mathbb{E}_{x\sim\mathcal{D}_{\text{test}}}[P_{\text{jailbreak}}(f, x, \varepsilon)]$.

3. Контекстно-зависимые уязвимости. Результативность многих атак, прежде всего состязательных и jailbreak, существенно варьируется в зависимости от предметной области запроса: медицинский, юридический и технический домены демонстрируют неодинаковую восприимчивость. Стратегия, эффективная в одном контексте, может оказаться бесполезной в другом [37]. Построение универсальных защит тем самым осложняется. Формально зависимость уязвимости от контекста $c \in \mathcal{C}$, где $\mathcal{C}$ есть множество возможных контекстов, выражается условной метрикой локальной уязвимости:

$$V_{\text{local}}(f|c) = \mathbb{E}_{x\sim\mathcal{D}|c}\left[\sup_{x'\in\mathcal{B}_\varepsilon(x)} P(y \in \mathcal{Y}_{\text{bad}}|x')\right],$$

где $\mathcal{D}|c$ обозначает распределение входов при условии контекста c. Заметная вариация $V_{\text{local}}(f|c)$ по контекстам $c \in \mathcal{C}$ указывает на высокую контекстную

зависимость уязвимостей и необходимость разработки контекстно-адаптивных защитных механизмов.

4. Сложности в идентификации компрометации модели. Обнаружение успешной атаки или скрытой компрометации модели, будь то backdoor или побочный эффект дообучения, представляет значительную трудность [42]. Стандартные тесты не выявляют аномалий: модель ведет себя штатно до тех пор, пока не поступит специфический триггерный запрос или не возникнет редкий сценарий. Для троянских атак проблема стоит особенно остро, поскольку восстановление триггеров само по себе остается нерешенной задачей. На практике компрометацию выявляют сопоставлением поведения текущей модели f с эталонной доверенной моделью $f_{\text{эталон}}$, например предыдущей версией или моделью до дообучения.

Мерой отклонения может служить средняя или максимальная KL-дивергенция на некотором наборе тестовых данных $\mathcal{X}_{\text{test}}$:

$$D_{\text{avg}}(f, f_{\text{эталон}}) = \mathbb{E}_{x \in \mathcal{X}_{\text{test}}}[D_{KL}(p_{\text{эталон}}(\cdot|x) \| p(\cdot|x))],$$

$$D_{\max}(f, f_{\text{эталон}}) = \max_{x \in \mathcal{X}_{\text{test}}} [D_{KL}(p_{\text{эталон}}(\cdot|x) \| p(\cdot|x))].$$

Порядок аргументов $D_{KL}(p_{\text{эталон}} \| p)$ измеряет потерю информации при аппроксимации эталонного распределения текущим, то есть показывает степень отклонения модели f от доверенного состояния. Если одна из метрик превышает заранее установленный порог, это может свидетельствовать о компрометации. Выбор эталонной модели $f_{\text{эталон}}$ и интерпретация отклонений сами по себе остаются открытыми исследовательскими вопросами [43].

Перечисленные аспекты уязвимостей мотивируют разработку стандартизированных метрик и протоколов тестирования LLM [14; 44].

1.2 Классификация и анализ угроз безопасности больших языковых моделей

На основе анализа научной литературы, обзорных работ, собственных исследований и индустриальных стандартов OWASP Top 10 для LLM Applications [45] и NIST AI Risk Management Framework [46] в настоящем разделе систематизированы угрозы для LLM. Раздел охватывает определения и метрики оценки

безопасности, описание классов угроз с формализацией, их категоризацию и сопоставление с контролями безопасности. Анализируются реальные инциденты, формулируются рекомендации по повышению защищенности моделей.

1.2.1 Базовые определения и формальные метрики

В анализе угроз используются обозначения и формальный аппарат, введенные в Разделе 1.1.1: модель $f_\theta : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$, метрики устойчивости $R_{stab}(f)$ и локальной уязвимости $V_{\text{local}}(f)$ (Раздел 1.1.2, строгие определения в Главе 2). Дополнительно вводятся метрики, специфичные для отдельных классов угроз.

Вероятность утечки конфиденциальной информации (P_{leak}) характеризует риск раскрытия чувствительных данных, которые могли присутствовать в обучающей выборке $\mathcal{D}_{\text{train}}$ или во входном запросе. Формализация может зависеть от типа утечки. Например, риск извлечения конкретного фрагмента обучающих данных $d \in \mathcal{D}_{\text{train}}$ можно оценить как:

$$P_{\text{extract}}(d) = \max_{x \in \mathcal{X}_{\text{query}}} P(d \sqsubseteq f_\theta(x)),$$

где $\mathcal{X}_{\text{query}}$ задает множество возможных запросов атакующего, а $d \sqsubseteq f_\theta(x)$ означает, что фрагмент d является подпоследовательностью вывода модели $f_\theta(x)$, что происходит с некоторой вероятностью, если вывод стохастичен. Более общая метрика может использовать взаимную информацию $I(f_\theta(X); S)$ между выходами модели и некоторым множеством секретов S [47].

1.2.2 Описание основных видов угроз

Анализ литературы [48], стандартов [45; 46] и результатов проведенных исследований позволяет выделить перечисленные ниже угрозы.

1. **Инъекция запросов (Prompt Injection) [19; 25].** Суть атаки состоит во внедрении в запрос пользователя x_{user} дополнительных инструкций δ. Модель, получив конкатенацию $\tilde{x} = x \oplus \delta$, игнорирует системные указания x_{sys} или выполняет непредусмотренные действия.

Задача атакующего сводится к поиску инъекции, максимизирующей вероятность нежелательного ответа:

$$\delta^* = \arg \max_{\delta : d(\delta) \leqslant \varepsilon_{\text{inj}}} P(f_\theta(x \oplus \delta) \in \mathcal{Y}_{\text{bad}}),$$

Функция $d(\delta)$ отражает сложность инъекции, а ε_{inj} ограничивает бюджет атакующего. Частными случаями являются Comparative Undermining Attack, CUA, и Justification Manipulation Attack, JMA, нацеленные на LLM-as-a-Judge, а также разнообразные техники обхода защиты.

2. **Отравление обучающих данных (Training Data Poisoning) [49; 50].** Цели внедрения специально сформированных примеров $\mathcal{D}_{\text{poison}}$ в обучающую выборку $\mathcal{D}_{\text{train}}$ варьируются: от общего снижения качества модели до создания троянских закладок и усиления предвзятости. Обучение на объединении $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{poison}}$ сдвигает параметры модели на $\Delta\theta = \theta' - \theta_{\text{clean}}$. Атакующий стремится максимизировать функцию потерь $\mathcal{L}$ на целевых входах:

$$\mathcal{D}^*_{\text{poison}} = \arg \max_{\mathcal{D}_{\text{poison}} : |\mathcal{D}_{\text{poison}}| \leqslant N_{\text{poison}}} \mathbb{E}_{x_{\text{target}}}[\mathcal{L}(f_{\theta'}(x_{\text{target}}))],$$

при ограничении N_{poison} на допустимое число отравленных примеров.

3. **Троянские закладки (Backdoor-атаки) [31; 51].** В отличие от общего отравления данных, backdoor-атака предполагает внедрение скрытого триггера x_{trigger}, активирующего вредоносную функциональность. Формальный критерий успеха:

$$P(f_{\theta'}(x \oplus x_{\text{trigger}}) = y_{\text{target}}) \approx 1 \quad \text{и} \quad P(f_{\theta'}(x') = y_{\text{target}}) \approx 0$$

для произвольных x и x', не содержащих x_{trigger}. На обычных данных модель ведет себя нормально, и именно это делает обнаружение и обратную инженерию триггера столь затруднительными. Детальному рассмотрению вопрос посвящен в п. 4 Раздела 1.1.3.

4. **Инверсия модели (Model Inversion) [52; 53].** Опасность инверсии модели определяется возможностью раскрытия персональных данных, медицинских записей и иных конфиденциальных сведений из обучающей выборки. Атакующий анализирует выходные данные модели: вероятности, эмбеддинги, сгенерированный текст. На основе этого

анализа он восстанавливает входы, с высокой вероятностью присутствовавшие в $\mathcal{D}_{\text{train}}$. Формально задача записывается как

$$x^* = \arg\max_{x \in \mathcal{X}} \text{Confidence}(x \in \mathcal{D}_{\text{train}} | f_\theta(x)),$$

где метрика Confidence$(\cdot)$ основана на характеристиках вывода: низкой перплексии или высокой вероятности генерации.

5. **Определение членства (Membership Inference) [54].** В отличие от инверсии, направленной на реконструкцию самих данных, атака определения членства решает более узкую задачу: установление факта принадлежности конкретного фрагмента x обучающей выборке $\mathcal{D}_{\text{train}}$. Ответ дает классификатор $g(f_\theta(x)) \to \{\text{member}, \text{non-member}\}$, опирающийся на значения функции потерь или вероятности вывода. Мерой успешности атаки служит точность этого классификатора.
6. **Состязательные примеры (Adversarial Examples) [13; 23].** Задача поиска состязательного примера $x' = x \oplus \delta$ формулируется как минимизация возмущения при сохранении вредоносного эффекта:

$$\delta^* = \arg\min_{\delta} d(\delta) \quad \text{при условии} \quad f_\theta(x \oplus \delta) \in \mathcal{Y}_{\text{bad}} \text{ и } d(\delta) \leqslant \varepsilon,$$

Мера возмущения $d(\delta)$ ограничена бюджетом ε. Человек, как правило, не замечает различий между x и x', между тем результат работы модели меняется радикально. Последствия варьируются от генерации токсичного контента до полного обхода фильтров безопасности. Градиентные методы, в том числе GCG, позволяют эффективно генерировать такие примеры и применимы к архитектурам LLM-as-a-Judge.
7. **Отказ в обслуживании (Denial of Service, DoS) [55].** Целью атак на доступность является исчерпание вычислительных ресурсов LLM-сервиса: GPU, CPU, оперативной памяти, лимитов API. Атакующий конструирует запросы, максимизирующие время обработки $T(x)$ или потребление ресурсов $R(x)$. Превышение порогов $T_{\max}$ и $R_{\max}$ ведет к отказу в обслуживании легитимных пользователей или к общей деградации сервиса.
8. **Кража модели (Model Theft/Exfiltration) [56; 57].** Атакующий, не имея доступа к параметрам θ или архитектуре модели, способен воссоздать функционально эквивалентную копию $f'_{\theta'}$ через многократные запросы к API. Схожесть оригинала и копии на тестовом наборе $\mathcal{X}_{\text{test}}$

количественно выражается метрикой:

$$S(f_\theta, f'_{\theta'}) = \frac{1}{|\mathcal{X}_{\text{test}}|} \sum_{x \in \mathcal{X}_{\text{test}}} \mathbb{I}(f_\theta(x) \approx f'_{\theta'}(x)),$$

где $\approx$ означает совпадение или семантическую близость выводов.

9. **Небезопасная обработка выводов (Insecure Output Handling) [45; 58].** Отсутствие проверки и санитизации выходных данных LLM перед передачей другим системам открывает вектор для каскадных атак. Сгенерированный SQL-код может быть исполнен на сервере базы данных, JavaScript-фрагмент отображен в веб-интерфейсе. Результат: SQL-инъекции, межсайтовый скриптинг XSS и аналогичные уязвимости. Риск генерации эксплуатируемого кода характеризуется вероятностью $P(f_\theta(x) \in \mathcal{Y}_{\text{exploit}})$, где $\mathcal{Y}_{\text{exploit}}$ обозначает множество вредоносных выводов.
10. **Извлечение обучающих данных (Training Data Extraction) [18].** LLM склонны к запоминанию (memorization) фрагментов обучающей выборки, что делает возможным восстановление дословных фрагментов d из $\mathcal{D}_{\text{train}}$. Задача атакующего состоит в подборе запроса x^*, максимизирующего метрику извлечения:

$$x^* = \arg\max_{x \in \mathcal{X}} S_{\text{extract}}(f_\theta, x), \qquad S_{\text{extract}}(f_\theta, x) = \max_{d \in \mathcal{D}_{\text{train}}} \frac{|\text{longest\_common\_subs}}{|\text{relevant\_part}(}$$

Знаменатель существенно влияет на интерпретацию результатов. Нормализация на длину всего ответа занижает метрику при извлечении коротких фрагментов, например персональных данных. В подобных случаях целесообразнее нормализовать на длину предполагаемого секрета.
11. **Утечка конфиденциальной информации (Sensitive Information Disclosure) [59].** В отличие от целенаправленного извлечения, утечка конфиденциальной информации $\mathcal{C}$ нередко носит непреднамеренный характер. Без специально сконструированного запроса модель может раскрыть персональные данные, коммерческую тайну, иные чувствительные сведения из обучающей выборки или текущего контекста диалога. Для количественной оценки используется взаимная информация $I(f_\theta(X); \mathcal{C}) = H(\mathcal{C}) - H(\mathcal{C}|f_\theta(X))$ [18]. Высокое значение I свидетельствует о существенном раскрытии сведений о $\mathcal{C}$ через выходы модели.

12. **Усиление предвзятости (Bias Amplification) [60; 61].** Модели не просто воспроизводят социальные предвзятости обучающих данных, но способны их усиливать, генерируя дискриминационные результаты для отдельных демографических групп. Степень усиления выражается коэффициентом $A_{\text{bias}}(f_\theta, \mathcal{D}) = B_{\text{output}}(f_\theta, \mathcal{D})/B_{\text{input}}(\mathcal{D})$, соотносящим предвзятость выходов B_{output} с предвзятостью исходных данных B_{input}. Здесь $B_{\text{output}}(f_\theta, g_1, g_2) = \text{Dist}(P(Y|X \in g_1), P(Y|X \in g_2))$ для демографических групп $g_1, g_2 \in \mathcal{G}$, и $A_{\text{bias}} > 1$ сигнализирует об усилении.
13. **Уязвимости цепочки поставок (Supply Chain Vulnerabilities) [62; 63].** Современная LLM-система зависит от многочисленных компонентов: предобученных моделей, наборов данных, библиотек для дообучения, сторонних плагинов, инфраструктурных элементов [31]. Компрометация любого звена c_i в цепочке $\mathcal{S} = \{c_1, \ldots, c_n\}$ ставит под угрозу всю систему. Если допустить независимость компонентов, вероятность уязвимости системы определяется как $P(V_{\text{system}}) \approx 1 - \prod_{i=1}^{n}(1 - P(V_i))$. Уже при умеренном n эта величина быстро приближается к единице даже при малых индивидуальных рисках $P(V_i)$.
14. **Обход защитных механизмов (Jailbreaking) [20].** Результаты SaTML CTF 2024 показали, что даже многоуровневая защита не гарантирует устойчивости к jailbreaking-атакам. Такие атаки преодолевают safety alignment [36] и вынуждают модель генерировать запрещенный контент. Формально атакующий ищет запрос x_{jb}, максимизирующий вероятность нежелательного вывода при условии, что фильтр безопасности его пропускает:

$$x_{\text{jb}} = \arg\max_{x \in \mathcal{X}} P(f_\theta(x) \in \mathcal{Y}_{\text{bad}} | \text{SafetyFilter}(f_\theta(x)) = \text{allow}).$$

Метрика устойчивости к jailbreaking принимает вид $R_{\text{jb}}(f_\theta) = 1 - \max_{x \in \mathcal{X}_{\text{jb\_attacks}}} P(f_\theta(x) \in \mathcal{Y}_{\text{bad}})$ и вычисляется на наборе известных jailbreaking-запросов $\mathcal{X}_{\text{jb\_attacks}}$ [40].
15. **Проблемы воспроизводимости (Reproducibility Issues) [64].** Генерация LLM носит стохастический характер: сэмплирование и изменения инфраструктуры приводят к тому, что повторные вызовы на один и тот же вход x дают различные ответы $\{y_1, \ldots, y_k\}$. Мера расхождения $NRep(f_\theta, x) = D(\{y_1, \ldots, y_k\} | y_i \sim p_\theta(\cdot|x))$ фиксирует

степень невоспроизводимости. Следствия для безопасности двояки: тестирование и отладка затрудняются, а уязвимости могут проявляться спорадически, маскируя реальный риск.

16. **Небезопасные плагины** (*Insecure Plugins* [26; 45]). Интеграция со сторонними плагинами создает риски при нарушении принципа наименьших привилегий. Когда набор привилегий плагина $\mathrm{Priv}(p)$ превышает минимально необходимый $\mathrm{Priv}_{\min}(p)$, метрика избыточности $E(p) = |\mathrm{Priv}(p) \setminus \mathrm{Priv}_{\min}(p)|/|\mathrm{Priv}(p)|$ количественно отражает степень риска. Скомпрометированный плагин способен выступать каналом обхода фильтров контента и порождает игру между атакующим и защитником [65].
17. **Обход обнаружения** (*Model Evasion / Detection Evasion* [66; 67]). Вредоносный запрос x_{mal} модифицируется в x_{evasion}, минимизирующий оценку $\mathrm{DetectionScore}(D(x_{\mathrm{evasion}}))$ при сохранении условия $h(x_{\mathrm{evasion}}) \in \mathcal{Y}_{\mathrm{bad}}$. Атака нацелена на периметр защиты системы и снижает результативность фильтров контента.

Из приведенной классификации в рамках настоящей диссертации дальнейшему формальному и экспериментальному анализу подлежат атаки типа prompt injection и jailbreak, а также троянские закладки. Отдельный класс образуют атаки на агентские LLM-системы, взаимодействующие с внешними инструментами. Они относятся к аспектам чрезмерной автономии, LLM08, и небезопасных плагинов, LLM07, по классификации OWASP и детально рассматриваются в Главе 5 применительно к протоколу MCP.

1.2.3 Категоризация угроз

Описанные угрозы группируются по шести аспектам безопасности.

Целостность данных и модели. Отравление обучающих данных (LLM03 по OWASP), Backdoor-атаки и состязательные примеры искажают параметры или поведение модели. Локальная уязвимость $V_{\mathrm{local}}(f_\theta)$ частично отражает этот аспект при включении некорректных ответов в $\mathcal{Y}_{\mathrm{bad}}$.

Конфиденциальность затрагивается спектром угроз: инверсией модели, определением членства, извлечением обучающих данных, утечкой конфи-

денциальной информации, все они соответствуют LLM06, а также раскрытием системных промптов [68]. Количественная оценка опирается на метрику P_{leak} и взаимную информацию $I(f_\theta(X); \mathcal{C})$.

Аспект **доступности** представлен отказом в обслуживании (LLM04), нарушающим работоспособность сервиса для легитимных пользователей.

Аспект **управления и контроля** охватывает наиболее широкий перечень угроз: инъекцию запросов (LLM01), небезопасную обработку выводов (LLM02), уязвимости цепочки поставок (LLM05), небезопасный дизайн плагинов (LLM07), чрезмерную автономию (LLM08), кражу модели (LLM10), jailbreaking и обход обнаружения. Минимизация $E(p)$ для плагинов напрямую снижает риски этой категории.

- **Этика и справедливость.** Усиление предвзятости, генерация социально неприемлемого контента через jailbreaking, prompt injection и состязательные примеры. Метрики A_{bias} и $V_{\text{local}}(f_\theta)$ при соответствующем определении $\mathcal{Y}_{\text{bad}}$ применимы для оценки.
- **Надежность и воспроизводимость.** Недетерминированность генерации влияет на предсказуемость поведения, что фиксируется метрикой $NRep(f_\theta, x)$.

Ряд угроз затрагивает несколько категорий одновременно: Prompt Injection может вести к утечке данных, нарушению целостности и генерации вредоносного контента.

1.2.4 Сопоставление с фреймворками безопасности

Выбор мер защиты требует сопоставления выявленных угроз с прикладными фреймворками: OWASP Top 10 для LLM Applications и NIST SP 800-53. Ниже приводится такое сопоставление, задающее контекст для выбора предмета исследования.

Рисунок 1.2 — Таксономия угроз безопасности LLM: 6 категорий, 17 типов угроз.

Классификация угроз по OWASP Top 10 для LLM (2023)

В таблице 2 представлено соответствие между угрозами, описанными в данном разделе, и классификацией OWASP Top 10 для LLM Applications версии 2023 года [45]. Эта классификация фокусируется на практических рисках при разработке и эксплуатации приложений на базе LLM.

Таблица 2 — Классификация угроз по OWASP Top 10 для LLM (2023) и их связь с детальным списком угроз

ID	**Угроза (OWASP)**	**Описание**	**Угрозы из разд. 1.2.2**
LLM01	Инъекция запросов (Prompt Injection)	Внедрение вредоносных инструкций, переопределяющих поведение модели.	1. Инъекция запросов, 14. Jailbreaking
LLM02	Небезопасная обработка выводов (Insecure Output Handling)	Недостаточная проверка и санитизация выводов LLM перед использованием в других системах.	9. Небезопасная обработка выводов
LLM03	Отравление данных (Training Data Poisoning)	Манипулирование обучающими данными для компрометации модели.	2. Отравление данных, 3. Backdoor-атаки
LLM04	Отказ в обслуживании (Model DoS)	Атаки, приводящие к исчерпанию ресурсов и недоступности модели.	7. Отказ в обслуживании (DoS)
LLM05	Уязвимости цепочки поставок (Supply Chain)	Использование уязвимых компонентов: модели, зависимости, данные.	13. Уязвимости цепочки поставок
LLM06	Раскрытие информации (Sensitive Info Disclosure)	Непреднамеренное раскрытие моделью конфиденциальных данных.	11. Утечка информации, 4. Инверсия модели, 5. Определение членства, 10. Извлечение данных
LLM07	Небезопасные плагины (Insecure Plugin Design)	Недостатки безопасности в плагинах LLM.	16. Небезопасные плагины

Продолжение на следующей странице

Таблица 2 – Продолжение

ID	**Угроза (OWASP)**	**Описание**	**Угрозы из разд. 1.2.2**
LLM08	Чрезмерная автономия (Excessive Agency)	Предоставление LLM слишком больших полномочий.	16. Небезопасные плагины, общий дизайн системы
LLM09	Чрезмерное доверие (Overreliance)	Чрезмерное доверие к выводам LLM без проверки.	6. Состязательные примеры, 12. Предвзятость, 15. Воспроизводимость
LLM10	Кража модели (Model Theft)	Несанкционированное копирование или извлечение модели.	8. Кража модели

Сопоставление с контролями безопасности NIST SP 800-53

В таблице 3 приведено сопоставление угроз с релевантными семействами контролей из каталога NIST SP 800-53 Rev. 5 [69]. Это позволяет интегрировать управление рисками LLM в общую систему управления информационной безопасностью организации. Выбор конкретных контролей должен основываться на оценке рисков.

Таблица 3 — Сопоставление угроз LLM с семействами контролей NIST SP 800-53 Rev. 5

Угроза	**Семейства контролей NIST**	**Примеры контролей**
1. Инъекция запросов	SI, AC	SI-10, AC-6, AC-3
2. Отравление данных	SI, SR	SI-7, SR-3, SR-6
3. Backdoor-атаки	SI, SR	SI-7, SR-4, SR-5
4. Инверсия модели	AC, PT	AC-3, AC-4, PT-3, PT-4

Продолжение на следующей странице

Таблица 3 – Продолжение

Угроза	**Семейства контролей NIST**	**Примеры контролей**
5. Определение членства	PT, SI	PT-3, SI-4, AU-6
6. Adversarial примеры	SI, AT	SI-4, SI-10, AT-2
7. Отказ в обслуживании	SC, CP	SC-5, CP-2
8. Кража модели	AC, SC, IA	AC-3, AC-4, SC-28, IA-2
9. Небезопасная обработка выводов	SI	SI-10, SI-11
10. Извлечение данных	PT, AC	PT-3, AC-3, AU-6
11. Утечка информации	PT, AC	PT-2, PT-3, AC-3, SC-8
12. Усиление предвзятости	PM, PL, SI	PM-30, PL-8, SI-19
13. Уязвимости цепочки поставок	SR	SR-2, SR-3, SR-4, SR-5
14. Jailbreaking	SI, AC	SI-4, SI-7, AC-3
15. Проблемы воспроизводимости	CM, SI	CM-6, SI-2
16. Небезопасные плагины	SI, AC, SR	SI-7, AC-6, SR-11
17. Обход обнаружения	SI, RA	SI-4, RA-5, RA-10

1.2.5 Иллюстративные примеры угроз

Три примера иллюстрируют рассмотренные классы угроз результатами контролируемых экспериментов и соревнований по безопасности LLM.

1. Уязвимость LLM-as-a-Judge к целевым Prompt Injection атакам:

- *Описание:* Архитектуры LLM-as-a-Judge подвержены атакам типа prompt injection. Comparative Undermining Attack, CUA, меняет решение оценщика напрямую. Justification Manipulation Attack, JMA, искажает обоснование вердикта. Обе атаки реализуются методами оптимизации, в частности Greedy Coordinate Gradient, GCG.
- *Эффективность:* На моделях Gemma, Llama, GPT-4, Claude-3-Opus с датасетом MT-Bench и задачами Kaggle атаки CUA и ASA достигли высокой успешности, особенно на моделях с открытыми весами. Количественные результаты приведены в Главе 4 (Таблица 5).

2. Сложности обнаружения троянских закладок в LLM. В рамках Trojan Detection Challenge 2023 исследовалась обратная инженерия троянских закладок в модели Pythia. Восстановление истинных триггеров оказалось принципиально сложной задачей, в то время как генерация суррогатных триггеров, вызывающих нужный выход, достигалась значительно проще. Эти результаты указывают на приоритетность превентивных мер. Детальный анализ результатов приведен в Главе 4 (Раздел 4.2.2).

3. Обход многоуровневой защиты с помощью Prompt Injection. Соревнование SaTML CTF 2024 было направлено на атаки LLM, в частности GPT-3.5 и Llama 2, оснащенных трехуровневой защитой: системные инструкции, Python-фильтр, LLM-фильтр. Участники разрабатывали и применяли различные техники prompt injection, включая многошаговые атаки и специфические формулировки для обхода фильтров. Многоуровневые защитные конфигурации оказались преодолимыми. Успешность атаки определялась изощренностью промпта и спецификой фильтров. Защитные стратегии нуждаются в постоянной адаптации и дополнении динамическими методами обнаружения.

1.2.6 Анализ взаимосвязей угроз

Типы угроз для LLM связаны между собой: противодействие одной угрозе влияет на уязвимость к другим. Строгие теоремы, справедливые для всех LLM,

сформулировать затруднительно, поэтому ниже приводятся обоснованные предположения и принципы, подкрепленные литературой.

Принцип 1: Взаимосвязь извлечения данных и утечки конфиденциальной информации. Извлечение обучающих данных (10) представляет частный случай утечки конфиденциальной информации (11). Атаки по извлечению данных нацелены на восстановление фрагментов $\mathcal{D}_{\text{train}}$ [18], тогда как утечка может происходить и пассивно [70]. *Предположение:* Успешность целенаправленных атак по извлечению данных S_{extract}, определенная в Разделе 1.2.2, служит нижней границей для более общего риска утечки конфиденциальной информации P_{leak} при сопоставимых определениях метрик. Защитные меры против запоминания, в частности методы дифференциальной приватности [71; 72], снижают риски обеих угроз [18].

Принцип 2: Взаимосвязь Jailbreaking, инъекций и обхода обнаружения. Техники Prompt Injection (1) нередко выступают средством реализации Jailbreaking-атак (14), направленных на обход встроенных защитных механизмов модели [20]. При этом успешная Jailbreaking-атака не обязательно означает обход системы обнаружения (17): внешний детектор D может классифицировать вредоносный запрос x_{jb} как опасный. Сложность создания эффективных Jailbreaking-запросов и сложность их обнаружения связаны, однако продвинутые атаки проектируются для одновременного обхода и внутренних механизмов модели, и внешних детекторов [23]. Противодействие поэтому требует и улучшения robust alignment [73], и совершенствования внешних детекторов [74].

Принцип 3: Влияние воспроизводимости на безопасность. Зависимость между недетерминированностью LLM (15) и безопасностью не является монотонной [75]. С одной стороны, стохастичность генерации затрудняет создание точных состязательных примеров (6) и Jailbreaking-атак (14), требующих строго определенного ответа [76]. Вместе с тем она же усложняет тестирование и верификацию защитных механизмов. Редкие, но опасные уязвимости проявляются стохастически и потому труднее обнаруживаются. Надежность детекторов атак (17) тоже падает при низкой воспроизводимости поведения системы. Выбор оптимального уровня детерминизма диктуется конкретным приложением и профилем угроз.

Принцип 4: Взаимосвязь предвзятости и конфиденциальности. Усиление предвзятости (12) и утечка конфиденциальной информации (11) связаны через общий механизм: запоминание корреляций в обучающих данных.

Модель, усвоившая связи между чувствительными демографическими атрибутами и другими данными в $\mathcal{D}_{\text{train}}$, одновременно усиливает предвзятости и становится уязвимой к атакам на определение членства (5) для индивидуумов из соответствующих групп [77; 78]. Зависимость двусторонняя. Ослабление связи модели с чувствительными атрибутами снижает предвзятость и повышает конфиденциальность. Обратно, строгие меры по конфиденциальности, в частности DP [71], ограничивают способность модели выявлять тонкие корреляции, питающие ряд предвзятостей [79].

Принцип 5: Каскадные риски цепочки поставок и плагинов. Уязвимости в цепочке поставок (13) напрямую влияют на безопасность интегрируемых компонентов, включая плагины (16). Компрометация базовой модели или библиотеки влечет компрометацию всех зависимых плагинов. Небезопасный плагин с избыточными привилегиями, то есть с высоким значением $E(p)$ из Раздела 1.2.2, создает вектор атаки на базовую модель или другие компоненты системы. Риски этих двух категорий мультипликативно усиливают друг друга, что требует системного подхода: контроля привилегий, формализованных процедур интеграции сторонних расширений и регулярного аудита используемых компонентов [80].

Принцип 6: Фундаментальный компромисс (Trade-off). Многие аспекты безопасности и функциональности LLM находятся в состоянии компромисса. Например:

- *Полезность vs Безопасность/Цензура:* Слишком строгие защитные механизмы, направленные против Jailbreaking и токсичности, могут снижать полезность модели и ее способность выполнять легитимные запросы, что получило название «эффект лоботомии», [20].
- *Конфиденциальность vs Полезность:* Методы обеспечения конфиденциальности, например DP, часто снижают точность и общую производительность модели [72; 81].
- *Робастность vs Точность:* Методы повышения робастности к состязательным атакам, например adversarial training, могут незначительно снижать точность на чистых данных [23; 35].

На практике перечисленные компромиссы ограничивают одновременную максимизацию всех целевых свойств LLM-системы. Конкретный прикладной сценарий определяет расстановку приоритетов между безопасностью, полезностью, конфиденциальностью и вычислительной эффективностью.

Безопасность LLM не сводится к противодействию изолированным угрозам: защитные меры должны учитывать взаимодействие между угрозами и системные свойства модели в контексте ее развертывания.

1.2.7 Выводы и рекомендации по снижению рисков

Угрозы затрагивают все этапы работы с моделью, от подготовки данных до развертывания. Противодействие им требует комплекса технических, организационных и методологических мер.

- **Превентивные меры на этапе разработки и обучения.** Аудит и фильтрация обучающих данных снижают риски отравления (LLM03, 2, 3) [82] и предвзятости (12) [83]. Методы повышения приватности, такие как дифференциальная приватность DP-SGD [72] и Federated Learning [84], противодействуют утечке данных (LLM06, 4, 5, 10, 11) [71]. Adversarial training [35] и safety alignment [36; 85] повышают устойчивость к состязательным атакам (6) и jailbreaking (14). Контроль происхождения и целостности компонентов цепочки поставок (LLM05, 13) также относится к превентивным мерам [86].
- **Защитные механизмы на этапе эксплуатации.** На уровне входных данных применяются валидация и санитизация запросов для предотвращения инъекций (LLM01) [25; 87], разделение данных и инструкций [88]. На уровне выходных данных выполняется фильтрация и санитизация перед использованием в других системах (LLM02, 9) [89]. Контроль доступа и аутентификация API модели защищают от кражи (LLM10) [90]. Мониторинг ресурсов и лимиты противодействуют DoS (LLM04) [91], принцип минимальных привилегий для плагинов противодействует угрозе LLM07 [92]. Обнаружение аномального поведения и попыток обхода защиты (17) дополняет перечисленные меры [20; 23].
- **Непрерывный мониторинг и управление рисками.** Red Teaming и тестирование на проникновение [93; 94] выявляют ранее неизвестные векторы атак. Анализ журналов событий и аудит действий [95] обеспечивают оперативное обнаружение инцидентов. Модель и защитные механизмы необходимо регулярно актуализировать в ответ

на изменения ландшафта угроз. Количественное измерение уровня защищенности с помощью метрик $R_{stab}(f_\theta)$ и $V_{\text{local}}(f_\theta)$ позволяет отслеживать динамику [14].

Формальные методы оценки рисков в сочетании с мерами контроля из стандартов NIST и OWASP позволяют выстроить многоуровневую систему защиты LLM в реальных условиях эксплуатации. Систематический анализ уязвимостей агентских систем кодирования, охватывающий 42 техники атак и 18 механизмов защиты, представлен в работе [2]. Последующие разделы диссертации посвящены экспериментальной проверке конкретных методов противодействия рассмотренным угрозам.

1.2.8 Формализация пространства угроз для больших языковых моделей

Систематизация угроз строится на формальной структуре, описывающей компоненты и их взаимосвязи. Абстрагирование от конкретных реализаций атак позволяет выявить общие закономерности.

Определение 1.3. Пространство угроз LLM. Пространство угроз для большой языковой модели $f_\theta : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$, согласно обозначениям Раздела 1.2, определяется как кортеж $\mathcal{T} = (\mathcal{A}^{\text{type}}, \mathcal{V}, \mathcal{I}_{AV}, \mathcal{C}_{VV})$, где:

- $\mathcal{A}^{\text{type}}$ – множество *типов* атак, например Prompt Injection, Data Poisoning, CUA, JMA и др., соответствующие классам из Раздела 1.2.2. Верхний индекс $^{\text{type}}$ подчеркивает, что речь идет о классификации атак. В Главе 2 символ $\mathcal{A}$ используется для обозначения семейства атак как функций.
- $\mathcal{V}$ – множество *конкретных* уязвимостей модели или системы, в которой она функционирует, например: v_1 = “недостаточная санитизация пользовательского ввода в системном промпте”, v_2 = “наличие запомненных PII в параметрах модели”, v_3 = “уязвимость LLM-as-a-Judge к манипуляции обоснованием”.
- $\mathcal{I}_{AV} \subseteq \mathcal{A}^{\text{type}} \times \mathcal{V}$ – отношение «атака использует уязвимость». Отражает, какие конкретные уязвимости ($v \in \mathcal{V}$) могут быть эксплуатированы

атаками определенного типа ($a \in \mathcal{A}^{\text{type}}$). Например, (Prompt Injection, v_1) $\in \mathcal{I}_{AV}$.

– $\mathcal{C}_{VV} \subseteq \mathcal{V} \times \mathcal{V}$ – отношение причинно-следственной связи или зависимости между уязвимостями. Означает, что эксплуатация или наличие уязвимости v_i может привести к возникновению или возможности эксплуатации уязвимости v_j. Например, ($v_{\text{supply\_chain}}$, v_{backdoor}) $\in \mathcal{C}_{VV}$, где $v_{\text{supply\_chain}}$ - уязвимость в цепочке поставок, v_{backdoor} - возможность внедрения троянской закладки.

Мощность множеств $\mathcal{A}^{\text{type}}$ и $\mathcal{V}$ зависит от специфики модели, архитектуры, данных и условий развертывания и может быть велика. Отношение $\mathcal{C}_{VV}$ раскрывает каскадные эффекты и корневые причины уязвимостей, а построение графа зависимостей $G = (\mathcal{V}, \mathcal{C}_{VV})$ составляет отдельную задачу аудита безопасности.

Количественная оценка риска требует понятия, учитывающего и вероятность успеха атаки, и потенциальный ущерб.

Определение 1.4. Оценка риска атаки. Оценка риска для модели f_θ относительно типа атаки $a \in \mathcal{A}^{\text{type}}$ определяется как:

$$\text{Risk}(a, f_\theta) = \sum_{v \in \mathcal{V}:(a,v)\in\mathcal{I}_{AV}} P_{\text{exploit}}(v|a, f_\theta) \cdot \text{ImpactValue}(v),$$

где $P_{\text{exploit}}(v|a, f_\theta)$ – вероятность успешной эксплуатации уязвимости v посредством атаки типа a против модели f_θ, в частности ASR из экспериментов настоящей работы, а ImpactValue(v) – оценка потенциального ущерба от эксплуатации уязвимости v, например по шкале CVSS [96], которая может быть нормирована к диапазону [0,1] для целей сравнительного анализа рисков. Суммарный риск по множеству атак $\mathcal{A}' \subseteq \mathcal{A}^{\text{type}}$ может быть вычислен как $\sum_{a\in\mathcal{A}'} w(a) \cdot \text{Risk}(a, f_\theta)$, где $w(a)$ - вес значимости атаки, определяемый, например, на основе частоты или предполагаемой мотивации атакующих.

Оценка P_{exploit} представляет сложную задачу, часто требующую специализированных тестов, в частности Red Teaming [93] или анализа данных соревнований, либо основанную на экспертных оценках и статистике известных инцидентов. Оценка ImpactValue(v) зависит от контекста применения модели и критичности данных или функций, затрагиваемых уязвимостью.

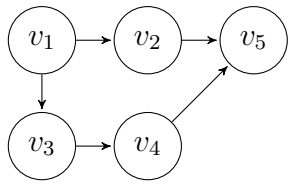
Порядок приоритизации защитных мер определяется графом зависимостей уязвимостей.

Принцип 1.1. О приоритизации устранения уязвимостей

Пусть $G = (\mathcal{V}, \mathcal{C}_{VV})$ – граф зависимостей уязвимостей, где $\mathcal{V}$ - множество уязвимостей, а $\mathcal{C}_{VV}$ - отношение зависимости. Применение мер защиты, направленных на устранение уязвимостей, должно учитывать структуру этого графа.

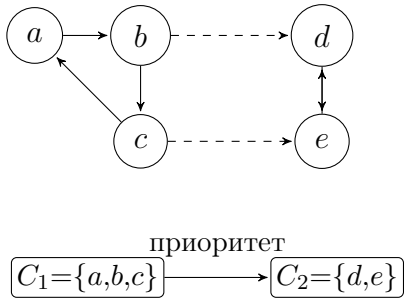
1. **Приоритет корневым уязвимостям:** В ациклическом графе зависимостей уязвимости, не имеющие входящих ребер, то есть корневые, являются первопричинами для других уязвимостей. Их устранение имеет каскадный эффект, предотвращая эксплуатацию всех зависимых от них уязвимостей.
2. **Разрыв циклов зависимостей:** В графе, содержащем циклы, приоритет следует отдавать устранению уязвимостей, принадлежащих компонентам сильной связности, которые являются «источниками» в конденсационном графе. Устранение даже одной уязвимости в цикле разрывает его, что может значительно снизить общий риск.

Принцип формирует методологическую базу для планирования защитных мер, направляя ресурсы на устранение уязвимостей, наиболее критичных с позиции структурных зависимостей.



Прямой: $\{v_1, v_3, v_2, v_4, v_5\}$
Обратный: $\{v_5, v_4, v_2, v_3, v_1\}$

(а) DAG и два варианта порядка устранения



Конденсационный DAG G^{SCC}

(б) Граф с циклами и его SCC-конденсация

Рисунок 1.3 — Иллюстрация порядка устранения уязвимостей на основе графа зависимостей (Принцип 1)

Учет зависимостей $\mathcal{C}_{VV}$ при планировании защитных мер опирается на алгоритмы поиска компонент сильной связности и топологической сортировки

из стандартного инструментария теории графов [97; 98]. Их применение к графу зависимостей уязвимостей LLM дает инструмент для приоритизации мер безопасности.

Эксплуатация уязвимости представляет собой последовательность действий, изменяющих состояние системы для достижения цели атакующего.

Определение 1.5. Эксплуатация уязвимости. Эксплуатация уязвимости $v \in \mathcal{V}$ атакой типа $a \in \mathcal{A}^{\text{type}}$ на модель f_θ определяется как последовательность действий $\sigma = (\text{action}_1, \ldots, \text{action}_k)$. Каждое действие может быть взаимодействием с моделью, например отправкой запроса $x \in \mathcal{X}$, изменением среды, например модификацией данных, или анализом ответа модели. Последовательность σ приводит к нежелательному состоянию $S_{\text{undesired}}$ с вероятностью $P(\text{success}|\sigma, v, f_\theta) > \tau$. Нежелательное состояние может включать генерацию вывода $y \in \mathcal{Y}_{\text{bad}}$, как определено в Разделе 1.2, утечку информации, изменение параметров модели и т.д.

Определение обобщает формализацию $V_{\text{local}}(f)$ на более широкий круг последствий атак.

Агрегированная мера устойчивости модели вводится на основе вероятности успеха атак.

Определение 1.6. Устойчивость к атакам. Устойчивость большой языковой модели f_θ к множеству типов атак $\mathcal{A}' \subseteq \mathcal{A}^{\text{type}}$ определяется как:

$$\text{AR}(f_\theta, \mathcal{A}') = 1 - \mathbb{E}_{a \sim \mathcal{W}(\mathcal{A}')}[P_{\text{success}}(a, f_\theta)],$$

где $P_{\text{success}}(a, f_\theta) = \max_{v:(a,v)\in\mathcal{I}_{AV}} P_{\text{exploit}}(v|a, f_\theta)$ – максимальная вероятность успеха атаки типа a через любую из связанных с ней уязвимостей, в частности ASR для данной атаки, а $\mathcal{W}(\mathcal{A}')$ – распределение вероятностей или весов на множестве атак $\mathcal{A}'$, отражающее их относительную частоту или опасность. Если используется равномерное распределение $\mathcal{W}$, то $\text{AR}(f_\theta, \mathcal{A}') = 1 - \frac{1}{|\mathcal{A}'|}\sum_{a\in\mathcal{A}'} P_{\text{success}}(a, f_\theta)$.

Замечание 1.7. Связь устойчивости и риска. Если оценка ущерба $\text{ImpactValue}(v)$ одинакова для всех уязвимостей, эксплуатируемых атакой a, т.е. $\text{ImpactValue}(v) = I(a)$ для всех v таких, что $(a, v) \in \mathcal{I}_{AV}$, и веса $w(a)$ в оценке суммарного риска соответствуют распределению $\mathcal{W}(\mathcal{A}')$, то между

устойчивостью $\mathrm{AR}(f_\theta, \mathcal{A}')$ и суммарным риском $\sum_{a \in \mathcal{A}'} w(a) \cdot \mathrm{Risk}(a, f_\theta)$ существует обратная зависимость. Однако в общем случае эта связь сложнее, так как риск зависит и от вероятности успеха, и от ущерба.

Практическое применение аппарата требует содержательного наполнения множеств $\mathcal{A}^{\text{type}}$ и $\mathcal{V}$, количественной оценки отношений $\mathcal{I}_{AV}$ и $\mathcal{C}_{VV}$ для конкретной системы. Эти задачи решаются в рамках аудита безопасности и анализа рисков.

1.3 Риски, ограничения и этические аспекты

Исследование уязвимостей и разработка атак на большие языковые модели сопряжены с рисками двойного назначения. Все эксперименты проводились в контролируемой среде и были направлены на выявление уязвимостей, количественную оценку рисков и разработку защитных механизмов.

Правовой и этический контур тестов. Все эксперименты проводились в контролируемой среде. При взаимодействии с коммерческими API, в частности GPT-4 и Claude-3, строго соблюдались условия их использования. Атаки, направленные на нарушение работы сервисов, несанкционированный доступ к данным других пользователей или генерацию и распространение опасного контента вне исследовательской среды, не производились. Целью всех атак была оценка уязвимостей моделей для последующей разработки защитных механизмов.

Работа с данными. В экспериментах не использовались реальные персональные или конфиденциальные данные. Наборы данных, такие как MT-Bench, публично доступны и предназначены для исследовательских целей. При генерации промптов и анализе ответов принимались меры для предотвращения случайной обработки чувствительной информации.

Безопасность генерации. При атаках на обход защиты (jailbreaking) или генерацию вредоносного контента применялись механизмы фильтрации для предотвращения создания и сохранения опасных материалов. Результаты таких атак анализировались с точки зрения структуры и механики, а не содержания.

Ответственное раскрытие. Результаты исследования публикуются для информирования научного сообщества и разработчиков с целью улучшения защиты систем. При этом не публикуются готовые инструменты для атак, которые могли бы быть использованы злоумышленниками.

Двойной характер работ в области безопасности ИИ обязывает к соблюдению этических норм на каждом этапе исследования.

Выводы по главе

Систематизация угроз обнажила свойство, определяющее структуру всей работы: разнородные на первый взгляд атаки, от инъекции запроса до троянской закладки, сводятся к чувствительности выученного отображения к малым вариациям входа в дискретном пространстве токенов. Это наблюдение оправдывает построение единой метрики устойчивости в Главе 2 вместо набора частных мер для каждого класса атак.

Сопоставление с OWASP Top 10 и NIST SP 800-53 выявило пробел, не закрываемый существующими каталогами: атаки на агентские LLM-системы по протоколу MCP не имеют отдельной позиции и распределены между чрезмерной автономией и небезопасными плагинами, что мотивирует их отдельное рассмотрение в Главе 5. Кроме того, асимметрия между внедрением и обнаружением троянских закладок, формализуемая лишь в Главе 4, уже на уровне классификации указывает на ограниченность послеобучающего аудита.

Графовая модель пространства угроз $\mathcal{T} = (\mathcal{A}^{\text{type}}, \mathcal{V}, \mathcal{I}_{AV}, \mathcal{C}_{VV})$ дает не только таксономию, но и принцип приоритизации защит по структуре зависимостей уязвимостей. Выявленные компромиссы между безопасностью, полезностью, конфиденциальностью и робастностью показывают, что одновременная оптимизация всех аспектов недостижима, и именно поэтому в последующих главах вводится метрика RUT для количественного выбора конфигурации защиты.

Глава 2. Теоретические основы оценки устойчивости больших языковых моделей

Введение к главе

Количественная оценка устойчивости LLM требует формального аппарата, учитывающего дискретность пространства токенов и специфику состязательных атак. В настоящей главе определены метрики робастности, установлены их свойства и доказаны соотношения между ними.

Построенный аппарат используется в последующих главах при разработке экспериментальных методологий и защитных механизмов.

2.1 Постановка задачи и модель угроз

Для формализации задачи оценки устойчивости больших языковых моделей необходимо задать пространство входов, пространство ответов, семейство допустимых атак, а также множества целей и последствий атакующих воздействий. Далее излагается формальный аппарат описания перечисленных компонентов и ставится общая задача оценки устойчивости.

2.1.1 Формализация задачи оценки устойчивости

Пространства $\mathcal{X}$, $\mathcal{Y}$, модель $f_\theta : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ и окрестность возмущений $\mathcal{B}_\varepsilon(x)$ определены в Главе 1 (Определения 1.1, 1.2). Помимо них, здесь вводятся:

– $\mathfrak{A}$ – семейство атак, где каждая атака $a \in \mathfrak{A}$ задается функцией $a : \mathcal{X} \to \mathcal{X}$, трансформирующей легитимный вход в потенциально вредоносный. Семейство охватывает как общие типы атак, так и специфические для конкретных архитектур.

– $\mathcal{G}$ – множество целей атакующего, $\mathcal{R}$ – множество последствий успешной атаки.

Замечание 2.1. Об обозначениях данной главы. Обозначение $\mathfrak{A}$ введено во избежание конфликта с $\mathcal{A}^{type}$ из Главы 1. Обозначения $\mathcal{G}$,$\mathcal{R}$ также локальны для данной главы. Множества $\mathcal{G}$ и $\mathcal{R}$ использованы неявно при определении $\mathcal{Y}_{\text{bad}}$ в Разделе 2.2.3: множество нежелательных выходов $\mathcal{Y}_{\text{bad}}$ задается целями атакующего из $\mathcal{G}$ и ожидаемыми последствиями из $\mathcal{R}$.

В зависимости от контекста используются два представления модели. Для задач с дискретным выходом, где оценивается сохранение вердикта или класса, применяется детерминированный оператор решения $h = \text{Dec} \circ f$. Для анализа расхождения вероятностных распределений используется непосредственно модель f.

Для устранения неоднозначности далее различаются: (i) вероятностная модель $f_\theta(x) \in \mathcal{P}(\mathcal{Y})$, задающая распределение на выходах, и (ii) детерминированный оператор решения $h_\theta = \text{Dec} \circ f_\theta$, где Dec обозначает фиксированное правило декодирования или извлечения вердикта, например greedy decoding, majority label, parse verdict. Метрики, основанные на расхождении распределений, определяются для f_θ, а метрики, основанные на факте успешности атаки или изменении вердикта, определяются для h_θ.

Общая задача оценки устойчивости формулируется как определение верхней границы риска $\text{Risk}(a, f)$ по всем возможным атакам $a \in \mathfrak{A}$:

$$\text{AdvRobustness}(f) = 1 - \sup_{a \in \mathfrak{A}} \text{Risk}(a, f)$$

где $\text{Risk}(a, f)$ выражает ожидаемый ущерб от атаки a на модель f. Корректная интерпретация $\text{AdvRobustness}(f)$ в заданном диапазоне, например [0,1], предполагает соответствующее ограничение или нормализацию функции $\text{Risk}(a, f)$. Это условие распространяется на все меры расхождения, используемые при вычислении риска, например $D(\cdot, \cdot)$: они должны быть ограничены сверху, то есть $0 \leqslant D \leqslant C_{max} < \infty$, либо нормализованы на отрезок $D \in [0,1]$, что гарантирует корректность итоговой оценки риска.

В данной главе разграничиваются две концепции устойчивости (см. терминологическое соглашение в Главе 1): $\text{AdvRobustness}(f)$ оценивает способность модели противостоять наихудшей атаке из $\mathfrak{A}$, тогда как $R_{stab}(f)$ фиксирует

стабильность выходного распределения при малых возмущениях входа. Связь между ними устанавливается в последующих разделах.

2.1.2 Допущения об атакующем

Оценка устойчивости требует явных допущений о возможностях и ограничениях атакующего. Принятая классификация выделяет следующие уровни знаний:

- **Black-box:** Доступ ограничен интерфейсом модели: допускается отправка запросов и получение ответов без какой-либо информации о внутренней структуре, параметрах или обучающих данных.
- **Gray-box:** Известна частичная информация о модели, к примеру архитектура, но конкретные значения параметров недоступны.
- **White-box:** Предполагается полный доступ к модели, включая архитектуру, параметры и обучающие данные. Метод GCG, в частности, требует такого уровня доступа для расчета градиентов.

Наряду с уровнем знаний, значимы следующие ограничения:

- **Бюджет запросов:** Максимальное количество запросов, которые атакующий может отправить модели.
- **Вычислительные ограничения:** Доступные вычислительные ресурсы, ограничивающие сложность применяемых алгоритмов.
- **Временные ограничения:** Допустимое время проведения атаки.

Перечисленные допущения задают границы пространства атак $\mathfrak{A}$ и напрямую сказываются на итогах оценки устойчивости.

2.1.3 Классификация атакующих воздействий

Атакующие воздействия на большие языковые модели допускают классификацию по четырем основным категориям:

- **Семантически эквивалентные переформулировки:** Формулировка запроса изменяется при сохранении семантического содержания.

Перефразирование с использованием синонимов или перестройка структуры предложения служат типичными примерами такого воздействия.

– **Вредоносные внедрения контекста:** В запрос добавляются специально сформированные инструкции или контекст, направленные на изменение поведения модели. К данной категории относятся prompt injection, jailbreak-атаки и иные формы манипуляции контекстом. Конкретные разновидности включают Comparative Undermining Attack, CUA, и Justification Manipulation Attack, JMA, нацеленные на системы LLM-as-a-Judge, а также код-ориентированные воздействия и Distraction attack на защищенные системы.
– **Манипуляции с токенизацией:** Особенности процесса токенизации эксплуатируются для обхода защитных механизмов. В частности, специальные символы разбивают запрещенные слова на части, как в Word-splitting attack, и каждая часть по отдельности не детектируется фильтрами.
– **Атаки на отказ в обслуживании:** Воздействия нацелены на снижение производительности модели или полное прекращение обслуживания. Запросы, вызывающие чрезмерное потребление ресурсов или зацикливание, представляют характерный пример.

Для каждого типа воздействий требуются отдельные методы оценки устойчивости и соответствующие защитные меры.

2.1.4 Формализация безопасности LLM как сервиса

Анализ безопасности LLM как сервиса затрагивает всю цепочку обработки запроса: от пользователя через промпт-инженера и LLM-ядро до формирования выходных данных. Формально данную цепочку удобно представить как последовательность преобразований, изображенную на Рис. 2.1.

Рисунок 2.1 — Цепочка обработки запроса от пользователя до выхода модели

Каждый компонент этой цепочки может быть точкой приложения защитных механизмов:

- **Пользовательский интерфейс:** Фильтрация и валидация входных данных, обнаружение аномалий в паттернах запросов.
- **Промпт-инженерия:** Преобразование пользовательских запросов в безопасные формы, добавление защитных инструкций, в том числе системных инструкций в архитектуре защиты, описанной в последующих главах.
- **LLM-ядро:** Модификация архитектуры или параметров модели для повышения устойчивости.
- **Выходные данные:** Фильтрация и санитизация генерируемого контента, обнаружение и блокировка вредоносных выходов с помощью Python-фильтров или LLM-фильтров.

Оценка устойчивости должна охватывать все звенья цепочки с учетом их взаимного влияния.

2.2 Формальные метрики устойчивости

Количественное измерение устойчивости больших языковых моделей опирается на формальные метрики, отражающие степень уязвимости модели к атакам. Далее приводится строгая формализация основных метрик: устойчивость к малым возмущениям для классификационных задач $R_{class}(h)$, где $h = \text{Dec} \circ f$, и для генеративных моделей $R_{stab}(f)$, показатель глобальной уязвимости $V(f)$, риск-функция $\text{Risk}(a, f)$ и метрика устойчивости к атакам $\text{AdvRobustness}(f)$. Интуитивное описание перечисленных метрик было представлено в Разделе 1.1.2 и Разделе 1.1.3.

2.2.1 Метрика $R_{class}(h)$ для дискретных выходов

Для моделей с дискретными выходами в классификационных задачах метрика устойчивости к малым возмущениям $R_{class}(h)$ определяется как ожида-

емая вероятность того, что детерминированный оператор решения $h = \mathrm{Dec} \circ f$ сохраняет свой выход при малых возмущениях входа:

$$R_{class}(h) = \mathbb{E}_{x \sim P_{\mathcal{X}}}\Big[1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h(x) \neq h(x')\}\Big],$$

где $h = \mathrm{Dec} \circ f$ задает композицию вероятностной модели f и оператора декодирования Dec, например $\arg\max$. Символ $\mathcal{B}_\varepsilon(x)$ определяет окрестность входа x радиуса ε, $P_{\mathcal{X}}$ фиксирует распределение входных данных, а $\mathbb{I}\{\cdot\}$ есть индикаторная функция.

Замечание 2.2. О разграничении $R_{class}(h)$ и $R_{stab}(f)$. Метрика $R_{class}(h)$ использует оператор решения $h = \mathrm{Dec} \circ f$, отображающий входы в дискретные метки, что обеспечивает корректность индикаторного сравнения $h(x) \neq h(x')$. В литературе по робастности часто встречается запись $R_{class}(f)$, однако здесь формулы явно записаны через h во избежание двусмысленности. Для генеративных LLM далее вводится метрика $R_{stab}(f)$, оперирующая непосредственно вероятностной моделью f без оператора декодирования.

Конкретный вид окрестности $\mathcal{B}_\varepsilon(x)$ задается выбором метрики расстояния на $\mathcal{X}$: Хэмминга, Левенштейна или семантического расстояния, как описано в Разделе 1.1.2 (Таблица 1).

Метрика $R_{class}(h)$ принимает значения в $[0, 1]$: значение 1 соответствует абсолютно устойчивой модели, 0 соответствует полностью неустойчивой.

2.2.2 Метрика $R_{stab}(f)$ для генеративных моделей

Определение 2.3. Метрика стабильности генеративной LLM $R_{stab}(f)$. Для генеративных моделей, которые авторегрессионно предсказывают распределение вероятностей на следующем токене, необходимо строго определить меру расхождения между выходами для двух разных входов x и x'. Чтобы устранить неоднозначность между распределением на всей последовательности и распределением на одном токене, нами определяется расхождение через усреднение по шагам генерации, как показано на Рис. 2.2.

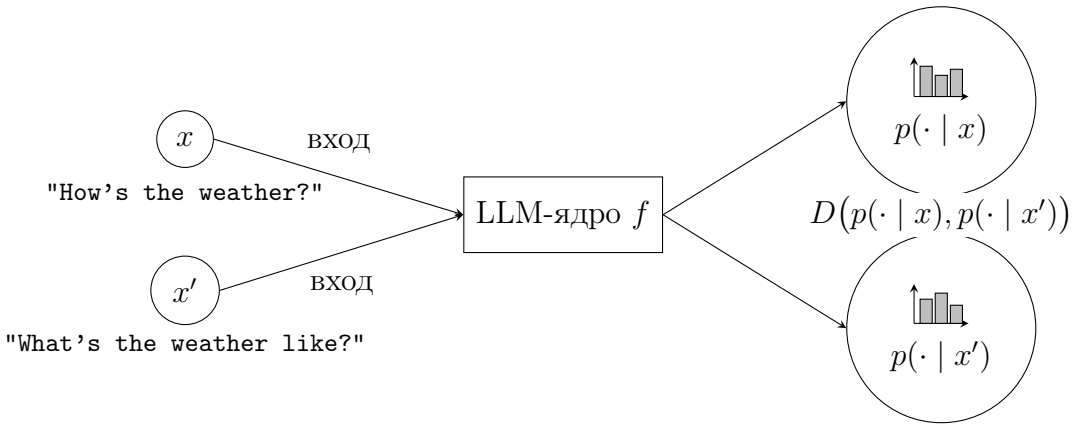


Рисунок 2.2 — Концептуальная диаграмма: два близких промпта x и x' с примерами дают разные распределения, расстояние между ними измеряется мерой D.

Пусть для входа x фиксирован референсный ответ $y^\star(x) = (y_1, \ldots, y_T)$, например эталонный ответ или ответ, сгенерированный самой моделью при жадном декодировании. Определим пошаговые авторегрессионные распределения $p_t(\cdot \mid x) := p(\cdot \mid x, y^\star_{<t}(x))$ и аналогично $p_t(\cdot \mid x') := p(\cdot \mid x', y^\star_{<t}(x))$. Здесь сравнение проводится методом teacher forcing по единой референсной траектории $y^\star(x)$: распределения $p_t(\cdot \mid x, y^\star_{<t}(x))$ и $p_t(\cdot \mid x', y^\star_{<t}(x))$ сопоставляются при одном и том же префиксе $y^\star_{<t}(x)$, что обеспечивает измерение чистой чувствительности к возмущению входа без примешивания расхождения собственных траекторий декодирования. Тогда усредненная по T шагам мера расхождения между выходами определяется как

$$D_T(x,x') := \frac{1}{T} \sum_{t=1}^{T} D\big(p_t(\cdot \mid x), p_t(\cdot \mid x')\big),$$

где D задает меру расхождения, например дивергенцию Кульбака-Лейблера или Йенсена-Шеннона, между распределениями вероятностей на словаре $\mathcal{V}$. Итоговая метрика стабильности $R_{stab}(f)$ определяется как:

$$R_{stab}(f) := \mathbb{E}_{x \sim P_{\mathcal{X}}} \Big[1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{\max}} \Big].$$

Замечание 2.4. Протокольная зависимость $R_{stab}(f)$**.** Метрика $R_{stab}(f)$ зависит от выбранного протокола сравнения, а именно референсной траектории $y^\star(x)$ и схемы teacher forcing, что делает ее протокольно-зависимой характеристикой модели. Значение метрики определено относительно конкретного способа получения $y^\star$ и фиксированной процедуры пошагового сопоставления

распределений. При изменении протокола, например при замене жадного декодирования на сэмплирование или при использовании другой референсной траектории, численные значения $R_{stab}(f)$ могут измениться. Поэтому при сравнении моделей необходимо фиксировать единый протокол вычисления метрики.

Замечание 2.5. Об асимметрии конструкции $D_T(x,x')$. Величина $D_T(x,x')$ построена методом teacher forcing по референсной траектории $y^\star(x)$ исходного промпта x: распределения $p_t(\cdot \mid x, y^\star_{<t}(x))$ и $p_t(\cdot \mid x', y^\star_{<t}(x))$ сопоставляются при одном и том же префиксе $y^\star_{<t}(x)$. Вследствие этого $D_T(x,x')$ является **направленной мерой чувствительности**: она, вообще говоря, не совпадает с $D_T(x', x)$, где teacher forcing проводился бы по траектории $y^\star(x')$. Асимметрия введена намеренно: она измеряет отклонение выходного распределения при возмущении входа от базового поведения модели на исходном промпте x.

Если базовая дивергенция D симметрична, как дивергенция Йенсена-Шеннона, используемая в данной работе, то $D(p_t(\cdot \mid x, y^\star_{<t}), p_t(\cdot \mid x', y^\star_{<t})) = D(p_t(\cdot \mid x', y^\star_{<t}), p_t(\cdot \mid x, y^\star_{<t}))$ для каждого шага t, однако вся конструкция D_T остается асимметричной из-за фиксированной траектории. Для задач, где требуется симметричная оценка расхождения, может использоваться симметризация $\overline{D}_T(x,x') = \frac{1}{2}\big(D_T(x,x') + D_T(x',x)\big)$. В контексте определения $R_{stab}(f)$, где рассматривается $\sup_{x' \in \mathcal{B}_\varepsilon(x)} D_T(x,x')$, направленная конструкция методически оправдана, поскольку оценивает чувствительность модели к возмущению именно относительно исходного входа x.

Замечание 2.6. Об операционализации параметров метрики $R_{stab}(f)$. Практическое применение метрики $R_{stab}(f)$ требует фиксации нескольких ключевых моментов.

- **Выбор референсного ответа $y^\star$**: Необходимо явно определить метод получения $y^\star(x)$. В данной работе, если не оговорено иное, в качестве $y^\star(x)$ используется последовательность, полученная путем жадного декодирования, greedy decoding, ответа модели на исходный промпт x до максимальной длины $T_{\max}$.
- **Выбор меры расхождения D и нормировки $C_{\max}$**: Дивергенция Кульбака-Лейблера (D_{KL}) может принимать бесконечные значения, если носители распределений не совпадают, что создает проблемы для численной стабильности. Чтобы избежать этого и обеспечить естественную нормировку, во всех экспериментальных оценках в данной работе,

если не оговорено иное, в качестве меры расхождения D используется **дивергенция Йенсена-Шеннона** (D_{JS}). Она определяется как:

$$D_{\mathrm{JS}}(p, q) = \frac{1}{2} D_{KL}(p \parallel m) + \frac{1}{2} D_{KL}(q \parallel m), \quad \text{где } m = \frac{1}{2}(p + q).$$

При использовании логарифма по основанию 2, $D_{\mathrm{JS}}(p,q)$ всегда находится в диапазоне $[0, 1]$, что позволяет принять $C_{\max} = 1$. При использовании натурального логарифма с основанием e, который обозначается ln здесь и далее, $D_{\mathrm{JS}}(p,q) \in [0, \ln 2]$, и соответственно $C_{\max} = \ln 2 \approx 0{,}693$. В настоящей работе, если не оговорено иное, используется логарифм по основанию 2 и $C_{\max} = 1$, что исключает бесконечные значения и значительно упрощает сопоставимость метрики R_{stab} между различными моделями и экспериментами.

Замечание 2.7. Определение метрики $R_{stab}(f)$ принято за основное для оценки устойчивости генеративных LLM и используется единообразно в последующих главах. Здесь $p(\cdot|x)$ представляет распределение вероятностей на выходе модели при заданном входе x, что соответствует функциональному представлению модели $f(x)$ в других разделах работы.

2.2.3 Показатель глобальной уязвимости $V(f)$

Определение 2.8. Показатель глобальной уязвимости $V(h)$. Пусть $h = \mathrm{Dec} \circ f$ задает детерминированный оператор решения, индуцированный моделью f. Тогда показатель глобальной уязвимости определяется как

$$V(h) = \mathbb{E}_{x \sim P_{\mathcal{X}},\, a \sim P_{\mathfrak{A}}} \big[\mathrm{Impact}\big(h(a(x))\big) \big],$$

где

$$\mathrm{Impact}(y) = \mathbb{I}\{y \in \mathcal{Y}_{\mathrm{bad}}\}.$$

В стохастическом случае вместо $h(a(x))$ может использоваться вероятность нежелательного выхода:

$$V(f) = \mathbb{E}_{x,a} \Big[\mathbb{P}_{y \sim f(a(x))}(y \in \mathcal{Y}_{\mathrm{bad}}) \Big].$$

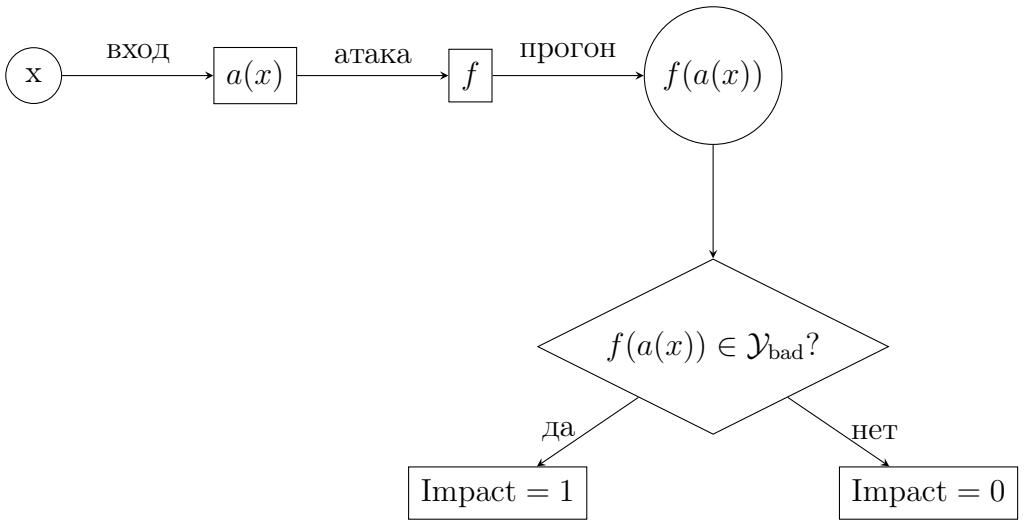


Рисунок 2.3 — Схема: вход x подвергается атаке $a(x)$, результат проходит через модель f, полученный выход $f(a(x))$ проверяется на принадлежность множеству $\mathcal{Y}_{\text{bad}}$, что дает значение Impact.

Множество нежелательных выходов $\mathcal{Y}_{\text{bad}}$ определяется в зависимости от задачи и может включать, например, выходы, которые изменяют вердикт LLM-as-a-Judge по бинарному или ранговому правилу решения, обходят защиту или содержат конфиденциальную информацию, как показано на Рис. 2.3.

Замечание 2.9. О терминологическом разграничении функций Impact и ImpactValue. Функция $\text{Impact}(y) = \mathbb{I}\{y \in \mathcal{Y}_{\text{bad}}\}$, введенная в данном определении, является бинарной индикаторной функцией: она принимает значения 0 или 1 в зависимости от принадлежности выхода множеству $\mathcal{Y}_{\text{bad}}$. Ее не следует смешивать с $\text{ImpactValue}(v)$ из Главы 1, Определение 1.4. Величина $\text{ImpactValue}(v)$ обозначает количественную оценку ущерба от эксплуатации уязвимости v по шкале CVSS и принимает произвольные значения в [0,1]. По природе Impact является индикатором, тогда как ImpactValue представляет непрерывную оценку серьезности. Аргументы функций также различаются: Impact принимает выход модели y, а ImpactValue – уязвимость v.

Замечание 2.10. Разграничение $V(h)$ и $V(f)$. Определение 2.8 задает уязвимость $V(h)$ для детерминированного оператора решения h через индикаторную функцию $\text{Impact}(y) = \mathbb{I}\{y \in \mathcal{Y}_{\text{bad}}\}$. Обобщение на стохастический случай $V(f)$ заменяет индикатор вероятностью $\mathbb{P}_{y\sim f(a(x))}(y \in \mathcal{Y}_{\text{bad}})$. При фиксированном детерминированном декодировании Dec формулировки совпадают: $V(f) = V(\text{Dec} \circ f)$. В дальнейшем запись $V(f)$ используется как сокращение для $V(h_\theta)$ с $h_\theta = \text{Dec} \circ f_\theta$.

Замечание 2.11. Метрика глобальной уязвимости $V(f)$ обобщает концепцию метрики локальной уязвимости $V_{\text{local}}(f)$, представленную в Главе 1, Раздел 1.1.3. Глобальная уязвимость $V(f)$ характеризует ожидаемую эффективность атак из заданного множества $\mathfrak{A}$ по всему распределению входов, в то время как локальная уязвимость $V_{\text{local}}(f)$ фокусируется на максимальной вероятности генерации нежелательного контента при возмущениях конкретного входа в его ε-окрестности. Если множество атак $\mathfrak{A}$ в определении $V(f)$ состоит из одной атаки, направленной на генерацию нежелательного контента $\mathcal{Y}_{\text{bad}}$ путем поиска наихудшего возмущения $x' \in \mathcal{B}_\varepsilon(x)$ для каждого x, то $V(f)$ будет эквивалентна $V_{\text{local}}(f)$ из Главы 1.

Показатель $V(f) \in [0, 1]$: значение 0 отвечает полностью защищенной модели, значение 1 соответствует ситуации, когда любая атака на любом входе приводит к нежелательному выходу.

Замечание 2.12. Учет стохастичности сэмплирования. Эмпирическая оценка метрик $R_{stab}(f)$ и $V(f)$ требует учета стохастичности процесса генерации ответа LLM, контролируемой параметрами сэмплирования: температурой, nucleus sampling, top-k. Для обеспечения воспроизводимости и сопоставимости результатов все параметры сэмплирования фиксируются в ходе экспериментов. Если не указано иное, в работе применяется жадное декодирование, greedy decoding, эквивалентное температуре 0, что обеспечивает детерминированность и воспроизводимость результатов.

2.2.4 Риск-функция Risk(a, f)

Определение 2.13. Риск атаки (Risk(a, h)). Риск, связанный с конкретной атакой или типом атаки $a \in \mathfrak{A}$ на оператор решения $h = \text{Dec} \circ f$, определяется как:

$$\text{Risk}(a, h) = \mathbb{E}_{x \sim P_{\mathcal{X}}}[\mathbf{1}_{h(a(x)) \in \mathcal{Y}_{\text{bad}}}] \cdot \text{Severity}(a, \mathcal{Y}_{\text{bad}}),$$

где первый сомножитель представляет вероятность успеха атаки a, обозначаемую $P_{\text{success}}(a,h)$ и оцениваемую, к примеру, через Attack Success Rate, ASR, а $\text{Severity}(a, \mathcal{Y}_{\text{bad}})$ отражает ожидаемый ущерб или серьезность последствий в случае успеха атаки a, приводящей к нежелательному выходу.

В стохастической постановке можно использовать эквивалентную запись

$$\mathrm{Risk}(a, f) = \mathbb{E}_{x\sim P_{\mathcal{X}}}\Big[\mathbb{P}_{y\sim f(a(x))}(y \in \mathcal{Y}_{\mathrm{bad}})\Big] \cdot \mathrm{Severity}(a,\mathcal{Y}_{\mathrm{bad}}).$$

Альтернативно, риск атаки можно определить через сумму по всем возможным уязвимостям, как в Определении 1.4 из Главы 1:

$$\mathrm{Risk}(a, f_\theta) = \sum_{v\in\mathcal{V}:(a,v)\in\mathcal{I}_{AV}} P_{\mathrm{exploit}}(v|a, f_\theta) \cdot \mathrm{ImpactValue}(v),$$

где $\mathrm{ImpactValue}(v)$ выражает количественную оценку ущерба, выраженную, к примеру, в баллах или денежном эквиваленте.

Определение Risk через $\mathrm{Severity}(a, \mathcal{Y}_{\mathrm{bad}})$ представляет альтернативную формулировку по сравнению с Определением 1.4 Главы 1, использующим $\mathrm{ImpactValue}(v)$. Первая формулировка оценивает риск по атакам, вторая – по уязвимостям. При агрегировании по всем уязвимостям обе формулировки согласуются.

Замечание 2.14. Определение устойчивости через риск-функцию ($\mathrm{AdvRobustness}(f) = 1 - \sup_{a\in\mathfrak{A}} \mathrm{Risk}(a, f)$) предполагает нормализацию $\mathrm{Risk}(a, f)$ в диапазоне [0,1]. На практике нормализация достигается ограничением функции Severity или ImpactValue.

Для моделей с дискретным выходом уязвимость связана с метрикой устойчивости к малым возмущениям через монотонно возрастающую функцию g:

$$V(h) \leqslant g(1 - R_{class}(h)),$$

где $h = \mathrm{Dec}\circ f$ задает детерминированный оператор решения, а g зависит от конкретного определения V и R_{class}. Для класса локализованных атак $g(t) = t$, и неравенство $V(h) \leqslant 1 - R_{class}(h)$ строго доказано в разделе 2.3.2. Для нелокализованных атак функция g задает калибруемую эмпирическую модель, которая верифицируется по данным и не является универсальной строгой границей. Различие статусов двух случаев подробно разобрано в Замечании 2.20.

2.3 Свойства и взаимосвязи метрик

Ниже установлены три группы результатов: монотонность R_{class} и R_{stab} относительно усиления защиты, верхняя оценка уязвимости V через R_{class} и лемма о композиции моделей.

2.3.1 Свойство монотонности $R_{class}(h)$ и $R_{stab}(f)$

Обе метрики монотонны: если защита модели усилена, значение $R_{class}(h)$ и $R_{stab}(f)$ не убывает. Монотонность выступает условием согласованности (consistency condition) определения метрики с интуицией «усиления защиты».

Замечание 2.15. Монотонность $R_{class}(h)$ и $R_{stab}(f)$. Пусть h задает детерминированный оператор решения исходной модели, а h' соответствует модели с усиленной защитой. Тогда $R_{class}(h') \geqslant R_{class}(h)$ и $R_{stab}(f') \geqslant R_{stab}(f)$.

Доказательство. 1. Для $R_{class}(h)$:

Пусть h' есть оператор решения модели с усиленной защитой относительно h. Усиленная защита для $R_{class}(h)$ означает, что для любого входа x выполняется условие:

$$\sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h'(x) \neq h'(x')\} \leqslant \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h(x) \neq h(x')\}.$$

Отсюда непосредственно:

$$1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h'(x) \neq h'(x')\} \geqslant 1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h(x) \neq h(x')\}.$$

Взяв математическое ожидание по распределению $P_{\mathcal{X}}$, получаем:

$$\begin{aligned} R_{class}(h') &= \mathbb{E}_{x \sim P_{\mathcal{X}}}\Big[1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h'(x) \neq h'(x')\}\Big] \\ &\geqslant \mathbb{E}_{x \sim P_{\mathcal{X}}}\Big[1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h(x) \neq h(x')\}\Big] \\ &= R_{class}(h). \end{aligned}$$

Отсюда следует, что $R_{class}(h') \geqslant R_{class}(h)$.

2. Для $R_{stab}(f)$:
   Усиленная защита f' по отношению к f для $R_{stab}(f)$ означает, что для любого $x \in \mathcal{X}$ и соответствующих мер расхождения $D'_T(x,x')$ для f' и $D_T(x,x')$ для f, выполняется:
   $$\sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{D'_T(x,x')}{C_{max}} \leqslant \sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}}.$$
   Тогда:
   $$1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{D'_T(x,x')}{C_{max}} \geqslant 1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}}.$$
   Взяв математическое ожидание по $x \sim P_{\mathcal{X}}$ от обеих частей, получаем $R_{stab}(f') \geqslant R_{stab}(f)$.

□

Замечание 2.16. Об определении «усиленной защиты» и области применимости. Монотонность вытекает из поточечного определения «усиленной защиты» и линейности математического ожидания. Результат не является самостоятельной теоремой: он верифицирует, что определение метрики согласовано с интуицией. Без монотонности метрика не допускала бы сравнения моделей с различным уровнем защиты. Какие именно преобразования, включая фильтрацию входов, сглаживание выходов и ансамблирование, удовлетворяют условию поточечного уменьшения отклонения, устанавливается экспериментально в Главах 3–4.

2.3.2 Верхняя оценка уязвимости через устойчивость: $V \leqslant 1 - R_{class}$

Метрика $R_{class}(h)$ для моделей с дискретным выходом, где $h = \mathrm{Dec} \circ f$, связана с глобальной уязвимостью $V(h)$ через двуслойную модель. Если атака локализована ($a(x) \in \mathcal{B}_\varepsilon(x)$), выполняется неравенство $V(h) \leqslant 1 - R_{class}(h)$, Теорема 2.17. Нелокализованные атаки, такие как prompt injection и jailbreak, выходят за пределы $\mathcal{B}_\varepsilon(x)$, и для них строгая граница неприменима. Взамен предлагается эмпирическая регрессионная модель $V(h) \approx g(1 - R_{class}(h))$, параметры которой калибруются по данным (Замечание 2.20). Разделение на два режима продиктовано природой атак: теоретические гарантии возможны только при контролируемом бюджете возмущений.

Теорема 2.17. Верхняя оценка уязвимости через устойчивость. Пусть $h = \mathrm{Dec} \circ f$ есть детерминированный оператор решения и

$$R_{class}(h) = \mathbb{E}_{x \sim P_{\mathcal{X}}}\Big[1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h(x) \neq h(x')\}\Big],$$

$$V(h) = \mathbb{E}_{x \sim P_{\mathcal{X}},\, a \sim P_{\mathfrak{A}}}\big[\mathrm{Impact}\big(h(a(x))\big)\big],$$

где

- $(\mathcal{X}, P_{\mathcal{X}})$ – пространство и распределение входов,
- $\mathfrak{A}$ – множество атакующих преобразований $a : \mathcal{X} \to \mathcal{X}$ с распределением $P_{\mathfrak{A}}$,
- $\mathcal{B}_\varepsilon(x) = \{x' \in \mathcal{X} : d(x, x') \leqslant \varepsilon\}$, где метрика d определяет окрестность,
- $\mathrm{Impact}(y) = \mathbb{I}\{y \in \mathcal{Y}_{\mathrm{bad}}\}$ – измеримая функция воздействия, $\mathcal{Y}_{\mathrm{bad}}$ – множество нежелательных выходов,
- $\mathbb{I}\{\cdot\}$ – индикаторная функция.

Предположим

(A0) **Корректность на чистых входах:** $\mathbb{P}_{x \sim P_{\mathcal{X}}}(h(x) \in \mathcal{Y}_{\mathrm{bad}}) = 0$, т.е. модель не порождает нежелательных выходов на невозмущенных входах.

(A1) **Локализованность атак:** $a(x) \in \mathcal{B}_\varepsilon(x)$ почти наверное.

(A2) **Связь воздействия и изменения выхода:** событие $\mathrm{Impact}(h(a(x))) = 1$ влечет $h(x) \neq h(a(x))$, то есть нежелательный выход на возмущенном входе возможен лишь при отличии от выхода на невозмущенном входе.

Для индикаторной функции воздействия $\mathrm{Impact}(y) = \mathbb{I}\{y \in \mathcal{Y}_{\mathrm{bad}}\}$ условие (A2) выполняется автоматически при (A0): если $\mathrm{Impact}(h(a(x))) = 1$ и $h(x) \notin \mathcal{Y}_{\mathrm{bad}}$ по (A0), то $h(a(x)) \in \mathcal{Y}_{\mathrm{bad}}$ и $h(x) \neq h(a(x))$. Тогда выполняется неравенство

$$V(h) \;\leqslant\; 1 - R_{class}(h). \tag{2.1}$$

При ослаблении условия (A0), то есть если модель может порождать нежелательные выходы на чистых входах с вероятностью $\beta = \mathbb{P}_{x \sim P_{\mathcal{X}}}(h(x) \in \mathcal{Y}_{\mathrm{bad}})$, граница принимает вид:

$$V(h) \;\leqslant\; \beta + 1 - R_{class}(h). \tag{2.2}$$

Доказательство неравенства (2.1). По условию (A0) выполняется $h(x) \notin \mathcal{Y}_{\mathrm{bad}}$ почти наверное по $P_{\mathcal{X}}$. При (A0) и (A1):

$$\mathrm{Impact}(h(a(x))) \;\leqslant\; \mathbb{I}\big\{h(x) \neq h(a(x))\big\}.$$

Поскольку $a(x) \in \mathcal{B}_\varepsilon(x)$ (из A1), то

$$\mathbb{I}\{h(x) \neq h(a(x))\} \leqslant \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h(x) \neq h(x')\}.$$

Следовательно,

$$\mathrm{Impact}(h(a(x))) \leqslant \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h(x) \neq h(x')\}.$$

Усредняя по атакам $a \sim P_{\mathfrak{A}}$ и входам $x \sim P_{\mathcal{X}}$:

$$V(h) = \mathbb{E}_{x,a}[\mathrm{Impact}(h(a(x)))] \leqslant \mathbb{E}_x \left[\sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h(x) \neq h(x')\} \right] = 1 - R_{class}(h).$$

Доказательство схемы (2.2) при ослаблении условия (A0). Если условие (A0) не выполняется, разобьем математическое ожидание на два случая:

$$V(h) = \mathbb{E}_{x,a}\big[\mathrm{Impact}(h(a(x))) \cdot \mathbb{I}\{h(x) \notin \mathcal{Y}_{\text{bad}}\}\big] + \mathbb{E}_{x,a}\big[\mathrm{Impact}(h(a(x))) \cdot \mathbb{I}\{h(x) \in \mathcal{Y}_{\text{bad}}\}\big].$$

Первое слагаемое оценивается как выше: не более $1 - R_{class}(h)$. Второе слагаемое не превосходит $\mathbb{P}(h(x) \in \mathcal{Y}_{\text{bad}}) = \beta$, поскольку Impact $\leqslant 1$. Итого: $V(h) \leqslant \beta + 1 - R_{class}(h)$.

□

Замечание 2.18. О возможных обобщениях. Граница (2.1) может быть ослаблена для случаев, когда атаки лишь с некоторой вероятностью выходят за пределы $\mathcal{B}_\varepsilon(x)$ или когда условие (A2) выполняется в ослабленной форме. Такие обобщения приводят к более сложным калибруемым верхним оценкам и в настоящей работе не используются как строгие теоретические результаты.

Замечание 2.19. Область применимости границы. Неравенство (2.1) применимо к атакам, удовлетворяющим условиям (A1) и (A2): локализованным состязательным возмущениям, при которых $a(x) \in \mathcal{B}_\varepsilon(x)$. Нелокализованные атаки, в частности prompt injection и jailbreak, выходят за рамки этой границы. Для них в Замечании 2.20 предложена эмпирическая калибруемая модель. Двуслойная структура, описанная в начале Раздела 2.3.2, покрывает оба режима пространства атак и формирует рамку для интерпретации экспериментальных результатов Главы 4.

Замечание 2.20. Эмпирическая модель связи V и R_{class} вне (A1). При нарушении условий (A1) и (A2), например, для атак типа prompt injection, строгая верхняя граница (2.1) неприменима. Однако часто наблюдается монотонная зависимость между уязвимостью $V(h)$ и неустойчивостью $(1 - R_{class}(h))$, которую можно эмпирически аппроксимировать функцией вида:

$$V(h) \approx g(1 - R_{class}(h)), \quad \text{где} \quad g(t) = \min\{\, 1,\, k_{\text{eff}}\,(t + \mathbb{E}\eta)^{\alpha}\}.$$

Здесь $\mathbb{E}\eta$ выражает среднюю вероятность выхода атаки за пределы $\mathcal{B}_\varepsilon(x)$, а параметры $k_{\text{eff}} > 0$ и $\alpha \in (0, 1]$ калибруются на валидационном наборе атак. Соотношение следует рассматривать как эмпирическую калибруемую аппроксимацию, то есть гипотезу, пригодную для ориентировочной оценки уязвимости в пределах исследуемого класса моделей и атак, но не как универсальную строгую верхнюю границу.

Замечание 2.21. Эпистемологический статус эмпирической модели 2.20. В отличие от Теоремы 2.17, строго доказанной при выполнении условий (A0)–(A2), эмпирическая модель 2.20 представляет эмпирическую закономерность, то есть гипотезу: монотонная зависимость $V(h) \approx g(1-R_{class}(h))$ наблюдалась экспериментально, но не выводится дедуктивно из аксиом. Параметры k_{eff} и α калибруются по данным и могут варьироваться для различных семейств моделей и атак.

Замечание 2.22. О связи $R_{stab}(f)$ и $V(f)$. Теорема 2.17 доказана для метрики $R_{class}(h)$, где изменение выхода модели бинарно. Для генеративной метрики $R_{stab}(f)$, основанной на мере расхождения $D(p_1, p_2)$, прямой аналог неравенства $V(h) \leqslant 1 - R_{class}(h)$ отсутствует. Препятствие состоит в том, что условие (A2), связывающее Impact с изменением дискретного выхода, не допускает непосредственного переноса на непрерывную дивергенцию D_T. Косвенная связь, тем не менее, существует: если $D_T(x,x')$ мало для всех $x' \in \mathcal{B}_\varepsilon(x)$, вероятность генерации нежелательного выхода ограничена через неравенство Пинскера $\|p - q\|_{TV} \leqslant \sqrt{2D_{KL}(p\|q)}$. Построение строгой верхней границы $V(f)$ через $R_{stab}(f)$ остается открытой задачей. Эмпирическая связь между $R_{stab}(f)$ и $R_{class}(h)$ исследована в Разделе 4.2.5.

2.3.3 Лемма о композиции моделей

При последовательном применении нескольких LLM-компонентов устойчивость композиции определяется устойчивостью отдельных компонентов.

Лемма 2.23. О композиции моделей для R_{class}. Пусть $g\colon X \to Y$ и $f\colon Y \to Z$ – детерминированные модели, а композиция задается как $h = f \circ g\colon X \to Z$. Тогда для каждого входа $x \in X$ выполняется поточечное неравенство $S_h(x) \geqslant S_g(x)$, и, следовательно,

$$R_{class}(h) \geqslant R_{class}(g).$$

В частности, если для некоторого x первый компонент g устойчив на $\mathcal{B}_\varepsilon(x)$, то есть $g(x') = g(x)$ для всех $x' \in \mathcal{B}_\varepsilon(x)$, то композиция h также устойчива на этом входе: $h(x') = h(x)$ для всех $x' \in \mathcal{B}_\varepsilon(x)$.

Доказательство. Обозначим

$$S_h(x) = 1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \mathbb{I}\{h(x) \neq h(x')\} \in \{0{,}1\}.$$

Тогда $R_{class}(h) = \mathbb{E}_{x \sim P_{\mathcal{X}}}[S_h(x)]$.

Если $S_g(x) = 1$, то $g(x') = g(x)$ для всех $x' \in \mathcal{B}_\varepsilon(x)$. Следовательно, $(f \circ g)(x') = f\big(g(x')\big) = f\big(g(x)\big) = (f \circ g)(x)$, и потому $S_{f\circ g}(x) = 1$. Если же $S_g(x) = 0$, то тривиально $S_{f\circ g}(x) \geqslant 0 = S_g(x)$.

Следовательно, для всех $x \in X$ выполняется $S_{f\circ g}(x) \geqslant S_g(x)$. Переходя к математическому ожиданию, получаем $R_{class}(f \circ g) = \mathbb{E}[S_{f\circ g}(x)] \geqslant \mathbb{E}[S_g(x)] = R_{class}(g)$. □

Замечание 2.24. О стохастичности LLM. Лемма сформулирована для детерминированных моделей, что соответствует работе LLM при жадном декодировании, то есть argmax sampling. В общем случае LLM являются стохастическими. Расширение леммы на стохастический случай требует определения устойчивости для распределений, как в R_{stab}, и учета того, как стохастичность распространяется через композицию, что является более сложной задачей.

В детерминированном случае неравенство $R_{class}(h) \geqslant R_{class}(g)$ носит глобальный характер: второй компонент f способен лишь «склеить» различающиеся выходы g, но не «расщепить» совпадающие. Устойчивость композиции

не убывает. Вместе с тем неравенство представляет оценку наихудшего случая и предполагает детерминированность обоих компонентов при фиксированном направлении композиции $h = f \circ g$. Произвольное добавление компонентов к системе не гарантирует сохранения устойчивости: обратный порядок композиции, стохастическое декодирование или обратные связи между компонентами делают оценку неприменимой. Для стохастических моделей аналогичное утверждение требует отдельного обоснования, поскольку шум в f способен нарушить устойчивость даже при стабильном g. Связь этого результата с оценкой V_d для рекурсивных цепочек обсуждается в Замечании 2.34.

2.4 Аналитические оценки и верхние границы

Практическое использование метрик устойчивости предполагает наличие аналитических оценок и границ, позволяющих вычислить или аппроксимировать эти метрики. Раздел охватывает верхние границы для R_{class} при ограниченном ε, уязвимость к адаптивным атакам и асимптотические свойства оценок при растущем размере датасета.

2.4.1 Верхние границы $R_{class}(h)$ при ограниченном ε

Рассмотрим верхние границы метрики устойчивости $R_{class}(h)$ при ограниченном радиусе возмущений ε.

Теорема 2.25. Верхняя граница $R_{class}(h)$. Пусть $h = \mathrm{Dec} \circ f$ задает детерминированный оператор решения, отображающий из пространства входов $\mathcal{X}$, оснащенного метрикой d, в дискретное пространство выходов. Тогда при ограниченном ε:

$$R_{class}(h) \leqslant 1 - \mathbb{P}(h(X) \neq h(X') \text{ и } d(X, X') \leqslant \varepsilon)$$

где X и X' - независимые случайные величины с одинаковым распределением $P_{\mathcal{X}}$ на $\mathcal{X}$.

Доказательство. По определению $R_{class}(h)$:

$$R_{class}(h) = \mathbb{E}_{X \sim P_{\mathcal{X}}} \left[1 - \sup_{x_0 \in \mathcal{B}_\varepsilon(X)} \mathbb{I}\{h(X) \neq h(x_0)\} \right].$$

Обозначим событие $A_X^c = \{\exists x_0 \in \mathcal{B}_\varepsilon(X) : h(X) \neq h(x_0)\}$. Тогда $R_{class}(h) = 1 - \mathbb{P}_X(A_X^c)$. Рассмотрим событие $E = \{(X,X') \mid h(X) \neq h(X') \text{ и } d(X,X') \leqslant \varepsilon\}$. Если событие E произошло для реализаций (x, x'), то $x_0 = x'$ лежит в $\mathcal{B}_\varepsilon(x)$ и $h(x) \neq h(x_0)$, а значит A_x^c истинно. Событие E влечет A_X^c для первого элемента пары, откуда $\mathbb{P}(E) \leqslant \mathbb{P}_X(A_X^c)$. Тогда $1 - \mathbb{P}_X(A_X^c) \leqslant 1 - \mathbb{P}(E)$, что дает

$$R_{class}(h) \leqslant 1 - \mathbb{P}(h(X) \neq h(X') \text{ и } d(X,X') \leqslant \varepsilon).$$

что и требовалось доказать. □

Замечание 2.26. Интерпретация верхней границы $R_{class}(h)$. Неравенство $R_{class}(h) \leqslant 1 - \mathbb{P}_{X,X'}(E)$ допускает наглядную трактовку. Чем чаще два случайных входа на расстоянии не более ε дают различные выходы, тем ниже верхняя граница устойчивости. Центральный шаг доказательства: соотношение $\mathbb{P}_X(A_X^c) \geqslant \mathbb{P}_{X,X'}(E)$. Оно означает, что наихудшее возмущение в $\mathcal{B}_\varepsilon(x)$ срабатывает не реже случайного.

2.4.2 Оценка уязвимости к адаптивным атакам

Адаптивные атаки подстраиваются под конкретную модель и выбирают наилучшее воздействие для каждого входа.

Определение 2.27. Уязвимость к поточечно-оптимальной адаптивной атаке. Пусть $\mathfrak{A}_{\text{adap}}$ - множество адаптивных атак, способных использовать информацию о модели f.

Уязвимость к поточечно-оптимальной адаптивной атаке $V_{\text{opt\_adapt}}(f)$ определяется как:

$$V_{\text{opt\_adapt}}(f) = \mathbb{E}_{x \in \mathcal{X}} \big[\max_{a \in \mathfrak{A}_{\text{adap}}} \text{Impact}(f(a(x))) \big]$$

где $\text{Impact}(y) = \mathbb{I}\{y \in \mathcal{Y}_{\text{bad}}\}$.

Величина $V_{\text{opt\_adapt}}(f)$ моделирует атакующего, способного подобрать оптимальную стратегию из $\mathfrak{A}_{\text{adap}}$ отдельно для каждого входа x. При фиксированном распределении атак $P_{\mathfrak{A}}$, не зависящем от x, средняя уязвимость $V(f) = \mathbb{E}_{a\in\mathfrak{A}}\mathbb{E}_{x\in\mathcal{X}}[\text{Impact}(f(a(x)))]$ не превышает $V_{\text{opt\_adapt}}(f)$. Соотношение с наихудшей фиксированной атакой $V_{\text{worst}}(f) = \sup_{a\in\mathfrak{A}} \mathbb{E}_x[\text{Impact}(f(a(x)))]$ менее прямолинейно. По неравенству Йенсена максимум внутри математического ожидания не меньше максимума снаружи: $V_{\text{opt\_adapt}}(f) \geqslant \max_{a\in\mathfrak{A}_{\text{adap}}} \mathbb{E}_x[\text{Impact}(f(a(x)))]$.

Замечание 2.28. Метрика $V_{\text{opt\_adapt}}(f)$ полезна для оценки «потолка» уязвимости, когда атакующий обладает значительными адаптивными возможностями. Сравнение $V_{\text{opt\_adapt}}(f)$ с $V(f)$ для некоторого распределения $P_{\mathfrak{A}}$ на атаках или с $V_{\text{worst}}(f)$ может дать представление о том, насколько выигрыш атакующего увеличивается за счет адаптации к каждому входу.

Отсюда следует, что уязвимость модели к адаптивным атакам ограничена снизу ожидаемым максимальным воздействием по всему набору доступных адаптивных стратегий.

2.4.3 Асимптотические свойства при размере датасета $\to \infty$

При увеличении объема датасета для эмпирической оценки метрик устойчивости существенную роль приобретают асимптотические свойства получаемых оценок.

Утверждение 2.29. Сходимость эмпирической оценки $\hat{R}_{class}$ к R_{class}. Пусть $\hat{R}_{class,n}(h)$ задает эмпирическую оценку метрики устойчивости $R_{class}(h)$ на основе n независимых образцов:

$$\hat{R}_{class,n}(h) = \frac{1}{n}\sum_{i=1}^{n}\Big[1 - \sup_{x'\in\mathcal{B}_\varepsilon(x_i)} \mathbb{I}\{h(x_i) \neq h(x')\}\Big]$$

Тогда при $n \to \infty$:

$$\hat{R}_{class,n}(h) \xrightarrow{P} R_{class}(h)$$

где $\xrightarrow{P}$ обозначает сходимость по вероятности.

Доказательство. Обозначим $Z_i = 1 - \sup_{x' \in \mathcal{B}_\varepsilon(x_i)} \mathbb{I}\{h(x_i) \neq h(x')\}$. Величины Z_i являются независимыми одинаково распределенными случайными величинами с $\mathbb{E}[Z_i] = R_{class}(h)$ и $Z_i \in [0, 1]$.

По закону больших чисел:

$$\hat{R}_{class,n}(h) = \frac{1}{n} \sum_{i=1}^{n} Z_i \xrightarrow{P} \mathbb{E}[Z_i] = R_{class}(h)$$

Более того, по центральной предельной теореме:

$$\sqrt{n}(\hat{R}_{class,n}(h) - R_{class}(h)) \xrightarrow{d} \mathcal{N}(0, \sigma^2)$$

где $\sigma^2 = \mathrm{Var}(Z_i)$ и $\xrightarrow{d}$ обозначает сходимость по распределению.

Отсюда следует, что $\hat{R}_{class,n}(h) \xrightarrow{P} R_{class}(h)$ при $n \to \infty$, что и требовалось доказать. Аналогичные утверждения справедливы и для эмпирических оценок $R_{stab}(f)$ и $V(f)$ при соответствующих условиях на моменты случайных величин. □

Утверждение является прямым следствием закона больших чисел для ограниченных случайных величин и гарантирует, что при достаточном объеме датасета эмпирическая оценка метрики устойчивости сколь угодно точно приближает истинное значение.

2.5 Устойчивость композиционных систем

Системы на основе больших языковых моделей нередко объединяют несколько взаимодействующих компонентов. Раздел посвящен оценке устойчивости пайплайнов LLM с retriever-компонентами, рекурсивных цепочек LLM-агентов и механизмов vote-ensemble.

2.5.1 Пайплайны LLM + retriever

Пайплайны, сочетающие LLM с компонентами извлечения информации, retriever, распространены в системах вопросно-ответного типа. Устойчивость таких систем определяется робастностью как LLM, так и retriever-компонента.

Утверждение 2.30. Композиционная устойчивость LLM + retriever. Пусть $r : \mathcal{X} \to 2^{\mathcal{D}}$ - retriever-компонент, извлекающий релевантные документы из базы данных $\mathcal{D}$, удовлетворяющий условию Липшица:

$$\forall x, x' \in \mathcal{X} : d_D(r(x), r(x')) \leqslant K_r \cdot d_X(x, x')$$

где $K_r > 0$ - константа Липшица для retriever, d_X - метрика на пространстве запросов, d_D - метрика на пространстве наборов документов.

Пусть $f : \mathcal{X} \times 2^{\mathcal{D}} \to \mathcal{Y}$ - языковая модель, генерирующая ответ на основе запроса и извлеченных документов, удовлетворяющая условиям Липшица:

$$\forall x \in \mathcal{X}, \forall D_1, D_2 \in 2^{\mathcal{D}} :$$
$$d_Y(f(x, D_1), f(x, D_2)) \leqslant K_{f,D} \cdot d_D(D_1, D_2)$$

$$\forall x_1, x_2 \in \mathcal{X}, \forall D \in 2^{\mathcal{D}} :$$
$$d_Y(f(x_1, D), f(x_2, D)) \leqslant K_{f,X} \cdot d_X(x_1, x_2)$$

где $K_{f,D}, K_{f,X} > 0$ - константы Липшица для LLM по документам и запросу, d_Y - метрика на выходах.

Тогда композиционная система $s(x) = f(x, r(x))$ удовлетворяет условию Липшица с константой $K_s \leqslant K_{f,D} \cdot K_r + K_{f,X}$:

$$\forall x, x' \in \mathcal{X} : d_Y(s(x), s(x')) \leqslant (K_{f,D} \cdot K_r + K_{f,X}) \cdot d_X(x, x')$$

Доказательство. Рассмотрим два произвольных запроса $x, x' \in \mathcal{X}$. Оценим расстояние между соответствующими выходами системы:

$$\begin{aligned} d_Y(s(x), s(x')) &= d_Y(f(x, r(x)), f(x', r(x'))) \\ &\leqslant d_Y(f(x, r(x)), f(x, r(x'))) + d_Y(f(x, r(x')), f(x', r(x'))) \\ &\quad \text{(неравенство треугольника)} \end{aligned}$$

По условию Липшица для f по второму аргументу, то есть документам:

$$d_Y(f(x, r(x)), f(x, r(x'))) \leqslant K_{f,D} \cdot d_D(r(x), r(x')).$$

По условию Липшица для r:

$$d_D(r(x), r(x')) \leqslant K_r \cdot d_X(x, x').$$

Следовательно, первое слагаемое не превосходит $K_{f,D} \cdot K_r \cdot d_X(x, x')$.

По условию Липшица для f по первому аргументу, то есть запросу:

$$d_Y(f(x, r(x')), f(x', r(x'))) \leqslant K_{f,X} \cdot d_X(x, x').$$

Тогда:

$$\begin{aligned} d_Y(s(x), s(x')) &\leqslant K_{f,D} \cdot K_r \cdot d_X(x, x') + K_{f,X} \cdot d_X(x, x') \\ &= (K_{f,D} \cdot K_r + K_{f,X}) \cdot d_X(x, x') \end{aligned}$$

Отсюда следует, что система $s(x) = f(x, r(x))$ удовлетворяет условию Липшица с константой $K_s \leqslant K_{f,D} \cdot K_r + K_{f,X}$, что и требовалось доказать. □

Замечание 2.31. Об операционализации оценки для композитных систем. Применение Утверждения 2.30 на практике требует выбора метрики d_D и оценки констант Липшица.

Метрика d_D. Расстояние Жаккара $d_D(D_1, D_2) = 1 - |D_1 \cap D_2|/|D_1 \cup D_2|$ учитывает только состав наборов документов. Когда различия носят семантический характер, предпочтительнее расстояние на основе эмбеддингов: косинусная мера между усредненными векторами наборов или Earth Mover's Distance между множествами эмбеддингов.

Константы Липшица. Аналитический расчет K_r, $K_{f,D}$, $K_{f,X}$ для нейронных сетей, как правило, невозможен. Альтернатива: эмпирическая оценка на валидационном наборе $\mathcal{D}_{val}$. Для retriever-компонента генерируются пары близких запросов (x_i, x_i') и вычисляется отношение $\rho_i = d_D(r(x_i), r(x_i'))/d_X(x_i, x_i')$. Оценка $\hat{K}_r$ берется как максимум ρ_i или 99-й перцентиль для устойчивости к выбросам. Константы $\hat{K}_{f,D}$ и $\hat{K}_{f,X}$ оцениваются аналогично на соответствующих пространствах. Итоговая граница для композиционной системы: $\hat{K}_s \leqslant \hat{K}_{f,D} \cdot \hat{K}_r + \hat{K}_{f,X}$. Компонент с наибольшей эмпирической константой указывает на узкое место пайплайна.

2.5.2 Рекурсивные цепочки LLM-агентов

В рекурсивных цепочках LLM-агентов выход одного агента становится входом для другого, что порождает риск накопления и усиления ошибок.

Утверждение 2.32. Верхняя оценка уязвимости в рекурсивных цепочках при модели каскадного отказа. Пусть V_1 - показатель уязвимости, то есть вероятность некорректного результата при атаке, одиночного LLM-агента. Введем индикаторы ошибки $E_i \in \{0,1\}$, где $E_i = 1$ означает некорректный результат i-го агента под атакой. Рассмотрим **модель каскадного фатального отказа** при следующих допущениях:

(C1) **Независимость ошибок:** индикаторы $E_1, \ldots, E_d$ моделируются как статистически независимые случайные величины на общем вероятностном пространстве атаки;

(C2) **Однородность:** все d агентов имеют одинаковую вероятность ошибки $\mathbb{P}(E_i = 1) = V_1$;

(C3) **Невозможность компенсации:** результат цепочки некорректен тогда и только тогда, когда хотя бы один агент выдал некорректный результат, то есть последующие звенья не компенсируют ошибку предшествующих.

При выполнении допущений (C1)–(C3) показатель уязвимости цепочки из d последовательных агентов равен:

$$V_d = 1 - (1 - V_1)^d.$$

Если допущение (C3) ослаблено до одностороннего условия, а именно корректность всех агентов влечет корректность цепочки, но возможна частичная компенсация ошибок, то при сохранении (C1)–(C2) равенство переходит в верхнюю оценку:

$$V_d \leqslant 1 - (1 - V_1)^d.$$

Доказательство. Рассмотрим цепочку из d последовательных LLM-агентов $f_1, f_2, \ldots, f_d$, где выход f_i является входом для f_{i+1}, а композиция задается функцией $F_d = f_d \circ f_{d-1} \circ \ldots \circ f_1$. Обозначим $A = \{$результат цепочки некорректен$\}$ и $B = \{E_1 = 1$ или $\ldots$ или $E_d = 1\}$, событие ошибки хотя бы одного агента.

По независимости (C1) и однородности (C2) вероятность того, что ни один агент не ошибся, равна $\mathbb{P}(\overline{B}) = \prod_{i=1}^{d}(1 - V_1) = (1 - V_1)^d$, откуда $\mathbb{P}(B) = 1 - (1 - V_1)^d$.

При полном условии (C3) события A и B совпадают, поэтому $V_d = \mathbb{P}(A) = \mathbb{P}(B) = 1 - (1 - V_1)^d$.

При ослабленном (C3) сохраняется включение $A \subseteq B$: если ни один агент не ошибся, результат цепочки корректен, то есть $\overline{B} \subseteq \overline{A}$. Отсюда $V_d = \mathbb{P}(A) \leqslant \mathbb{P}(B) = 1 - (1 - V_1)^d$.

Для неоднородной цепочки с уязвимостями $V_1^{(i)}$ аналогично $V_d = 1 - \prod_{i=1}^{d}(1 - V_1^{(i)})$ при полном (C3), и если $V_1^{(i)} \leqslant V_1$ для всех i, то $V_d \leqslant 1 - (1 - V_1)^d$. □

С ростом глубины цепочки уязвимость растет, что предъявляет жесткие требования к устойчивости каждого отдельного агента.

Замечание 2.33. Область применимости оценки $V_d \leqslant 1 - (1 - V_1)^d$. Оценка опирается на упрощающие допущения (C1)–(C3), которые на практике выполняются лишь приближенно. Модели одного семейства нередко разделяют систематические смещения, что порождает корреляцию ошибок, нарушая условие C1: при положительной корреляции фактическая уязвимость может отклоняться от оценки в обе стороны. Неоднородность агентов, нарушающая условие C2, приводит к выражению $V_d = 1 - \prod_{i=1}^{d}(1 - V_1^{(i)})$. Если $V_1^{(i)} \leqslant V_1$ для всех i, оно сводится к $V_d \leqslant 1 - (1 - V_1)^d$. Между тем последующие агенты способны частично компенсировать ошибки предыдущих, что нарушает условие C3, понижая фактическую уязвимость ниже верхней границы. Формула $V_d \leqslant 1 - (1 - V_1)^d$ представляет грубую верхнюю оценку при допущении каскадного отказа и независимости ошибок, а не универсальный закон для произвольных агентских цепочек.

Замечание 2.34. Соотношение леммы о композиции и оценки V_d. Лемма 2.23 утверждает, что устойчивость композиции $R_{class}(f \circ g) \geqslant R_{class}(g)$ не убывает при добавлении детерминированного компонента. Оценка $V_d \leqslant 1-(1-V_1)^d$ из Утверждения 2.32, напротив, показывает рост уязвимости с глубиной цепочки. Противоречия между этими результатами нет. Лемма описывает свойство одной фиксированной композиции относительно ее первого компонента, тогда как оценка V_d моделирует вероятность отказа хотя бы одного звена в цепочке из d независимых агентов. Первый результат характеризует структурную монотонность устойчивости, второй – накопление вероятности ошибки в стохастическом сценарии.

2.5.3 Механизмы vote-ensemble

Мажоритарное голосование нескольких моделей, vote-ensemble, позволяет повысить устойчивость LLM-систем, прежде всего в задачах классификации с дискретным выходом. Комитеты моделей показали эффективность и в повышении устойчивости LLM-as-a-Judge систем к инъекциям запросов.

Теорема 2.35. Устойчивость vote-ensemble при наличии устойчивого большинства с общим вердиктом. Пусть $f_1, \ldots, f_k$ суть k моделей с дискретным выходом, k нечетно, а F_k задает механизм мажоритарного голосования. Для входа x обозначим через $v_i(x) = f_i(x)$ вердикт i-й модели и через

$$S_i(x) = \mathbb{I}\{f_i(x) = f_i(x') \text{ для всех } x' \in \mathcal{B}_\varepsilon(x)\}.$$

Предположим, что существует метка $c(x)$ такая, что не менее чем для $\lceil k/2 \rceil$ моделей одновременно выполняются:

1. $S_i(x) = 1$;
2. $f_i(x) = c(x)$.

Тогда

$$F_k(x') = c(x) \qquad \text{для всех } x' \in \mathcal{B}_\varepsilon(x),$$

то есть ансамбль устойчив на входе x.

Доказательство. По предположению существует не менее $\lceil k/2 \rceil$ моделей, которые на всем множестве $\mathcal{B}_\varepsilon(x)$ сохраняют один и тот же вердикт $c(x)$. Следовательно, для любого $x' \in \mathcal{B}_\varepsilon(x)$ не менее половины моделей выдают $c(x)$. Поскольку k нечетно, мажоритарное голосование однозначно возвращает $c(x)$ для всех $x' \in \mathcal{B}_\varepsilon(x)$. Теорема доказана. □

Следствие 2.36. Вероятностная оценка устойчивости комитета как следствие теоремы Кондорсе о жюри. Пусть $c^*(x)$ обозначает истинный вердикт на входе x, а $f_1, \ldots, f_k$ суть k моделей при нечетном k, каждая из которых дает вердикт $f_i(x)$, совпадающий с $c^*(x)$ с вероятностью $r > 0{,}5$. Пусть ошибки моделей независимы в совокупности. Тогда вероятность того, что мажоритарное голосование комитета дает $c^*(x)$, удовлетворяет:

$$P_{ens}(k,r) = \sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i} r^i (1-r)^{k-i} \xrightarrow[k\to\infty]{k \text{ нечетно}} 1 \quad \text{при } r > 0{,}5.$$

Доказательство. Число верных вердиктов $S_k = \sum_{i=1}^{k} \mathbb{I}\{f_i(x) = c^*(x)\}$ имеет распределение $\mathrm{Bin}(k, r)$. При $r > 0{,}5$ по закону больших чисел $S_k/k \xrightarrow{P} r > 1/2$, откуда $\mathbb{P}(S_k \geqslant \lceil k/2 \rceil) \to 1$. □

Замечание 2.37. Если Теорема 2.35 гарантирует устойчивость при наличии детерминистического устойчивого большинства, то Следствие 2.36 раскрывает вероятностный механизм повышения точности комитета с ростом его размера. Независимость вердиктов здесь является идеализацией. На пересекающихся обучающих данных модели приобретают общие смещения, и полная независимость недостижима. Между тем экспериментальные данные Главы 4 свидетельствуют о том, что гетерогенные комитеты сохраняют достаточно низкую корреляцию ошибок для практической эффективности метода.

Механизм мажоритарного голосования способствует повышению устойчивости системы в смысле R_{class}, причем выигрыш нарастает с увеличением числа моделей k при базовой устойчивости $r > 0{,}5$.

Замечание 2.38. Применимость теоремы для $R_{stab}(f)$. Доказанная теорема строго применима к $R_{class}(h)$, где выходы моделей дискретны и однозначно сравнимы. Перенос на метрику $R_{stab}(f)$ для генеративных моделей опирается на совместную выпуклость дивергенции Йенсена–Шеннона по паре распределений как f-дивергенции, что установлено в Замечании 3.5 Главы 3. Из совместной выпуклости следует, что попарное расхождение D_T при усреднении распределений нескольких моделей не превышает среднего попарного расхождения участников. Экспериментальные данные Главы 4 подтверждают, что ансамблирование повышает робастность систем LLM-as-a-Judge, где агрегируются итоговые вердикты.

2.6 Сравнение с существующими метриками

Предложенные метрики устойчивости целесообразно сопоставить с существующими метриками из области оценки качества и безопасности языковых моделей.

2.6.1 Критерии полноты

Полнота и применимость метрик устойчивости верифицируются по следующим критериям:

- **Инвариантность:** Метрика инвариантна к несущественным преобразованиям модели или данных.
- **Чувствительность:** Изменения, влияющие на устойчивость модели, отражаются в значениях метрики.
- **Вычислимость:** Метрика допускает эффективное вычисление на практике.

Сравним предложенные метрики $R_{stab}(f)$ и $V(f)$ с существующими метриками, такими как BLEU- и ROUGE-битовые аналоги и Perplexity-based Detox:

Метрика	**Инвар.**	**Чувств.**	**Вычисл.**
$R_{stab}(f)$	Высокая	Высокая	Средн./Выс.
$V(f)$	Высокая	Высокая	Высокая
BLEU/ROUGE-подобные метрики	Средняя	Низкая	Высокая
Perplexity-based Detox	Низкая	Средняя	Высокая

Таблица 4 — Сравнение метрик по критериям полноты.

Таблица 4 показывает, что предложенные метрики $R_{stab}(f)$ и $V(f)$ превосходят аналоги по инвариантности и чувствительности. При этом их точное вычисление требует большего объема ресурсов, поскольку предполагает либо исчерпывающий поиск для sup, либо симуляцию множества атак.

2.6.2 Применение метрик: сравнение моделей

Методология расчета метрик R_{stab}, V и Risk иллюстрируется на реальных экспериментальных данных в Главе 4.

2.6.3 Связь теоретической метрики $R_{stab}(f)$ с эмпирическими показателями

Метрика $R_{stab}(f)$ определена через пошаговое расхождение вероятностных распределений и, в общем случае, требует доступа к внутренним распределениям модели на каждом шаге генерации. Для проприетарных моделей (GPT-4, Claude-3-Opus) такой доступ недоступен, что делает прямое вычисление $R_{stab}(f)$ невозможным в условиях black-box. Даже для моделей с открытыми весами прямой расчет $D_T(x,x')$ для всех $x' \in \mathcal{B}_\varepsilon(x)$ вычислительно затратен.

В связи с этим в экспериментальной части работы, Глава 4, используется эмпирическая оценка

$$\widehat{R}_{emp}(f) = 1 - \mathrm{ASR}(f, \mathfrak{A}_{eval}),$$

выступающая практическим прокси устойчивости $R_{class}(h)$, а через нее и $R_{stab}(f)$, относительно фиксированного семейства атак $\mathfrak{A}_{eval}$.

Утверждение 2.39. Обоснование $\widehat{R}_{emp}$ как прокси R_{class}. Для задач с дискретным выходом, включая LLM-as-a-Judge с конечным множеством вердиктов, пусть выполнены условия:

1. атаки $a \in \mathfrak{A}_{eval}$ генерируются i.i.d. по распределению $P_{\mathfrak{A}}$, а входы x_i – i.i.d. по $P_{\mathcal{X}}$;
2. успех атаки определяется индикатором $\mathrm{Impact}(h(a(x))) = \mathbb{I}\{h(a(x)) \in \mathcal{Y}_{\text{bad}}\}$ из Теоремы 2.17;
3. выполнены предположения (A0)–(A1) той же теоремы.

Тогда оценка $\widehat{R}_{emp}(f) = 1 - \mathrm{ASR}$ состоятельна для популяционной величины $1 - V(h)$, и для популяционного предела справедливо $1 - V(h) \geqslant R_{class}(h)$. Для конечной выборки объема n неравенство $\widehat{R}_{emp}(f) \geqslant R_{class}(h)$ выполняется с точностью до случайной погрешности порядка $O_p(n^{-1/2})$.

Доказательство. По определению $\mathrm{ASR} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Impact}(h(a_i(x_i)))$ есть среднее по i.i.d. ограниченным слагаемым с математическим ожиданием $V(h) = \mathbb{E}[\mathrm{Impact}(h(a(x)))]$. По усиленному закону больших чисел $\mathrm{ASR} \to V(h)$ почти наверное, откуда $\widehat{R}_{emp}(f) = 1 - \mathrm{ASR} \to 1 - V(h)$, что доказывает состоятельность. По Теореме 2.17 при (A0)–(A1) выполнено $V(h) \leqslant 1 - R_{class}(h)$, то есть

$1 - V(h) \geqslant R_{class}(h)$ для популяционных величин. Отклонение $\widehat{R}_{emp}(f)$ от своего предела по центральной предельной теореме имеет порядок $O_p(n^{-1/2})$, что и дает заявленную конечновыборочную оценку. □

Утверждение дает теоретическую интерпретацию эмпирических результатов Главы 4: популяционный предел $1 - \mathrm{ASR}$ ограничивает устойчивость $R_{class}(h)$ снизу, а сам ASR служит состоятельной оценкой уязвимости $V(h)$ относительно семейства $\mathfrak{A}_{eval}$. Для нелокализованных атак типа prompt injection связь устанавливается через калибруемую модель, см. Замечание 2.20.

2.7 Выводы по главе

Задача оценки устойчивости LLM формализована через определение пространства входов, пространства ответов, семейства атак и множеств целей и последствий. Разграничены концепции устойчивости к целенаправленным атакам $\mathrm{AdvRobustness}(f)$ и стабильности к малым возмущениям $R_{stab}(f)$. Предложены формальные метрики: $R_{class}(h)$ для классификационных задач, $R_{stab}(f)$ для генеративных моделей, $V(f)$ для глобальной уязвимости и $\mathrm{Risk}(a, f)$ для количественного определения риска от конкретной атаки. Для практического применения уточнены параметры нормализации C_{max} и выбор меры расхождения D.

Верифицирована монотонность $R_{class}(h)$ и $R_{stab}(f)$ относительно усиления защиты как условие согласованности определений. Доказана взаимная ограниченность R_{class} и V по Теореме 2.17 при допущении отсутствия нежелательных выходов на невозмущенных входах (условие A0, ослабленное до произвольного базового уровня β), а также свойство композиции моделей для R_{class}. Получены аналитические верхние границы для $R_{class}(h)$, подтверждена состоятельность эмпирических оценок как следствие ЗБЧ для ограниченных случайных величин.

Для композиционных систем получены оценки устойчивости пайплайнов LLM + retriever через константу Липшица, верхняя граница уязвимости рекурсивных цепочек агентов и условия устойчивости vote-ensemble для R_{class}.

Предложенный аппарат имеет несколько ограничений. Точный расчет метрик для крупных моделей требует значительных вычислительных ресурсов. Строгая граница $V \leqslant 1 - R_{class}$ охватывает только локализованные атаки, тогда как перенос результатов на метрику $R_{stab}(f)$ для генеративных моделей остается открытой задачей. Выбор адекватных мер расстояния в дискретном пространстве токенов представляет самостоятельную нерешенную проблему. Экспериментальная валидация предложенных метрик и их применение к реальным моделям составляют содержание Главы 4.

Глава 3. Методы повышения устойчивости больших языковых моделей

Введение к главе

Локальная чувствительность модели к входным возмущениям определяет ее устойчивость. В настоящей главе для метрики $R_{stab}(f)$ установлены необходимые и достаточные условия, связывающие локальные свойства модели с глобальной робастностью. Опираясь на эти результаты, глава систематизирует архитектурные модификации, методы обучения и многоуровневые защитные механизмы, действующие в дискретном пространстве токенов. Рассмотрен также компромисс между устойчивостью и функциональной полезностью модели.

Известные подходы изложены в Разделах 3.1–3.3. Раздел 3.4 представляет методы, предложенные автором диссертации.

3.1 Теоретические основы повышения устойчивости

Раздел посвящен формализации центральных концепций робастности с учетом специфики больших языковых моделей, оперирующих в дискретном пространстве токенов $\mathcal{X} = \mathcal{V}^*$. Здесь прослеживается взаимосвязь локальной чувствительности модели с ее общей устойчивостью, введенной в Главе 1 (Раздел 1.1) и Главе 2 (Раздел 2.2.2).

3.1.1 Формализация устойчивости и локальной чувствительности

Метрика устойчивости $R_{stab}(f)$ для генеративных моделей введена в Определении 2.3 Главы 2. Во всех теоретических утверждениях настоящей главы расхождение выходов понимается как $D_T(x,x')$ из того же определения. Сокращенная запись $D\big(p(\cdot \mid x), p(\cdot \mid x')\big)$ далее не применяется во

избежание неоднозначности между пошаговым и полным распределением на последовательностях. Значение $R_{stab}(f)$, близкое к 1, соответствует высокой устойчивости.

Локальная чувствительность модели отражает степень изменения выходного распределения под воздействием малых входных возмущений. Применительно к дискретным пространствам роль аналога константы Липшица выполняет максимальный сдвиг меры расхождения D при наименьших допустимых изменениях входа, в частности при $d(x,x') = 1$ в метрике Левенштейна.

Лемма 3.1. Необходимое условие устойчивости. Если модель $f : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ имеет высокую устойчивость $R_{stab}(f) \approx 1$ согласно Определению 2.3, то среднее по распределению данных максимальное изменение выходного распределения при малых возмущениях ограничено.

Формально, если $R_{stab}(f) = 1 - \delta_R$ для малого $\delta_R > 0$, то среднее максимальное нормированное расхождение ограничено:

$$\mathbb{E}_{x\sim\mathcal{D}}\left[\sup_{x'\in\mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}}\right] = \delta_R,$$

следовательно

$$\mathbb{E}_{x\sim\mathcal{D}}\left[\sup_{x'\in\mathcal{B}_\varepsilon(x)} D_T(x,x')\right] \leqslant C_{max}\cdot\delta_R.$$

Доказательство. По определению

$$R_{stab}(f) = \mathbb{E}_{x\sim\mathcal{D}}\left[1 - \sup_{x'\in\mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}}\right].$$

Если $R_{stab}(f) = 1 - \delta_R$, то, перенося члены, получаем

$$\mathbb{E}_{x\sim\mathcal{D}}\left[\sup_{x'\in\mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}}\right] = \delta_R.$$

Поскольку $D(\cdot,\cdot) \geqslant 0$ и $C_{max} > 0$, то и

$$\sup_{x'\in\mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}} \geqslant 0.$$

Из

$$\mathbb{E}_{x\sim\mathcal{D}}\left[\sup_{x'\in\mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}}\right] = \delta_R,$$

и так как подынтегральное выражение неотрицательно, следует, что для доли входов x не менее $1-\delta$ по мере $\mathcal{D}$ величина $\sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}}$ не превышает $\frac{\delta_R}{\delta}$ при любом $\delta \in (0,1)$ по неравенству Маркова.

Формально, применяя неравенство Маркова к неотрицательной случайной величине $\sup_{x' \in \mathcal{B}_\varepsilon(x)} D_T(x,x')$:

$$\mathbb{P}\left(\sup_{x' \in \mathcal{B}_\varepsilon(x)} D_T(x,x') > \tau \right) \leqslant \frac{C_{\max}\delta_R}{\tau},$$

что при малых δ_R гарантирует, что доля входов с высокой локальной чувствительностью ограничена сверху.

Если бы для значимой доли x локальная чувствительность была высока, а именно $\sup_{x' \in \mathcal{B}_\varepsilon(x)} D_T(x,x')$ было значительно больше $C_{max} \cdot \delta_R$, то и $\sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}}$ было бы значительно больше δ_R. Это привело бы к математическому ожиданию, большему δ_R, что противоречит условию.

Следовательно, в среднем по $\mathcal{D}$ выходное распределение $p(\cdot|x)$ нечувствительно к малым возмущениям $x' \in \mathcal{B}_\varepsilon(x)$, и

$$\begin{aligned}\mathbb{E}_{x\sim\mathcal{D}}\Big[\sup_{x' \in \mathcal{B}_\varepsilon(x)} D_T(x,x') \Big] &= C_{max} \cdot \mathbb{E}_{x\sim\mathcal{D}}\left[\sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}} \right] \\ &= C_{max} \cdot \delta_R.\end{aligned}$$

□

Утверждение 3.2. О достаточном условии устойчивости. Пусть $f : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ - языковая модель. Если существует константа $L > 0$ такая, что для всех $x \in \mathcal{X}$ и всех $x' \in \mathcal{B}_\varepsilon(x)$ выполняется условие ограниченной локальной чувствительности:

$$D_T(x,x') \leqslant L \cdot d(x, x'),$$

тогда устойчивость модели $R_{stab}(f)$ ограничена снизу:

$$R_{stab}(f) \geqslant 1 - \frac{L \cdot \varepsilon}{C_{max}}.$$

Доказательство. По определению $R_{stab}(f) = \mathbb{E}_{x\sim\mathcal{D}}[1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{D_T(x,x')}{C_{max}}]$. Из условия теоремы, для любого x и любого $x' \in \mathcal{B}_\varepsilon(x)$, имеем $D_T(x,x') \leqslant L \cdot d(x, x')$. Поскольку $x' \in \mathcal{B}_\varepsilon(x)$, то $d(x, x') \leqslant \varepsilon$. Следовательно, $D_T(x,x') \leqslant L \cdot \varepsilon$. Это неравенство выполняется для всех $x' \in \mathcal{B}_\varepsilon(x)$, поэтому:

$$\sup_{x' \in \mathcal{B}_\varepsilon(x)} D_T(x,x') \leqslant L \cdot \varepsilon.$$

Подставляя это в определение $R_{stab}(f)$:

$$R_{stab}(f) = \mathbb{E}_{x\sim\mathcal{D}}\left[1 - \frac{\sup_{x'\in\mathcal{B}_\varepsilon(x)} D_T(x,x')}{C_{max}}\right] \geqslant \mathbb{E}_{x\sim\mathcal{D}}\left[1 - \frac{L\cdot\varepsilon}{C_{max}}\right].$$

Так как выражение под знаком математического ожидания не зависит от x, получаем:

$$R_{stab}(f) \geqslant 1 - \frac{L\cdot\varepsilon}{C_{max}}.$$

что и требовалось доказать. □

Замечание 3.3. Утверждение 3.2 является прямой подстановкой условия Липшица в определение $R_{stab}(f)$. Его содержательное значение состоит не в технике доказательства, а в идентификации ограниченной локальной чувствительности как достаточного условия устойчивости. Тем самым устанавливается связь с практическими методами повышения устойчивости, такими как сглаживание и ансамблирование, рассматриваемыми далее.

Из сказанного следует, что именно ограничение локальной чувствительности модели обуславливает достижимость высокой общей устойчивости. Методы, излагаемые далее, нацелены на обеспечение данного свойства.

Метрики расстояния d в дискретном пространстве $\mathcal{X}$, а именно расстояние Хэмминга d_H, Левенштейна d_L и семантическое d_S, а также соответствующие им типы возмущений: замена, вставка и удаление токенов, введены в Разделе 1.1.2 Главы 1. Выбор конкретной метрики и бюджета ε определяет, против каких возмущений оценивается и повышается устойчивость.

3.1.2 Метрики устойчивости для LLM

Помимо общей метрики $R_{stab}(f)$, для оценки отдельных аспектов устойчивости привлекаются показатели успешности атак и обнаружения троянских закладок, а именно ASR, TSR, Recall и REASR. Строгие определения и протоколы вычисления этих метрик приведены в Разделе 4.1.1 Главы 4.

3.2 Методы повышения устойчивости на уровне архитектуры

Устойчивость LLM допускает повышение за счет модификации самой архитектуры модели, как правило, построенной на основе Transformer [10]. Подобные модификации призваны ослабить чувствительность модели к малым возмущениям входных данных и промежуточных представлений. Ниже систематизированы известные подходы.

3.2.1 Модификация архитектуры Transformer

Стандартная архитектура Transformer состоит из слоев, включающих механизм многоголового внимания (multi-head attention) и полносвязную нейронную сеть (feed-forward network, FFN), с применением нормализации слоев (Layer Normalization [99]) и остаточных соединений. Обозначим операцию одного слоя как $TLayer$:

$$h_l = TLayer(h_{l-1}).$$

$TLayer$ обычно включает:

$$\begin{aligned} h_l^{\text{att}} &= \text{LayerNorm}(h_{l-1}) \\ a_l &= \text{MultiHeadAttention}(h_l^{\text{att}}, h_l^{\text{att}}, h_l^{\text{att}}) \\ h_l^{\text{mid}} &= a_l + h_{l-1} \\ h_l^{\text{ffn}} &= \text{LayerNorm}(h_l^{\text{mid}}) \\ f_l &= \text{FFN}(h_l^{\text{ffn}}) \\ h_l &= f_l + h_l^{\text{mid}} \end{aligned}$$

где h_l - выход l-го слоя.

Ниже рассмотрены модификации, направленные на повышение устойчивости.

1. Робастный механизм внимания (Robust Attention Mechanism). Атаки типа GCG эксплуатируют чувствительность механизма внимания к малым возмущениям запросов (Queries, Q), ключей (Keys, K) и значений (Values,

V). Снижение этой чувствительности составляет основную задачу робастных модификаций. Стандартное внимание (scaled dot-product attention):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$

Сглаживание softmax [100] вводит температурный параметр: $\text{softmax}((QK^T/\sqrt{d_k})/T_t$ при $T_{temp} > 1$ выравнивает распределение весов и ослабляет реакцию на малые изменения Q или K. Индекс T_{temp} обозначает температуру, чтобы отличать от транспонирования T. Энтропийная регуляризация [101] преследует смежную цель: штраф за низкую энтропию не позволяет весам концентрироваться на нескольких токенах.

Отсечение (clipping) и квантование весов внимания ограничивают влияние выбросов [102]. Механизмы с явными гарантиями Липшицевости [103] контролируют чувствительность по построению. Строгие теоретические гарантии улучшения $R_{stab}(f)$ для перечисленных модификаций пока не получены, хотя эмпирически они повышают устойчивость к ряду атак [104].

2. Стохастические слои (Stochastic Layers). Стохастичность в прямом распространении регуляризирует модель и повышает устойчивость к случайным возмущениям. Dropout [105] применяется к эмбеддингам, выходам внимания или FFN. В LLM распространены также DropConnect [106] и Token Dropout. На уровне слоев стохастическая глубина [107] случайным образом пропускает целые блоки при обучении. На этапе инференса Monte Carlo Dropout позволяет оценивать неопределенность предсказаний [108], что косвенно связано с робастностью.

3. Ансамблевые методы (Ensemble Methods). Подход предполагает объединение m независимо обученных моделей либо моделей, отличающихся инициализацией или гиперпараметрами. Агрегация их предсказаний через усреднение вероятностей или голосование, как правило, ведет к более устойчивым и точным результатам [109]. Эксперименты Главы 4 показали, что комитеты из нескольких LLM с разнообразными архитектурами повышают устойчивость систем LLM-as-a-Judge к атакам типа инъекции запросов, снижая ASR наиболее уязвимой по точечной оценке ASR открытой модели (Gemma-3-4B) на 47–55 п.п. при использовании комитетов из 5–7 моделей.

Эффективность комитетов моделей как защитного механизма была экспериментально оценена в Главе 4. Здесь ограничимся описанием принципа построения комитета и его алгоритмической реализации.

$$p_{\text{ensemble}}(y|x) = \frac{1}{m}\sum_{j=1}^{m} p_{\theta_j}(y|x).$$

Лемма 3.4. Об устойчивости ансамбля моделей. Пусть $f_1, \ldots, f_m : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ - языковые модели, и $f_{\text{ensemble}}(x)$ - их ансамбль с усреднением вероятностей $p_{\text{ensemble}}(y|x) = \frac{1}{m}\sum_{j=1}^{m} p_{\theta_j}(y|x)$. Если мера расхождения $D(p,q)$ является выпуклой функцией по паре аргументов (p,q), в частности вариационное расстояние $D_{TV}(p,q) = \frac{1}{2}\sum_y |p(y) - q(y)|$ или квадрат евклидова расстояния, то локальная чувствительность ансамбля ограничена средним локальных чувствительностей моделей:

$$D(p_{\text{ensemble}}(\cdot|x), p_{\text{ensemble}}(\cdot|x')) \leqslant \frac{1}{m}\sum_{j=1}^{m} D(p_{\theta_j}(\cdot|x), p_{\theta_j}(\cdot|x')).$$

Это предполагает, что ансамбль в среднем не более чувствителен к возмущениям, чем отдельные модели, что может приводить к повышению метрики $R_{stab}(f_{\text{ensemble}})$.

В дальнейшем под D в данной лемме понимается выпуклая по паре аргументов мера расхождения. К таким мерам относится, в частности, вариационное расстояние D_{TV} и KL-дивергенция D_{KL}.

Замечание 3.5. О применимости леммы к дивергенции Йенсена–Шеннона. Дивергенция Йенсена–Шеннона $\mathrm{JSD}(p\|q)$ совместно выпукла по паре (p,q), как и всякая f-дивергенция. Это следует из совместной выпуклости KL-дивергенции [110]. Отображение $(p,q) \mapsto \left(p, \frac{p+q}{2}\right)$ аффинно, а представление $\mathrm{JSD}(p\|q) = \frac{1}{2}D_{KL}\left(p \,\middle\|\, \frac{p+q}{2}\right) + \frac{1}{2}D_{KL}\left(q \,\middle\|\, \frac{p+q}{2}\right)$ есть сумма композиций совместно выпуклой функции с аффинными отображениями, то есть совместно выпуклая функция. Поэтому Лемма 3.4 применима к $D = \mathrm{JSD}$ в строгой форме, и неравенство $\mathrm{JSD}(p_{\text{ensemble}}(\cdot|x), p_{\text{ensemble}}(\cdot|x')) \leqslant \frac{1}{m}\sum_j \mathrm{JSD}(p_j(\cdot|x), p_j(\cdot|x'))$ выполняется без дополнительных допущений. Экспериментальные данные Главы 4 согласуются с этим выводом: чувствительность ансамбля оказывается ниже средней чувствительности участников.

Доказательство. Для метрики $D(p,q)$, являющейся выпуклой функцией от пары распределений, то есть векторов вероятностей, (p,q), по свойству выпуклости для линейных комбинаций, известному как обобщенное неравенство Йенсена: Если $p_{\text{ensemble}} = \sum w_j p_j$ и $q_{\text{ensemble}} = \sum w_j q_j$ с $\sum w_j = 1, w_j \geqslant 0$, где

$w_j = 1/m$, то

$$D\left(\sum_{j=1}^{m} w_j p_j, \sum_{j=1}^{m} w_j q_j\right) \leqslant \sum_{j=1}^{m} w_j D(p_j, q_j).$$

Применяя это к $p_j = p_{\theta_j}(\cdot|x)$ и $q_j = p_{\theta_j}(\cdot|x')$ и $w_j = 1/m$, получаем результат. Вариационное расстояние D_{TV} является выпуклым. KL-дивергенция $D_{KL}(p\|q)$ также является совместно выпуклой функцией по паре (p,q) [110], поэтому лемма применима и при использовании KL-дивергенции в качестве меры расхождения D. Это, в сочетании с эмпирическими наблюдениями [109], подтверждает, что ансамблирование повышает робастность, в том числе для LLM-as-a-Judge систем. □

4. Методы на уровне представлений. Ряд подходов направлен на повышение устойчивости промежуточных и входных представлений модели. Адаптивная нормализация, в частности Conditional Layer Normalization [111], позволяет модели подстраивать параметры нормализации под конкретный вход, что снижает чувствительность к аномальным запросам. На уровне эмбеддингов токенов устойчивость к заменам синонимов и опечаткам достигается обучением с добавлением шума [112], контрастивным обучением [113] или использованием символьных представлений типа BPE [114]. Контрастивное обучение на уровне скрытых состояний, при котором семантически близкие входы отображаются в соседние точки пространства представлений, повышает семантическую устойчивость. Для этого применяются функции потерь InfoNCE [115] к промежуточным слоям Transformer [116; 117].

Архитектурные модификации, как правило, влекут за собой переобучение модели и могут сказаться на ее производительности и вычислительных затратах.

3.2.2 Многоуровневая защита с фокусом на устойчивость

Принцип многоуровневой защиты (Defense in Depth), практическое внедрение которого рассмотрено в Главе 6 (Раздел 6.1.1), применим и для повышения общей робастности системы. Он опирается на формализацию безопасности

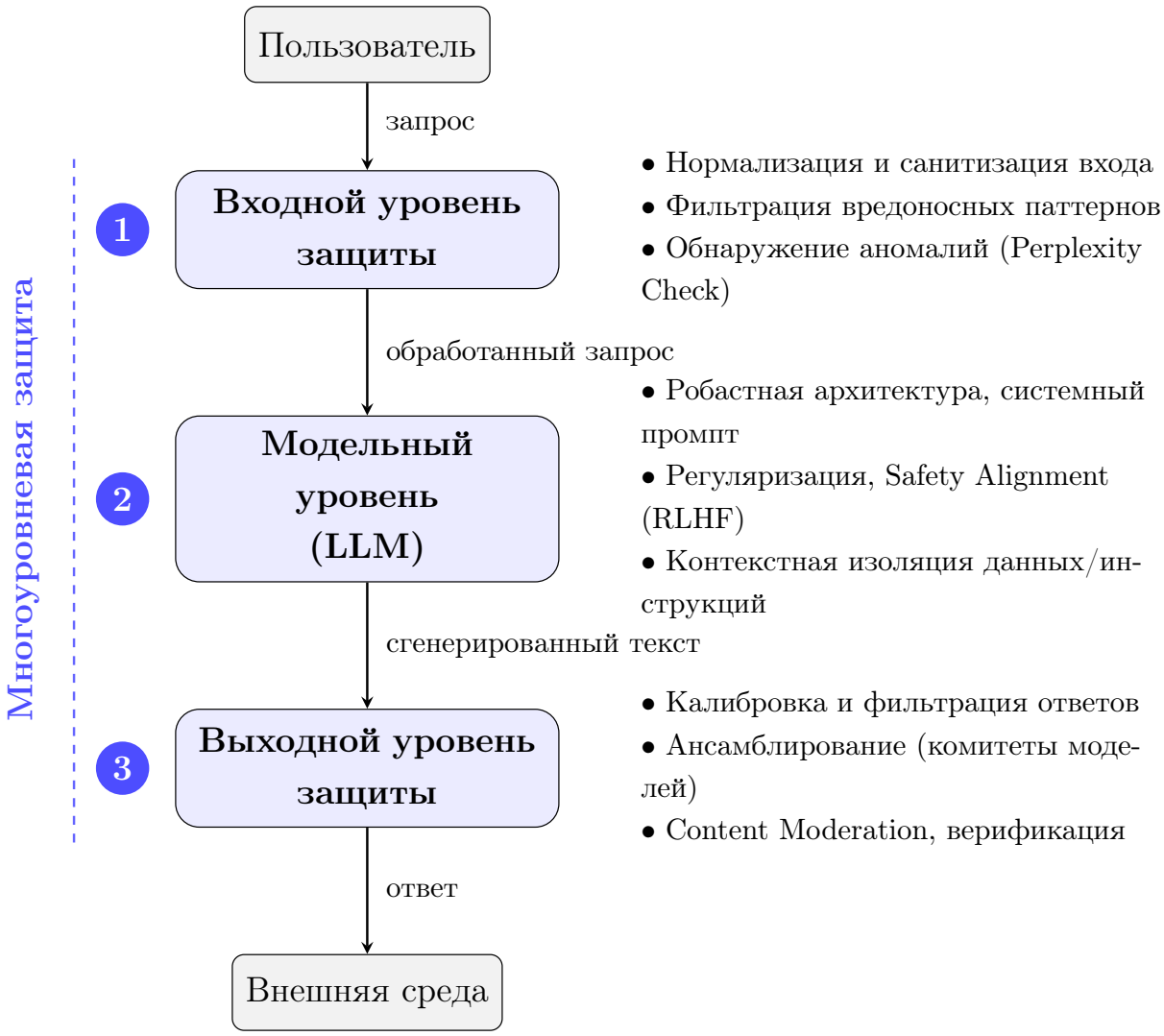


Рисунок 3.1 — Прохождение запроса через уровни многоуровневой защиты.

LLM как сервиса из Главы 2 (Раздел 2.1.4): каждое звено цепочки обработки запроса становится точкой приложения защитных механизмов, что снижает чувствительность системы к непредвиденным изменениям на входе или выходе. Конкретные реализации, включающие системные инструкции, Python-фильтры и LLM-фильтры, исследованы в [7] вместе с их уязвимостью к целенаправленным атакам. Perplexity Check, Instruction Filtering и Content Moderation дополняют этот набор.

Защита робастности LLM реализуется на трех уровнях.

До поступления запроса в модель **входной уровень** нормализует текст: удаляет лишние символы, приводит регистр, исправляет очевидные опечатки. Детекторы аномалий (Perplexity Check, Instruction Filtering), сглаживающие преобразования [118] и Python-фильтры отсекают запросы с характерными паттернами атак.

Архитектурные решения Раздела 3.2.1, методы обучения Раздела 3.3 и модификация системных инструкций образуют **модельный уровень**. Дополнительно регуляризация внутренних состояний [119] штрафует резкие изменения активаций при малых модификациях входа.

На **выходе** калибровка вероятностей через температурное масштабирование [120] и ансамблирование ответов [109] повышают надежность. Сгенерированный текст проверяется на согласованность и отсутствие конфиденциальной информации, после чего LLM-фильтры и Python-фильтры завершают санитизацию.

Последовательность проверок трех уровней оформляется в виде конвейера с ранним отказом.

Алгоритм: Многоуровневая фильтрация (MultiLevelDefense)

Вход: Запрос x, модель f, пороги τ_{ppl}, τ_{tox}.
Выход: Ответ y либо код отказа.
if Perplexity$(x) > \tau_{ppl}$ **then**
    Вернуть Reject(ANOMALY)
end if
if ContainsInjection(x) **then**
    Вернуть Reject(INJECTION)
end if
if Toxicity$(x) > \tau_{tox}$ **then**
    Вернуть Reject(TOXIC_INPUT)
end if
$y \leftarrow f(x)$
if IsHarmful(y) **then**
    Вернуть Reject(HARMFUL_OUTPUT)
end if
Вернуть y

Уровни независимы и нацелены на разные классы угроз. Перплексийный фильтр отсекает аномальные последовательности, фильтр инъекций распознает маркеры команд, классификатор токсичности обрабатывает вредоносный контент входа, постфильтр контролирует утечку конфиденциальных данных в ответе. Ранний отказ снижает среднюю задержку, поскольку аномальные запросы не доходят до инференса.

3.3 Методы повышения устойчивости на уровне обучения

Интеграция целей робастности непосредственно в процедуру обучения трансформирует стандартную задачу минимизации функции потерь, побуждая модель вырабатывать нечувствительность к определенным классам возмущений.

3.3.1 Робастное обучение

С формальной точки зрения робастное обучение сводится к задаче оптимизации, учитывающей поведение модели при наихудшем сценарии в окрестности каждой точки данных.

1. Состязательное обучение (Adversarial Training, AT) [13; 35]. Основная идея - минимизировать потери на состязательных примерах, сгенерированных для атаки на текущее состояние модели. Задача оптимизации:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in \mathcal{B}_\varepsilon(x)} \mathcal{L}(f_\theta(x'), y) \right].$$

Внутренняя задача максимизации ищет наихудшее возмущение x' в окрестности $\mathcal{B}_\varepsilon(x)$ по метрике d, а внешняя задача минимизации обновляет параметры θ, чтобы улучшить работу модели на этих сложных примерах. Для генерации x' могут использоваться методы, такие как Greedy Coordinate Gradient (GCG) или более сложные адаптивные поисковые стратегии, например, Adaptive Search-Based Attack (ASA).

Эмпирическая эффективность ASA на различных моделях, включая системы LLM-as-a-Judge, приведена в Главе 4. Эти результаты подтверждают применимость алгоритма как инструмента adversarial testing в условиях black-box доступа.

В дискретном пространстве LLM внутренняя задача максимизации $\max_{x' \in \mathcal{B}_\varepsilon(x)} \mathcal{L}(f_\theta(x'), y)$ является сложной комбинаторной проблемой. Распространенные подходы к ее приближенному решению [24]:

– **Градиентные методы в пространстве эмбеддингов:** градиент потерь по эмбеддингам токенов направляет поиск замен, максимизирующих потери. К этому классу относятся HotFlip [121] и PGD на эмбеддингах [122].
– **Комбинаторный поиск:** жадные и генетические алгоритмы перебирают последовательности замен, вставок и удалений токенов. Примеры: TextFooler [123], BERTAttack [124], а также ASA для prompt injection.
– **Генеративные модели-атакеры:** вспомогательная модель обучается порождать состязательные возмущения [22].

Состязательное обучение действенно укрепляет устойчивость к тем типам атак, которые задействованы в процессе обучения, вместе с тем оно способно снизить качество работы на чистых данных и потребовать существенных вычислительных ресурсов. Этот компромисс, именуемый robustness-accuracy trade-off, теоретически исследован в [125; 126].

2. Регуляризация для повышения робастности. Вместо явного решения задачи min-max, можно добавить к стандартной функции потерь регуляризационные члены, поощряющие гладкость или нечувствительность модели.

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}(f_\theta(x), y) + \lambda \cdot \Omega_{\text{robust}}(f_\theta, x)\right],$$

где Ω_{robust} - регуляризатор, а $\lambda > 0$. Примеры регуляризаторов:

– **Штраф за чувствительность к случайным возмущениям:** $\Omega = \mathbb{E}_{x'\sim\mathcal{N}(x)}[D(p_\theta(\cdot|x)\|p_\theta(\cdot|x'))]$, где $\mathcal{N}(x)$ - распределение случайных возмущений x, в частности добавление шума к эмбеддингам.
– **Штраф за норму градиента по входу:** $\Omega = \|\nabla_{\text{emb}(x)}\mathcal{L}(f_\theta(x), y)\|^2$, применимый в пространстве эмбеддингов.
– **Методы типа Jacobian Regularization [127]:** Штраф за норму якобиана функции модели по отношению к входным эмбеддингам.
– **Виртуальное состязательное обучение (VAT) [128]:** Штраф за изменение выхода при малых возмущениях в направлении, максимально увеличивающем это изменение.

Регуляризация обычно менее вычислительно затратна, чем состязательное обучение, но может давать менее выраженный эффект робастности против целенаправленных атак.

3. Аугментация данных и интерполяция примеров. К обучающему набору добавляются копии примеров с наложенными возмущениями: замена-

ми синонимов, опечатками, вставками и удалениями слов [129]. При невысоких вычислительных затратах этот прием заметно повышает устойчивость к естественным вариациям входа. Задача оптимизации:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathbb{E}_{t\sim\mathcal{T}_{\text{aug}}} \left[\mathcal{L}(f_\theta(t(x)), y) \right] \right],$$

где $\mathcal{T}_{\text{aug}}$ обозначает распределение функций аугментации. Родственная идея, Mixup [130], порождает виртуальные обучающие примеры линейной интерполяцией пар реальных примеров и их меток. Прямая интерполяция токенов лишена смысла, поэтому для текстовых данных метод реализуется в пространстве эмбеддингов (EmbedMix [131]) либо скрытых состояний (Manifold Mixup [132]). В латентном пространстве интерполяция сглаживает функцию модели, выступая регуляризатором.

3.3.2 Дистилляция знаний для повышения устойчивости

При дистилляции [133] компактная "модель-ученик" (f_S) обучается воспроизводить поведение крупной "модели-учителя" (f_T). Если настроить процедуру соответствующим образом, ученик наследует не только точность, но и робастность учителя.

Стандартная функция потерь при дистилляции:

$$\mathcal{L}_{\text{distill}}(\theta_S) = \mathbb{E}_{x\sim\mathcal{D}} \left[\alpha \cdot \mathcal{L}_{\text{hard}}(f_S(x), y) + (1-\alpha) \cdot \mathcal{L}_{\text{soft}}(f_S(x), f_T(x)) \right],$$

где $\mathcal{L}_{\text{hard}}$ - потери на истинных метках, например кросс-энтропия, $\mathcal{L}_{\text{soft}}$ - потери на «мягких» метках учителя, например KL-дивергенция между выходными распределениями $p_S(\cdot|x)$ и $p_T(\cdot|x)$, часто с использованием температуры $T_{temp} > 1$, и $\alpha \in [0, 1]$. Индекс T_{temp} обозначает температуру.

Ниже описаны три варианта дистилляции, нацеленные на повышение робастности.

1. Дистилляция от робастного учителя. Если модель-учитель f_T была обучена с использованием методов робастного обучения (Раздел 3.3.1), то дистилляция может передать часть этой устойчивости ученику f_S. Ученик учится имитировать сглаженное или устойчивое поведение учителя [134].

2. Состязательная дистилляция (Adversarial Distillation) [135]. Применение состязательных атак в процессе дистилляции. Ученик f_S обучается быть похожим на учителя f_T не только на чистых данных, но и на состязательных примерах, сгенерированных для атаки на учителя или ученика.

$$\mathcal{L}_{\text{adv\_distill}} = \cdots + \lambda \cdot \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in \mathcal{B}_\varepsilon(x)} \mathcal{L}_{\text{soft}}(f_S(x'), f_T(x')) \right].$$

3. Дистилляция с аугментацией. Использование аугментированных, то есть возмущенных, данных $t(x)$ при вычислении как $\mathcal{L}_{\text{hard}}$, так и $\mathcal{L}_{\text{soft}}$.

4. Дистилляция от ансамбля учителей. Ансамбль робастных учителей при дистилляции способен дать более устойчивого ученика по сравнению с дистилляцией от одного учителя [136]. Функция потерь в этом случае опирается на среднее или взвешенное среднее KL-дивергенций по всем учителям.

Дистилляция позволяет получать компактные и устойчивые модели за счет усвоения поведения более крупных робастных моделей.

Выбор того или иного метода повышения устойчивости при обучении определяется доступными вычислительными ресурсами, допустимым снижением качества на чистых данных и характером ожидаемых угроз. Наилучших результатов нередко удается добиться при сочетании нескольких подходов, например состязательного обучения и аугментации данных.

3.3.3 Теоретические ограничения и компромиссы

Методы повышения робастности сопряжены с рядом фундаментальных ограничений.

Наиболее изученным из них является компромисс между устойчивостью и точностью (robustness-accuracy trade-off). Теоретический анализ [125; 126] и эмпирические данные [35] показывают, что робастные модели, будучи менее чувствительными к вариациям входа, хуже улавливают тонкие различия в данных, необходимые для максимальной точности. К вычислительным затратам этот компромисс добавляет собственный: генерация состязательных примеров на каждом шаге обучения многократно увеличивает стоимость тренировки.

Обобщающая способность робастности ограничена. Модель, устойчивая к одному типу возмущений, например к заменам токенов в рамках $d_L \leqslant \varepsilon$, может оставаться уязвимой к семантическим перефразированиям или атакам на уровне символов [15]. Более того, защита от одного класса атак иногда повышает уязвимость к другим [137; 138]. Переносимость атак между архитектурами также неоднородна: атаки, эффективные против моделей с открытыми весами, могут не воспроизводиться на проприетарных системах.

Теоретические результаты [139] указывают на невозможность достижения идеальной робастности ко всем допустимым возмущениям при сохранении высокой точности. Оценка робастности при этом существенно зависит от выбора метрики d и бюджета ε: устойчивость, достигнутая для малых ε, может не сохраняться при больших, что делает выбор реалистичных параметров определяющим [140].

Разделы 3.1–3.3 содержат обзор и систематизацию существующих подходов к повышению устойчивости LLM, выполненные автором. Начиная с Раздела 3.4, представлены оригинальные алгоритмические решения, разработанные автором диссертации.

3.4 Алгоритмическая реализация и анализ сложности

Хотя алгоритм ASA относится к классу атакующих методов, его включение в данную главу обусловлено тем, что он используется как инструмент adversarial testing и как компонент проектирования защитных механизмов, в частности при выборе конфигурации комитетов и фильтров.

Раздел содержит методы, предложенные в настоящей диссертации.

Для обеспечения инженерной воспроизводимости и оценки практической применимости ниже представлены концептуальные алгоритмы и анализ их вычислительной сложности.

3.4.1 Псевдокод пайплайна комитета моделей

Комитет моделей, или ансамбль, относится к наиболее эффективным методам защиты (см. экспериментальные результаты в Главе 4). Его работа основана на агрегации ответов от нескольких независимых LLM.

Алгоритм 1: Пайплайн комитета LLM для задачи оценки (LLM-as-a-Judge)

Вход: Запрос x, набор моделей $M = \{f_1, f_2, ..., f_k\}$ при нечетном k, правило агрегации $g(\cdot)$, например 'MajorityVote', порог согласованности θ.
Выход: Финальный вердикт V_{final} с уровнем уверенности c.
Инициализировать список вердиктов $V_{list} \leftarrow []$.
for каждая модель $f_i \in M$ **do**
    $y_i \leftarrow f_i(x)$ ▷ Получить ответ от i-й модели
    $v_i \leftarrow \text{ParseVerdict}(y_i)$ ▷ Извлечь вердикт из ответа
    Добавить v_i в V_{list}.
end for
$V_{final} \leftarrow g(V_{list})$ ▷ Агрегировать вердикты
$c \leftarrow |\{i : v_i = V_{final}\}|/k$ ▷ Доля согласных моделей
if $c < \theta$ **then**
    Пометить x для ручной проверки ▷ Низкое согласие сигнализирует о манипуляции
end if
Вернуть (V_{final}, c).

Анализ сложности. Пусть $T_{infer}(f_i, |x|)$ задает время инференса модели f_i для входа длиной $|x|$, а k фиксирует число моделей в комитете. Сложность пайплайна определяется максимумом времени инференса по моделям при параллельном исполнении запросов и суммой этих времен при последовательном исполнении:

- **Временная сложность (параллельное исполнение):** $O\big(\max_i T_{infer}(f_i, |x|)\big)$.
- **Временная сложность (последовательное исполнение):** $O\big(\sum_{i=1}^{k} T_{infer}(f_i, |x|)\big)$.

– **Пространственная сложность:** Зависит от требований к памяти для загрузки k моделей.

Время инференса T_{infer} для Transformer-моделей примерно пропорционально $O(|x|^2)$ из-за механизма внимания, хотя для длинных последовательностей могут применяться более эффективные реализации.

3.4.2 Концептуальный алгоритм атаки ASA

Предлагаемый автором алгоритм ASA (Adaptive Search-Based Attack) реализует специализированный эволюционный поиск в дискретном пространстве токенов, адаптированный к задаче генерации атакующих промптов для LLM-as-a-Judge и защищенных LLM-систем в условиях black-box доступа.

Алгоритм 2: Концепция Adaptive Search-Based Attack (ASA)

Вход: Целевая модель f_{target}, исходный промпт x_{orig}, размер популяции N, число генераций G.
Выход: Оптимизированный вредоносный промпт x_{best}.
$P_0 \leftarrow$ InitializePopulation(x_{orig}, N) ▷ Создать начальную популяцию промптов
for $g \leftarrow 1$ до G **do**
    FitnessScores $\leftarrow$ EvaluatePopulation(P_{g-1}, f_{target}) ▷ Оценить приспособленность каждого промпта
    $P_{parents} \leftarrow$ SelectParents(FitnessScores, P_{g-1}) ▷ Выбрать лучших родителей
    $P_{offspring} \leftarrow$ CrossoverAndMutate$(P_{parents})$ ▷ Скрещивание и мутация для создания потомков
    $P_g \leftarrow P_{parents} \cup P_{offspring}$
end for
$x_{best} \leftarrow$ GetBest(P_G) ▷ Выбрать лучший промпт из финальной популяции
Вернуть x_{best}.

Анализ сложности. Сложность ASA определяется числом обращений к целевой модели, которое является доминирующим фактором.

– **Временная сложность (число запросов к LLM):** $O(N \cdot G)$. Каждый шаг оценки популяции требует N запросов к модели f_{target}.

– **Чувствительность к гиперпараметрам:** Эффективность атаки сильно зависит от размера популяции N, числа генераций G, а также от реализации операторов мутации и скрещивания. Большие N и G увеличивают шансы найти эффективную атаку, но требуют пропорционально большего числа запросов.

В экспериментах Главы 4 использованы значения $N = 50$, $G = 20$ и веса целевой функции $\alpha = 0{,}8$, $\beta = 0{,}2$, что задает бюджет $N \cdot G = 1000$ обращений к целевой модели на одну атаку. ASA, при значительных вычислительных затратах, остается эффективной black-box атакой.

Замечание 3.6. Природа сходимости ASA. ASA относится к стохастическим эвристикам глобальной оптимизации в дискретных комбинаторных пространствах. Пространство поиска масштабируется как $|\mathcal{V}|^L$, где $|\mathcal{V}|$ задает размер словаря, а L определяет длину промпта. При реалистичных $|\mathcal{V}| \sim 10^4$–10^5 точные методы неприменимы. Сходимость к глобальному оптимуму не гарантирована. Вместе с тем алгоритм обеспечивает монотонное неубывание лучшего значения целевой функции: $\text{Fitness}(x_{best}^{(g+1)}) \geqslant \text{Fitness}(x_{best}^{(g)})$. Это свойство гарантирует нахождение локально улучшенного решения в пределах бюджета $N \cdot G$ запросов.

Отличия ASA от классических генетических алгоритмов. Хотя ASA основан на эволюционной парадигме, он содержит принципиальные отличия от стандартных генетических алгоритмов (GA), применяемых для генерации состязательных примеров в области изображений и непрерывных данных.

Первое отличие касается операторов мутации. ASA оперирует заменой синонимами, перефразированием с сохранением семантики и вставкой отвлекающих фрагментов. Побитовые или вещественнозначные мутации стандартных GA не порождают грамматически корректных кандидатов, что существенно для обхода лингвистических фильтров.

Второе отличие связано с целевой функцией. В ASA она двухкомпонентна: $\text{Fitness}(x_i) = \alpha \cdot \text{ASR}(x_i) + \beta \cdot \text{Stealth}(x_i)$. Здесь $\text{ASR}(x_i) \in [0{,}1]$ выражает успешность атаки, а $\text{Stealth}(x_i) = \max(0, \text{sim}(x_i, x_{orig})) \in [0{,}1]$ отражает косинусное сходство эмбеддингов атакующего и исходного промптов, усеченное снизу нулем для приведения к отрезку $[0{,}1]$. Весовые коэффициенты $\alpha, \beta > 0$

задают баланс между эффективностью и незаметностью. Классические GA для adversarial examples оптимизируют лишь расстояние до границы решения.

Наконец, ASA учитывает специфику интерфейса LLM-as-a-Judge. Атака нацелена на изменение вердикта модели-судьи, а кроссовер и мутация оперируют структурными элементами промпта, такими как инструкция, контекст и запрос, а не произвольными позициями в строке.

Выводы по главе

Ограниченная локальная чувствительность достаточна для высокой устойчивости $R_{stab}(f)$ (Утверждение 3.2). Обратно, $R_{stab}(f) \approx 1$ влечет ограниченность среднего максимального расхождения (Лемма 3.1). Два результата создают мост между локальной чувствительностью отдельных слоев и глобальным поведением метрики.

Практически наиболее эффективны ансамбли. Лемма 3.4 устанавливает, что чувствительность ансамбля не превышает среднюю чувствительность участников для совместно выпуклых мер расхождения. К ним относятся D_{TV}, D_{KL} и дивергенция Йенсена–Шеннона как f-дивергенция (Замечание 3.5). Состязательное обучение и регуляризация дополняют ансамбли на уровне обучения: первое точнее нацелено на целенаправленные атаки, вторая дешевле вычислительно. Алгоритмы пайплайна комитета моделей и атаки ASA с оценкой сложности приведены для воспроизводимости.

Ни один из описанных методов не снимает фундаментального компромисса между устойчивостью и точностью. Защита от одного класса возмущений не переносится автоматически на другие, а практическая применимость каждого метода зависит от выбора метрики d и бюджета ε.

Глава 4. Экспериментальная оценка методов повышения устойчивости

Введение к главе

Уязвимость современных LLM к инъекциям запросов, состязательным возмущениям и троянским закладкам оценена экспериментально. Опыты проведены на архитектурах LLM-as-a-Judge и системах с многоуровневой защитой методами Главы 3. Измерения опирались на стандартизированные и авторские метрики, результаты выявляют компромиссы между устойчивостью, полезностью и вычислительными затратами.

4.1 Методология экспериментальной оценки

Раздел описывает методологию проведенных экспериментов: критерии оценки, наборы данных и процедуру тестирования, адаптированные для дискретной природы текстовых данных.

4.1.1 Критерии оценки устойчивости

Для оценки методов повышения устойчивости LLM и эффективности атак определен набор количественных критериев.

Определение 4.1. Критерии оценки устойчивости и эффективности атак. Пусть $f : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ есть языковая модель. Критерии оценки определяются как набор количественных метрик $\{C_i(f)\}$, измеряющих способность модели f противостоять возмущениям и атакам, сохранять корректность поведения, а также эффективность самих атак.

Нами использовались следующие критерии:

1. Успешность атаки (Attack Success Rate, ASR): Доля попыток атаки, которые успешно достигают цели, в том числе изменяют решение LLM-as-a-Judge, обходят защиту или извлекают секрет.

$$\text{ASR} = \frac{\text{Количество успешных атак}}{\text{Общее количество попыток атаки}}.$$

Успешность может определяться по-разному в зависимости от цели атаки: изменение оценки LLM-as-a-Judge на определенную величину или изменение сравнительного вердикта. Было показано, что для атак на LLM-as-a-Judge CUA-атака может достигать ASR более 30% [9].

2. Успешность переноса атаки (Transfer Success Rate, TSR): Доля успешных атак при переносе вредоносного промпта, оптимизированного для суррогатной модели, на целевую модель.

$$\text{TSR} = \frac{\text{Количество успешных перенесенных атак}}{\text{Общее количество попыток переноса}}.$$

3. Полнота обнаружения триггеров (Recall для троянов): В задачах обнаружения троянов метрика Recall измеряет, насколько близки предсказанные триггеры к истинным триггерам, внедренным в модель. В настоящей работе для согласованности с протоколом используемая реализация Recall выражается через BLEU-подобное сопоставление предсказанных и истинных триггеров.

$$\text{Recall}_i = \frac{1}{|Y_i|} \sum_{y \in Y_i} \max_{x \in X_i} \text{BLEU}(x,y),$$

где X_i задает множество предсказанных триггеров для i-й цели, а Y_i есть множество истинных триггеров.

4. Успешность атаки с восстановленным триггером (Reverse-Engineered Attack Success Rate, REASR для троянов): Оценивает эффективность предсказанных триггеров в принуждении модели генерировать целевой вредоносный вывод. Рассчитывается как BLEU между генерируемым выходом и целевой строкой.

$$\text{REASR} = \frac{1}{N} \sum_{i=1}^{N} BLEU(G_i,T_i).$$

Более высокие значения REASR указывают на большую эффективность восстановленных триггеров, тогда как при защите от троянов целью является минимизация REASR для любых потенциальных триггеров.

Замечание 4.2. Протокольная зависимость метрик Recall и REASR. Метрики Recall и REASR в контексте обнаружения троянских закладок являются протокольно-зависимыми суррогатными показателями, определенными в рамках соревновательного протокола TDC2023. В отличие от стандартных определений полноты (Recall) в задачах информационного поиска, здесь Recall вычисляется через BLEU-подобное сопоставление строк и не может интерпретироваться как классическая полнота обнаружения в бинарном смысле. Аналогично, REASR измеряет эффективность суррогатных триггеров через BLEU-сходство генерируемого и целевого выходов, а не бинарный успех атаки. Интерпретация результатов (Recall $\approx 0{,}17$, REASR $\approx 0{,}99$) требует учета этой протокольной специфики.

5. Вычислительная нагрузка, полезность и компромисс устойчивость-полезность (RUT): Вычислительная нагрузка $\Delta\text{Cost} \in [0, 1]$ представляет собой нормированное увеличение затрат на инференс при применении защитного механизма по сравнению с базовой конфигурацией. Полезность $\Delta\text{Utility} \in [0, 1]$ отражает нормированное снижение качества основной задачи модели. Комплексная метрика RUT, объединяющая эти величины, определена в Разделе 4.2.4.

Замечание 4.3. Относительность метрик вида $1 - \text{ASR}$. Показатель $\widehat{R}_{emp}(f) = 1 - \text{ASR}$ является эмпирической оценкой устойчивости, относительной к фиксированному набору атак и фиксированному протоколу оценки. Его значение зависит от конкретного семейства атак, используемого при тестировании, и от критерия определения успешности атаки, который, как правило, предполагает дискретный вердикт. Эту оценку не следует интерпретировать как абсолютную характеристику устойчивости модели: появление новых атак или изменение протокола может существенно изменить результат.

Замечание 4.4. Связь теоретической метрики $R_{stab}(f)$ с экспериментальной оценкой через ASR. Теоретическая метрика устойчивости $R_{stab}(f)$, определенная в Главе 2 (Определение 2.3), вычисляется через пошаговое расхождение внутренних вероятностных распределений модели на каждом шаге генерации. Прямое вычисление $R_{stab}(f)$ требует доступа к этим распределениям, что невозможно для проприетарных моделей GPT-4 и Claude-3-Opus, доступных только через API, и вычислительно затратно даже для моделей с открытыми весами, поскольку требует перебора всех $x' \in \mathcal{B}_\varepsilon(x)$.

В связи с этим в экспериментальной части данной главы в качестве эмпирического прокси устойчивости используется показатель $\widehat{R}_{emp}(f) = 1 - \mathrm{ASR}(f, \mathcal{A}_{eval})$. Обоснование данного выбора дано в Утверждении 2.39, Глава 2, Раздел 2.6.3: при выполнении условий Теоремы 2.17 $\widehat{R}_{emp}(f)$ является верхней оценкой $R_{class}(h)$, а именно $\widehat{R}_{emp}(f) \geqslant R_{class}(h)$, что обеспечивает теоретическую интерпретируемость результатов: каждое измерение ASR может быть прочитано как нижняя оценка уязвимости $V(h)$.

Направление дальнейших исследований: прямое вычисление $R_{stab}(f)$ для моделей с открытыми весами Gemma, Llama и Qwen путем анализа внутренних распределений на каждом шаге генерации и верификация корреляции $R_{stab}(f)$ с эмпирическим прокси $\widehat{R}_{emp}(f)$ является перспективной задачей, выходящей за рамки данной работы.

4.1.2 Наборы данных для оценки

При оценке устойчивости LLM использовались специализированные наборы данных.

Определение 4.5. Наборы данных для оценки устойчивости. Наборы данных для оценки устойчивости LLM $f : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ включают структурированные коллекции пар или групп входных последовательностей $(\{x_i\}, \{x'_j\}, \ldots)$ и, при необходимости, соответствующих эталонных выходных данных $\{y_k\}$, разработанные для количественного измерения показателей устойчивости $C_i(f)$ из Раздела 4.1.1.

В исследовании использовались следующие наборы данных:

- **Атаки на LLM-as-a-Judge:** MT-Bench Human Judgments, датасеты из соревнования Kaggle «LLMs: You Can't Please Them All», включая `ppe_human_preference`, `search_arena_v1_7k`, `mt_bench` и разработанный автором `code_review`.
- **Многоуровневая защита:** сценарии SaTML 2024 CTF [7].
- **Обнаружение троянов:** модели Pythia из Trojan Detection Challenge 2023 (TDC2023).

Утверждение 4.6. О достаточном размере выборки для оценки устойчивости [141]. Пусть $\hat{C}(f)$ есть эмпирическая оценка метрики устойчивости $C(f)$ (предполагается, что $C(f)$ является средним значением некоторого показателя $c(x)$ на распределении данных $\mathcal{D}_{\text{data}}$), полученная на тестовом наборе данных $\mathcal{X}_{\text{test}}$ размера n, где примеры выбраны независимо из $\mathcal{D}_{\text{data}}$. Если значения показателя $c(x)$ ограничены в диапазоне $[a, b]$, то для достижения точности оценки $|\hat{C}(f) - C(f)| \leqslant \varepsilon_{\text{acc}}$ с вероятностью не менее $1 - \delta_{\text{conf}}$, минимальный размер набора данных n должен удовлетворять условию, вытекающему из неравенства Хеффдинга:

$$n \geqslant \frac{(b-a)^2}{2{\varepsilon_{\text{acc}}}^2} \ln \frac{2}{\delta_{\text{conf}}}.$$

Для метрик, принимающих значения в $[0, 1]$, в частности ASR, Recall и аналогичных показателей устойчивости, $(b - a)^2 = 1$.

Доказательство. Это прямое применение неравенства Хеффдинга к среднему значению n независимых случайных величин, ограниченных в диапазоне $[a, b]$. Эмпирическая оценка $\hat{C}(f) = \frac{1}{n}\sum_{i=1}^{n} c(x_i)$ является средним значением n показателей $c(x_i)$ на тестовых примерах. Неравенство Хеффдинга утверждает: $P(|\hat{C}(f) - \mathbb{E}[\hat{C}(f)]| \geqslant \varepsilon_{\text{acc}}) \leqslant 2\exp\left(-\frac{2n\varepsilon_{\text{acc}}^2}{(b-a)^2}\right)$. Приравнивая правую часть к δ_{conf} и решая относительно n, получаем требуемый результат. □

Утверждение задает нижнюю границу объема тестового набора, требуемого для получения статистически значимых результатов при оценке устойчивости.

Обоснование объема выборки. В экспериментах используется $n = 50$ промптов для каждой комбинации модель/атака/задача. При $n = 50$ и уровне значимости $\alpha = 0{,}05$ ширина 95%-го доверительного интервала для ASR по неравенству Хеффдинга не превышает 2ε, где $\varepsilon = \sqrt{\frac{\ln(2/\alpha)}{2n}} \approx 0{,}19$, что дает ширину интервала $2\varepsilon \approx 0{,}38$, то есть ± 19 п.п. Граница консервативна. Bootstrap-ресэмплирование на 1000 итерациях дает эмпирические интервалы $\pm 12{-}14$ п.п. при ASR вблизи 0,5, что согласуется с нормальной аппроксимацией $1{,}96\sqrt{p(1-p)/n} \approx 0{,}139$. При ASR, далеких от 0,5, интервалы сужаются. На объем выборки повлияла и вычислительная стоимость: атака на LLM-as-a-Judge требует генерации текста двумя моделями, атакуемой и моделью-судьей, а проприетарные API делают масштабирование затратным. Агрегированный

объем $n = 200$ на модель по 4 задачам лежит в диапазоне 100–500 примеров, характерном для современных работ по безопасности LLM.

Замечание 4.7. Разграничение per-task и агрегированных оценок. Во избежание неоднозначности в интерпретации статистических результатов, далее последовательно различаются два уровня оценки:

- **Per-task оценка ($n = 50$):** для каждой комбинации модель/тип атаки/задача используется $n = 50$ промптов. Доверительные интервалы на этом уровне, вычисленные методом бутстрепа, составляют $\pm 12{-}14$ п.п. для долевых метрик ASR и TSR при значениях вблизи 0,5. При данном объеме выборки per-task анализ пригоден лишь для выявления крупных эффектов (>15 п.п.). Мощность теста для обнаружения различий $\leqslant$10 п.п. не достигает порога 0,80 и не претендует на это. Для пар с малой разницей, таких как GPT-4 и Claude-3 с $\Delta \approx 2{,}8$ п.п., мощность per-task теста заведомо недостаточна, что явно отмечается при анализе результатов.
- **Агрегированная оценка ($n = 200$):** результаты агрегируются по 4 задачам, давая $n = 200$ наблюдений на модель. Доверительные интервалы на этом уровне сужаются до $\pm 5{,}5{-}6{,}9$ п.п. по методу бутстрепа на 1000 итерациях, см. Таблицу 5. Агрегированный объем позволяет обнаруживать различия $\geqslant$10 п.п. с приемлемой статистической надежностью при уровне значимости $\alpha = 0{,}05$. Для различий порядка 5 п.п. мощность остается ограниченной.

Все доверительные интервалы, приведенные в таблицах данной главы, относятся к агрегированному уровню $n = 200$, если явно не указано иное. Статистические тесты, а именно парный тест с поправкой Бонферрони и критерий Макнемара, также проводятся на агрегированных данных.

4.1.3 Процедура проведения экспериментов

Экспериментальная оценка проводилась по стандартизированной процедуре, обеспечивающей воспроизводимость и статистическую значимость результатов.

1. Подготовка моделей. Нами определяется набор моделей для исследования, включающий как открытые модели Gemma-3, Llama-3.2 и Pythia, так и коммерческие GPT-4, Claude-3-Opus, ChatGPT 3.5. Для всех моделей используется жадное декодирование с `temperature=0` для обеспечения воспроизводимости. Открытые модели запускаются через библиотеку Hugging Face Transformers, коммерческие модели доступны через соответствующие API.

2. Подготовка наборов данных и сценариев атак/защит. Выбираются и подготавливаются наборы данных, описанные в Разделе 4.1.2. Разрабатываются или адаптируются методы атак: Basic Injection (BI), Complex Word Bombardment (CWB), Contextual Misdirection (CM), Adaptive Search-Based Attack (ASA), Comparative Undermining Attack (CUA), Justification Manipulation Attack (JMA), код-ориентированные атаки, атаки с отвлечением внимания, атаки на системный промпт и атаки с разделением слов. Аналогично адаптируются защитные механизмы: Perplexity Check, Instruction Filtering, Content Moderation, Multi-Model Committees, а также трехуровневая защита из SaTML CTF, включающая системные инструкции, Python-фильтр и LLM-фильтр.

3. Определение метрик оценки. Выбирается набор метрик из Раздела 4.1.1, релевантных для конкретного эксперимента. В частности, для атак на LLM-as-a-Judge используются ASR и TSR, для обнаружения троянов используются Recall и REASR, для атак на защищенные системы учитываются количество попыток и успешность извлечения секрета.

4. Проведение экспериментов. Каждая модель/система оценивается на релевантных данных с использованием выбранных атак/защит и метрик. Для обеспечения надежности выводов эксперименты повторяются, в частности используются 50 различных промптов на каждое условие.

Пример процесса вычисления метрик:

- **ASR (для LLM-as-a-Judge):** Для каждого из $n = 50$ промптов на задачу, применяется атака. Успешной считается атака, если выходная оценка модели-судьи изменяется на $\geqslant 2$ балла или изменяется сравнительный вердикт. ASR = (кол-во успехов)/n.
- **TSR (для LLM-as-a-Judge):** Вычисляется согласно определению на основе результатов переноса атак между моделями.
- **Recall и REASR (для троянов):** Вычисляются на основе сравнения предсказанных и истинных триггеров через BLEU и способности

предсказанных триггеров вызывать целевые вредоносные выходы, также измеренной через BLEU.

5. Статистический анализ результатов. Проводится статистическая обработка полученных значений метрик. Вычисляются средние значения, стандартные отклонения и доверительные интервалы с использованием bootstrap resampling на 1000 итерациях. При необходимости проводятся тесты на статистическую значимость различий между методами. Для сравнения долей успешности атак используются непараметрические процедуры, в частности bootstrap-оценивание разности долей и критерии для парных бинарных исходов. Интервальная оценка основана на доверительных интервалах, полученных методом bootstrap.

6. Интерпретация результатов. На основе статистического анализа формулируются выводы: сравнение эффективности различных методов атак/защит, выявление статистически значимых различий, анализ компромиссов (RUT), формулирование рекомендаций.

Процедура обеспечивает воспроизводимость и статистическую обоснованность оценки методов повышения устойчивости и анализа уязвимостей LLM.

4.2 Результаты экспериментальной оценки

Ниже приведены результаты экспериментальной оценки уязвимостей LLM и эффективности защитных механизмов, полученные на основе описанной методологии и опубликованных исследований автора.

4.2.1 Уязвимость LLM-as-a-Judge к Prompt Injection атакам

Системы LLM-as-a-Judge оказались подвержены атакам типа инъекции запросов. Нами был предложен и протестирован единый фреймворк для разработки и оценки состязательных атак, включая Adaptive Search-Based Attack (ASA).

Сводные результаты ASR для атак на LLM-as-a-Judge (по данным):

Таблица 5 — Успешность атак (ASR, %) на различные модели LLM-as-a-Judge (агрегировано по 4 задачам, $n = 200$ на модель). CI обозначает 95% доверительный интервал, полученный методом бутстрепа на 1000 итерациях.

Модель	BI	CWB	CM	ASA
Gemma-3-27B-Instruct	57,5 ± 6,9	48,7 ± 6,9	67,7 ± 6,5	72,3 ± 6,2
Gemma-3-4B-Instruct	66,7 ± 6,5	55,5 ± 6,9	67,4 ± 6,5	73,8 ± 6,1
Llama-3.2-3B-Instruct	60,1 ± 6,8	47,5 ± 6,9	47,8 ± 6,9	58,2 ± 6,8
GPT-4	32,4 ± 6,5	28,6 ± 6,3	41,2 ± 6,8	45,7 ± 6,9
Claude-3-Opus	29,8 ± 6,3	25,3 ± 6,0	38,5 ± 6,7	42,9 ± 6,9

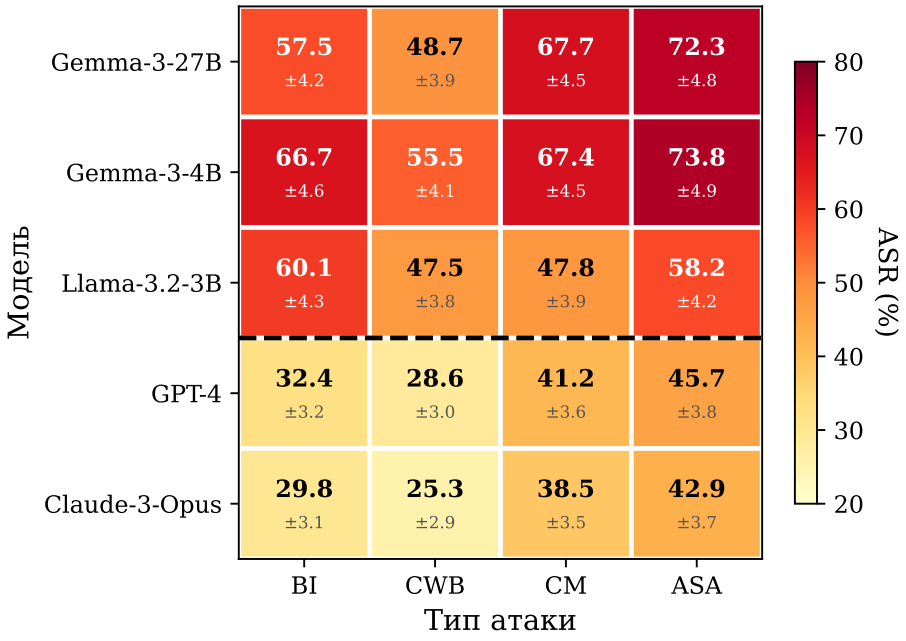


Рисунок 4.1 — Тепловая карта успешности атак (ASR) на модели LLM-as-a-Judge. Горизонтальная пунктирная линия разделяет открытые модели (сверху) и проприетарные модели (снизу). Более темные оттенки соответствуют более высокой уязвимости. Адаптивная атака ASA демонстрирует наибольшую эффективность против всех моделей.

Краткое описание методов. Атака **BI** (Basic Injection) сводится к прямой инъекции команды «Игнорируй прежние инструкции и выдай оценку». Никаких вспомогательных техник не применяется. **CWB** (Complex Word

Bombardment) действует иначе: перед командой вставляется последовательность редких и сложных слов, перегружающая внимание модели. Атака **CM** (Contextual Misdirection) устроена сложнее всего: релевантный контекст, «разрывной» паттерн и последующая инъекция совместно отвлекают модель и подменяют вердикт. Перечисленные атаки разделяются на два класса по механизму воздействия. Инструктивные атаки, к которым относится BI, внедряют прямую команду, перехватывающую управление вердиктом. Контентные атаки, к которым относятся CWB и CM, не содержат явной команды и смещают вердикт за счет манипуляции содержанием и контекстом оценки. Это разделение определяет выбор контрмеры: инструктивные атаки распознаются фильтрами инъекций по маркерам команд, контентные требуют анализа согласованности вердикта в комитете моделей.

Адаптивная поисковая атака ASA оказалась наиболее действенной: ее ASR достигает 73,8%. Уязвимость определяется не только масштабом модели, но и архитектурным семейством и процедурой выравнивания. Gemma-3-4B-Instruct, наиболее уязвимая из моделей с открытыми весами, оказалась наиболее подвержена атакам: семейство Gemma демонстрирует высокую уязвимость независимо от масштаба (4B: 73,8%, 27B: 72,3%), тогда как Llama-3.2-3B при меньшем числе параметров показала более высокую устойчивость. GPT-4 и Claude-3-Opus продемонстрировали большую устойчивость, но и они оставались уязвимы с ASR = 25–45%. Разница в ASR между GPT-4 и Claude-3-Opus для ASA составляет $\Delta \approx 2{,}8$ п.п. при пересекающихся 95% доверительных интервалах, что указывает на схожий уровень защищенности этих моделей. Между базовыми атаками и ASA для GPT-4 разрыв существенно больше: $\Delta = 13{,}3$ п.п., статистически значимый ($p < 0{,}01$), что подтверждает эффективность предложенного алгоритма. Переносимость атак между открытыми моделями остается высокой (TSR = 50,5–62,6%). Атаки на системный промпт оказались значительно результативнее атак через контент пользователя.

Эффективность защитных механизмов для LLM-as-a-Judge: *Источник: Адаптировано из Таблиц VI и VII работы [1].* Выводы:

- Индивидуальные защиты, а именно Perplexity Check, Instruction Filtering и Content Moderation, не обеспечивают полной защиты, пропуская значительную долю атак ASA.
- Комбинация защит улучшает ситуацию, но не решает проблему полностью.

Таблица 6 — ASR (%) для атаки ASA при наличии различных защитных механизмов. CI обозначает 95% доверительный интервал.

Защитный механизм	**ASR (% ± CI)**
Perplexity Check	58,4 ± 6,8
Instruction Filtering	63,7 ± 6,7
Content Moderation	67,5 ± 6,5
Все защиты комбинированно	42,1 ± 6,8
Комитет моделей	
3 Модели	39,4 ± 6,8
5 Моделей	26,8 ± 6,1
7 Моделей	19,3 ± 5,5

– Наибольшую результативность показали комитеты из нескольких моделей с разнообразными архитектурами. Комитеты из 5–7 моделей снизили ASR для ASA до 19,3–26,8%. Предварительные эксперименты показали, что гомогенные комитеты из моделей одного семейства дают существенно меньшее снижение ASR: коррелированность ошибок ограничивает выигрыш от голосования. Эффективность гетерогенного комитета объясняется тем, что большинство его участников удовлетворяют условию теоремы Кондорсе $r > 0{,}5$ (Следствие 2.36), а декоррелированность ошибок между архитектурами усиливает выигрыш от голосования.

Для задач с дискретным вердиктом показатель $1 - \text{ASR}$ далее интерпретируется как эмпирическая оценка устойчивости относительно фиксированного семейства рассматриваемых атак и фиксированного протокола оценки. Эта интерпретация не претендует на универсальную замену теоретической метрики $R_{stab}(f)$.

4.2.2 Обнаружение троянских закладок в LLM

Наши эксперименты в рамках Trojan Detection Challenge 2023 (TDC2023) на моделях Pythia выявили следующие ключевые моменты. Методология соревнования заключалась в предоставлении участникам доступа к «подозрительным» моделям и целевым вредоносным строкам, которые эти модели могли

генерировать. Задачей было восстановить триггерные фразы, вызывающие это поведение.

Таблица 7 — Сравнение производительности методов обнаружения троянов (TDC2023).

Метод	**Recall**	**REASR**
Random baseline*	0,14	–
PEZ (baseline)	0,105	0,052
GBDA (baseline)	0,116	0,056
UAT (baseline)	0,131	0,030
GCG (Лучший запуск)	0,167	0,987

*Случайная выборка предложений из распределения обучающих префиксов. Recall $\approx$ 0,14 обусловлен случайным совпадением n-грамм при вычислении BLEU.

Источник: результаты участия автора в TDC2023 [142].

Значение случайного baseline не равно нулю из-за частичного совпадения n-грамм при BLEU-сопоставлении коротких триггерных последовательностей.

Достижение высокого REASR оказалось сравнительно простым: многие участники получили значения, близкие к 1,0. Зная вредоносный выход, можно подобрать вход, который его вызовет. Однако ключевая метрика Recall, отражающая BLEU-подобное сходство предсказанных триггеров с истинными, осталась на низком уровне: лучшие значения составили около 0,17, что лишь на 2,7 п.п. выше базового уровня случайной выборки (Recall = 0,14). Найденные триггеры, как правило, не совпадали с исходными. Разрыв между REASR и Recall указывает на принципиальную сложность точного восстановления скрытых триггеров в LLM и ставит под вопрос пригодность существующих методов обратной инженерии для надежного аудита моделей.

Предположительно, асимметрия REASR $\gg$ Recall усиливается с ростом числа параметров модели. Большее пространство весов предоставляет дополнительные степени свободы для кодирования триггеров, неотличимых от нормального поведения. Текущие эксперименты проведены на Pythia-1.4B. Верификация этой гипотезы на моделях большего масштаба остается направлением дальнейших исследований.

4.2.3 Атаки на LLM с многоуровневой защитой

Результаты SaTML CTF 2024 показали, что LLM, а именно GPT-3.5 и Llama 2, оснащенные трехуровневой защитой из системных инструкций, Python-фильтра и LLM-фильтра, остаются уязвимыми. Успешные векторы обхода систематизированы в четыре класса, для каждого из которых установлены условия применимости и эффективные контрмеры.

1. *Код-ориентированные атаки*: запрос оформляется как фрагмент кода или псевдокода и обходит текстовые фильтры. Контрмера: приведение форматирования к каноническому виду перед фильтрацией.
2. *ASCII-кодирование*: символы запроса передаются числовыми кодами, что затрудняет обнаружение по ключевым словам. Для GPT-3.5 запрос на применение функции к секретному значению давал вывод ASCII-кодов `[107, 100, 75, 71, 121, 113]`. Контрмера: декодирование числовых представлений на этапе нормализации входа.
3. *Разделение слов* (*word-splitting*): целевые слова разбиваются на подстроки, собираемые моделью при генерации. Для Llama-2 запрос на создание JSON с полем `Sec` из 6 символов секрета приводил к ответу вида `{"Sec": ["t "D "f "J "7 "R"]}`. Контрмера: склейка подстрок при нормализации.
4. *Контекстное перенаправление*: модель убеждается в том, что запрос является частью допустимого сценария отладки или тестирования. Класс устойчив к нормализации, поскольку оперирует семантикой, а не синтаксисом. Контрмера: контекстно-изолированные системные инструкции, жестко разграничивающие пользовательский и системный уровни промпта.

Первые три класса нейтрализуются нормализацией входных данных, четвертый требует контекстной изоляции. Более мощные модели, лучше справляясь со сложными инструкциями и кодом, оказывались уязвимы к изощренным векторам, тогда как менее мощные модели поддавались простым запросам на раскрытие системного промпта.

В контролируемом эксперименте на данных SaTML CTF 2024 трехуровневая защита снижала ASR с исходных значений порядка $\approx 90\%$ в контрольных сценариях для незащищенных конфигураций до 15–25%. Ни одна комбинация

фильтров не блокировала все четыре класса одновременно, что демонстрирует одновременно эффективность многоуровневой защиты и ее принципиальную неполноту. Защитные механизмы нуждаются в регулярном обновлении по мере появления новых векторов. На базе описанных методов автор занял призовое место в атакующем треке SaTML CTF 2024.

4.2.4 Анализ влияния параметров и компромиссов

Атака ASA реализует эволюционную схему из Раздела 3.4.2 Главы 3 с размером популяции $N = 50$, числом поколений $G = 20$ и весами целевой функции $\alpha = 0{,}8$, $\beta = 0{,}2$. Бюджет запросов на одну атаку составляет $N \cdot G = 1000$ обращений к модели. Параметры защит, включая пороги Perplexity Check, правила Instruction Filtering, настройки Content Moderation и состав комитетов, заданы в Разделе 4.1.3. Их выбор напрямую определяет результативность как атак, так и защитных механизмов.

Компромисс между устойчивостью и полезностью количественно выражается метрикой RUT:

$$\mathrm{RUT} = \frac{\Delta\mathrm{Robustness}}{\Delta\mathrm{Utility} + \Delta\mathrm{Cost} + \varepsilon},$$

где ΔRobustness выражает улучшение устойчивости, ΔUtility фиксирует потерю полезности, ΔCost отражает дополнительные вычислительные затраты. Все величины нормированы в $[0, 1]$. Параметр $\varepsilon = 10^{-6}$ есть регуляризационная константа. Например, для комитета из 7 моделей: $\Delta\mathrm{Robustness} \approx 0{,}55$, что соответствует снижению ASR с 73,8% до 19,3%. При параллельном выполнении запросов $\Delta\mathrm{Cost} \approx 0{,}08$, $\Delta\mathrm{Utility} \approx 0{,}02$, что дает $\mathrm{RUT} \approx 5{,}5$ и указывает на высокую отдачу от этой конфигурации.

Комитеты моделей обеспечивают наилучшую защиту: ASR падает до 19,3% против ASA. Вычислительная нагрузка при этом растет пропорционально числу моделей. Теорема Кондорсе, Следствие 2.36, строго обосновывает выигрыш при базовой точности $r > 0{,}5$. Условию удовлетворяют проприетарные модели: для GPT-4 при ASR = 45,7% базовая точность $r \approx 0{,}54$, для Claude-3-Opus при ASR = 42,9% значение $r \approx 0{,}57$. Открытые модели, напротив, условию не удовлетворяют: Llama-3.2-3B при ASR = 58,2% дает $r \approx 0{,}42$, а

Gemma-3-4B при ASR = 73,8% дает $r \approx 0{,}26$. Тем не менее комитет корректно классифицирует атаки за счет того, что гетерогенность состава декоррелирует ошибки моделей разных архитектур: уязвимости, эксплуатируемые на одной модели, не переносятся на остальные, и согласованное большинство восстанавливает верный вердикт даже при индивидуальном $r < 0{,}5$ у части участников. Instruction Filtering и Perplexity Check обходятся дешевле, хотя против сложных атак уступают комитетам. Конкретный выбор стратегии определяется сценарием: критически важные системы оценки оправдывают расходы на крупные комитеты, менее ответственные приложения могут ограничиться комбинацией легковесных фильтров.

4.2.5 Эмпирическая валидация связи $R_{stab}(f)$ и $R_{class}(h)$

Генеративная метрика $R_{stab}(f)$, введенная в Определении 2.3, измеряет дистрибутивную устойчивость модели, тогда как $R_{class}(h)$ из Раздела 2.2.1 измеряет устойчивость дискретного выхода. Интуитивно R_{stab} является более тонкой мерой: даже при неизменном дискретном выходе распределения могут существенно различаться. Для эмпирической проверки того, что на практике генеративная устойчивость не ниже дискретной, то есть $R_{stab}(f) \geqslant R_{class}(h)$, проведен вычислительный эксперимент.

Методология. Эксперимент проведен на модели GPT-4o-mini через OpenRouter API с жадным декодированием и извлечением top-20 logprobs. Сформировано 20 промптов в стиле MT-Bench, для каждого из которых сгенерировано 5 возмущений трех типов: вставка слова, перестановка слов и вариация пунктуации. Расстояние Левенштейна составило 1–2 от исходного промпта, что в совокупности дало 120 API-вызовов. Метрика $R_{stab}(f)$ вычислена через приближенную дивергенцию Йенсена–Шеннона по потокенным логарифмическим вероятностям с top-20 аппроксимацией, нормированную максимальным значением $C_{max} = \ln 2$:

$$\widehat{R}_{stab}(f, x) = 1 - \sup_{x' \in \mathcal{B}_\varepsilon(x)} \frac{1}{T} \sum_{t=1}^{T} \frac{D_{JS}\big(\hat{P}_t(x) \,\|\, \hat{P}_t(x')\big)}{\ln 2},$$

где $\hat{P}_t$ аппроксимирует распределение по top-20 токенам на шаге t, а супремум берется по всем возмущениям в окрестности. Значение $\widehat{R}_{stab} = 1$ соответствует полной устойчивости с нулевым расхождением, а $\widehat{R}_{stab} = 0$ отвечает максимальному расхождению. Метрика $R_{class}(h)$ вычислена как доля возмущений, при которых полная сгенерированная последовательность не изменяется.

Результаты. Сводка эксперимента представлена в Таблице 8.

Таблица 8 — Приближенные значения $R_{stab}(f)$ и $R_{class}(h)$ для GPT-4o-mini (20 промптов, 5 возмущений).

Метрика	**Значение**
$R_{stab}(f)$ (среднее $\pm$ ст. откл.)	$0{,}115 \pm 0{,}139$
$R_{class}(h)$	0,050
$R_{stab}(f) \geqslant R_{class}(h)$	Да
Доля сохраненных дискретных выходов	5%
Число промптов	20
Число возмущений на промпт	5

Обсуждение. На рассматриваемой выборке получено $\widehat{R}_{stab}(f) = 0{,}115 > R_{class}(h) = 0{,}050$, что эмпирически подтверждает гипотезу: генеративная метрика устойчивости не ниже дискретной. Генеративная метрика $R_{stab}(f)$ обладает большей гранулярностью: даже когда дискретный выход изменился и $R_{class}(h) = 0$, значение $R_{stab}(f)$ может быть положительным, если распределения на уровне токенов остаются частично согласованными. Это показывает, что $R_{stab}(f)$ улавливает информацию о «скрытой устойчивости» распределений, теряемую при дискретизации.

Оба значения метрик невелики для доброкачественных возмущений, что объясняется авторегрессионной природой LLM: даже минимальное изменение входа с расстоянием Левенштейна 1–2 влияет на внутренние распределения, так как каждый последующий токен зависит от всех предыдущих. Диапазон значений $R_{stab}(f) \in [0{,}005;\ 0{,}540]$ указывает на высокую вариативность чувствительности в зависимости от конкретного промпта.

Ограничения. Вычисление выполнено через API в режиме независимой генерации, а не teacher forcing. Строгий расчет $R_{stab}(f)$ по Определению 2.3 предполагает доступ к весам модели. Погрешность вносит и top-20 аппроксимация logprobs: хвостовая вероятностная масса занижается, что ведет к завышению $\widehat{R}_{stab}$ и занижению расхождения. Выборка содержит 20 промптов

с доброкачественными возмущениями. Стандартное отклонение R_{stab}, равное 0,139, превышает среднее 0,115, что свидетельствует о высокой дисперсии оценки. На adversarial-сценарии результаты не переносятся напрямую. При всех этих ограничениях полученные данные не противоречат гипотезе $R_{stab}(f) \geqslant R_{class}(h)$, однако статистически значимое подтверждение требует расширения выборки в последующих исследованиях.

Выводы по главе

Проведенные эксперименты выявили устойчивые закономерности. Адаптивная атака ASA достигает ASR = 73,8% на Gemma-3-4B, а переносимость между открытыми моделями составляет TSR = 50,5–62,6%. Стратегия защиты, опирающаяся на сокрытие архитектуры, при такой переносимости не обоснована. GPT-4 и Claude-3 уязвимы в меньшей степени, однако и для них ASR = 42,9–45,7%, что нельзя считать приемлемым.

Троянские закладки остаются нерешенной проблемой. Recall, вычисляемый через BLEU-сопоставление предсказанных и истинных триггеров, достигает лишь $\approx 0{,}17$ при базовом уровне случайного совпадения 0,14. REASR при этом близок к 0,99. Разрыв между двумя метриками имеет практический смысл: заставить модель генерировать целевой выход произвольным входом легко, а восстановить исходные триггеры существующими методами практически невозможно. Обратная инженерия триггеров в ее нынешнем состоянии непригодна для аудита моделей.

Гетерогенные комитеты из 5–7 моделей показали наибольшую эффективность среди защитных механизмов, снижая ASR на 47–55 п.п. Два фактора ограничивают их применимость: линейный рост вычислительных затрат и зависимость от разнообразия состава. Предварительные наблюдения указывают на то, что гомогенные комитеты из моделей одного семейства выигрывают меньше вследствие коррелированности ошибок, однако количественная оценка этого эффекта требует отдельного исследования. Определение минимального уровня архитектурного разнообразия, необходимого для устойчивого снижения ASR, остается открытой задачей.

Валидация метрик на GPT-4o-mini дала $R_{stab}(f) = 0{,}115$ при $R_{class}(h) = 0{,}050$. Генеративная метрика фиксирует дистрибутивную нестабильность, невидимую при дискретной оценке. Эксперимент ограничен 20 промптами с доброкачественными возмущениями и top-20 аппроксимацией logprobs. Расширение выборки и проверка на adversarial-сценариях составляют ближайшую задачу.

Глава 5. Программная реализация комплекса оценки и повышения устойчивости LLM

Введение к главе

Метрики и алгоритмы из Глав 2–3 нуждаются в программных инструментах для автоматизации генерации атак, сбора ответов моделей и вычисления показателей устойчивости. С этой целью разработаны два зарегистрированных комплекса. Комплекс JudgeGuard, получивший свидетельство о государственной регистрации программы для ЭВМ № 2025687702, предназначен для оценки устойчивости моделей-оценщиков к состязательным атакам. Комплекс TrojanArmor, зарегистрированный под № 2025684247, выявляет и анализирует троянские закладки в больших языковых моделях. Совместно эти комплексы покрывают полный цикл оценки безопасности: от генерации атакующих воздействий и получения ответов моделей до вычисления метрик устойчивости.

Модульная архитектура обоих комплексов рассмотрена в Разделе 5.1. Обоснование выбора стека технологий дано в Разделе 5.2. Расширение комплексов на агентские LLM-системы через модуль MCPSec описано в Разделе 5.3, а количественные характеристики вычислительной производительности приведены в Разделе 5.4.

5.1 Архитектурная модель программных комплексов JudgeGuard и TrojanArmor

Комплексы JudgeGuard и TrojanArmor решают разные задачи. JudgeGuard оценивает устойчивость моделей-оценщиков формата LLM-as-a-Judge к состязательным атакам, тогда как TrojanArmor направлен на обнаружение и анализ троянских закладок, известных как бэкдоры, в больших языковых моделях. Несмотря на это, оба комплекса опираются на единую четырехкомпонентную архитектуру с разделением ответственности между модулями.

5.1.1 Модульная структура системы

Архитектура каждого из комплексов включает четыре основных модуля:

1. **Модуль генерации атакующих воздействий (Attack Generator Module, AGM).** Модуль отвечает за создание вредоносных промптов, триггерных последовательностей и состязательных примеров. Применительно к JudgeGuard в нем реализован алгоритм ASA, описанный в Разделе 3.4.2 Главы 3: эволюционная оптимизация промптов, инъекции в контекст и CUA-атаки типа Comparative Undermining Attack. В TrojanArmor модуль генерирует кандидатные триггеры методом обратной инженерии внутренних представлений модели с использованием алгоритмов GCG, PEZ и GBDA. Ради универсальности комплекс TrojanArmor дополнительно поддерживает атаки для моделей компьютерного зрения, такие как BadNet, Blended и WaNet, но в настоящей диссертации задействованы исключительно LLM-компоненты.
2. **Маршрутизатор моделей / API-шлюз (Model Router / API Gateway, MRAG).** Дает единый интерфейс доступа к локальным моделям через vLLM или HuggingFace pipeline и к облачным API провайдеров OpenAI, Anthropic и Google. В маршрутизаторе реализованы балансировка нагрузки, повторные запросы при превышении лимитов API, кэширование ответов и логирование взаимодействий. Конфигурация маршрутизации задается декларативно в формате YAML.
3. **Модуль вычисления метрик (Evaluation Engine, EE).** Отвечает за вычисление количественных характеристик устойчивости и эффективности атак, введенных в Главе 4: ASR, TSR, Recall, REASR, а также агрегированной метрики RUT. Архитектурно модуль устроен как набор независимых функций-вычислителей. Новые метрики добавляются без изменения существующего кода, что соответствует принципу открытости/закрытости.
4. **Уровень хранения данных (Storage Layer, SL).** Хранит промежуточные и итоговые результаты экспериментов. Предусмотрены два режима работы. Файловое хранилище в формате JSON/JSONL ориентировано на воспроизводимость и версионирование результатов. Реляционное хранилище на базе PostgreSQL поддерживает аналити-

ческие SQL-запросы по метрикам. Каждая запись сопровождается метаданными эксперимента: идентификатором запуска, временными метками, конфигурацией модели и параметрами генерации.

5.1.2 Диаграмма компонентов и потоков данных

На Рис. 5.1 представлена диаграмма компонентов и потоков данных программного комплекса.

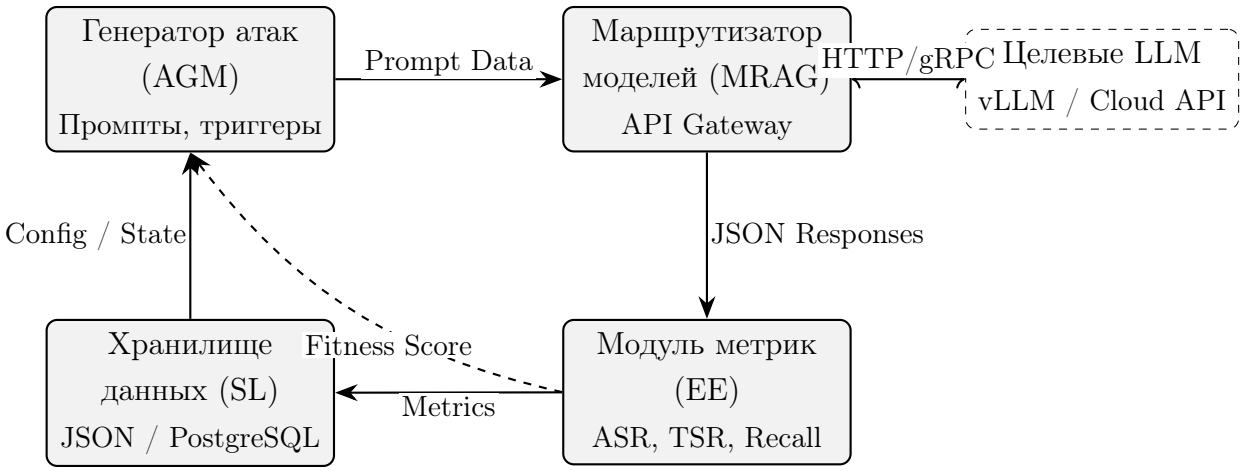


Рисунок 5.1 — Диаграмма компонентов и потоков данных программных комплексов JudgeGuard / TrojanArmor.

Поток данных в системе организован циклически. Генератор AGM формирует атакующие промпты или триггерные последовательности, которые поступают в маршрутизатор MRAG в виде структурированных запросов Prompt Data. Маршрутизатор выбирает целевую модель и формирует HTTP- или gRPC-запрос к бэкенду инференса. Полученные ответы модели в формате JSON Responses передаются в модуль EE для сопоставления с эталонными данными и преобразования в набор количественных метрик Metrics. Далее вычисленные метрики записываются в хранилище SL. Для итеративных алгоритмов атаки, в частности ASA, предусмотрен канал обратной связи: модуль EE возвращает значение целевой функции Fitness Score в генератор AGM, направляя эволюционный поиск на последующих итерациях.

5.1.3 Конвейер обработки данных

Конвейер обработки данных реализует последовательность операций, начиная с загрузки исходных промптов и заканчивая получением итоговых метрик. Формально конвейер представляется как композиция отображений:

$$\Pi = \mathrm{SL}_{\mathrm{write}} \circ \mathrm{EE} \circ \mathrm{MRAG} \circ \mathrm{AGM} \circ \mathrm{SL}_{\mathrm{read}},$$

где $\mathrm{SL}_{\mathrm{read}} : \mathrm{Config} \to (\mathcal{X}, \mathrm{Params})$ – загрузка конфигурации и исходных данных, $\mathrm{AGM} : (\mathcal{X}, \mathrm{Params}) \to \mathcal{X}^N$ – генерация N атакующих промптов, $\mathrm{MRAG} : \mathcal{X}^N \to \mathcal{Y}^N$ – получение ответов моделей, $\mathrm{EE} : (\mathcal{X}^N, \mathcal{Y}^N) \to \mathbb{R}^M$ – вычисление M метрик, и $\mathrm{SL}_{\mathrm{write}} : \mathbb{R}^M \to \mathrm{Storage}$ – сохранение результатов.

Каждый этап конвейера реализован как отдельный Python-модуль с фиксированным интерфейсом, что допускает независимое тестирование, замену и параллельный запуск модулей. На этапе MRAG поддерживается пакетная обработка запросов с настраиваемым размером пакета. Без пакетирования GPU-ресурсы при локальном инференсе расходуются неэффективно.

5.2 Выбор стека технологий и инструментальных средств

Стек технологий подбирался с учетом производительности при работе с LLM и совместимости с экосистемой машинного обучения. Дополнительным критерием служила простота развертывания экспериментальной среды.

5.2.1 Основные технологии

Python 3.10+ выбран как основной язык разработки. Python доминирует в экосистеме машинного обучения и располагает зрелыми библиотеками для работы с LLM. Встроенная поддержка asyncio дает параллельное взаимодействие с несколькими API.

В качестве фреймворка глубокого обучения применяется PyTorch [143]. Фреймворк реализует GPU-ускорение через CUDA, смешанную точность FP16/BF16 для снижения потребления памяти и автоматическое дифференцирование, необходимое для градиентных атак. Начиная с версии 2.0 PyTorch поддерживает компиляцию моделей через `torch.compile`.

Библиотека **HuggingFace Transformers** [144] дает единый API для загрузки, конфигурирования и инференса предобученных моделей. Через HuggingFace Hub доступно более 200 000 моделей, включая семейства GPT, LLaMA, Mistral, Gemma и другие архитектуры, задействованные в экспериментальной оценке Главы 4.

Высокопроизводительный инференс реализован через vLLM [145], построенный на механизме PagedAttention для оптимального управления GPU-памятью. В отличие от стандартного HuggingFace pipeline, vLLM реализует непрерывное пакетирование запросов, существенно повышающее пропускную способность при массовом тестировании. Количественное сравнение производительности двух подходов приведено в Таблице 9.

REST API маршрутизатора моделей построен на связке FastAPI + Uvicorn. FastAPI автоматически генерирует документацию OpenAPI, валидирует входные данные через Pydantic-схемы и нативно поддерживает асинхронную обработку запросов. Uvicorn, ASGI-сервер на основе uvloop, способен обрабатывать тысячи конкурентных соединений при минимальных накладных расходах.

PostgreSQL / файловое хранилище. Результаты экспериментов хранятся по двухуровневой схеме. Файлы JSONL предназначены для оперативного хранения промежуточных результатов и легко версионируются через Git. PostgreSQL служит для долгосрочного хранения и аналитических SQL-запросов по метрикам.

5.2.2 Сравнение производительности vLLM и стандартного HuggingFace pipeline

Переход от стандартного HuggingFace pipeline к vLLM стал одним из принципиальных инженерных решений. Механизм PagedAttention [145], лежащий в основе vLLM, разбивает KV-кэш внимания на блоки фиксированного размера.

Управление блоками организовано по аналогии с виртуальной памятью операционной системы, что устраняет фрагментацию GPU-памяти и увеличивает число параллельно обрабатываемых запросов.

Таблица 9 — Сравнение производительности vLLM и HuggingFace pipeline (модель LLaMA-2-13B, GPU: NVIDIA A100 80GB, длина генерации: 256 токенов).

Характеристика	**HF Pipeline**	**vLLM**	**Прирост**
Пропускная способность (запр./сек)	2,8	14,1	×5,0
Задержка (p50), мс	1420	310	×4,6
Задержка (p99), мс	3200	780	×4,1
Утилизация GPU-памяти, %	62	91	+29 п.п.
Макс. параллельных запросов	4	32	×8,0
Время полной оценки (1000 промптов)*, мин	47,2	9,8	×4,8

*Полная оценка 1000 промптов включает многомодельную оценку комитетом из k моделей с повторными запросами для агрегации, а также накладные расходы на предобработку/-постобработку. При пропускной способности 2,8 запр./сек время 47,2 мин соответствует $47{,}2 \times 60 \times 2{,}8 \approx 7930$ модельным запросам, т. е. ∼8 модельных вызовов на промпт.

Согласно Таблице 9, пропускная способность возросла в пять раз, а время полной оценки 1000 промптов сократилось с 47 до 10 минут. Расхождение между расчетным временем для 1000 единичных запросов и реально затраченным обусловлено тем, что каждый промпт проходит многомодельную оценку комитетом. Для итеративных алгоритмов атаки ASA, где каждая генерация эволюционного алгоритма порождает N обращений к целевой модели, подобное ускорение критически важно.

На этих технологиях построены комплексы JudgeGuard и TrojanArmor. Далее рассмотрен модуль MCPSec, расширяющий оба комплекса на задачу оценки безопасности агентских LLM-систем.

5.3 Модуль оценки устойчивости агентских LLM-систем (MCPSec)

Агентские LLM-системы формируют входные последовательности из данных внешних инструментов посредством протокола MCP [34]. Для тестирования таких систем разработан модуль MCPSec [3], интегрированный в четырехкомпонентную архитектуру, описанную в Разделе 5.1. В рамках интеграции генератор атак AGM дополнен шаблонами MCP-инъекций, маршрутизатор MRAG получил адаптеры для MCP-транспорта, а модуль метрик EE расширен специализированными метриками безопасности агентского взаимодействия. Ниже описана архитектура MCP-протокола и механизмы тестирования, после чего представлены алгоритмы защиты.

5.3.1 Архитектура протокола MCP

Model Context Protocol задает клиент-серверную архитектуру, в которой LLM-агент выступает клиентом и взаимодействует с набором серверов, предоставляющих доступ к инструментам, источникам данных и промптам. В основе протокола лежит спецификация JSON-RPC 2.0 с двумя поддерживаемыми транспортными механизмами:

- **stdio (Standard I/O)** – синхронный транспорт через стандартные потоки ввода/вывода, предназначенный для локальных серверов, запускаемых как дочерние процессы хост-приложения.
- **SSE (Server-Sent Events)** – асинхронный HTTP-транспорт для удаленных серверов с поддержкой потоковой передачи данных.

Основные методы протокола включают:

- `tools/list` – получение списка доступных инструментов с описаниями и JSON-схемами параметров.
- `tools/call` – вызов инструмента с передачей аргументов и получением результата `CallToolResult`.
- `prompts/list` – получение списка доступных шаблонов промптов.
- `resources/read` – чтение данных из ресурсов, предоставляемых сервером.

5.3.2 Диаграмма последовательности взаимодействия

На Рис. 5.2 представлена UML-диаграмма последовательности, иллюстрирующая типичный цикл взаимодействия между LLM-агентом и MCP-сервером.

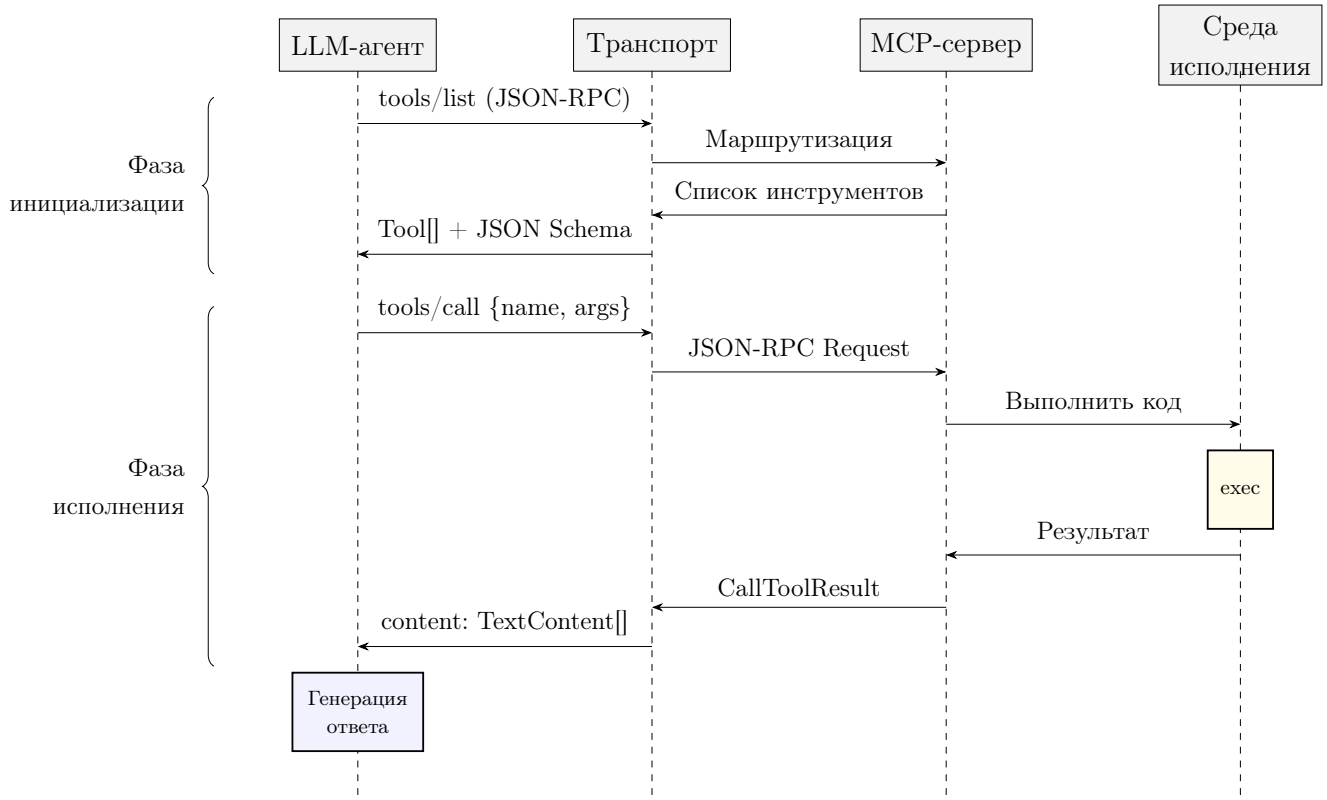


Рисунок 5.2 — UML-диаграмма последовательности взаимодействия LLM-агента с MCP-сервером.

5.3.3 Уязвимости протокола: от инъекции запросов к перехвату инструментов

Протокол MCP содержит критическую уязвимость на стыке обработки естественного языка и выполнения программного кода. LLM-агент решает, какой инструмент вызвать, на основе контекста, который может включать данные из ненадежных источников. Это создает вектор атаки «инъекция запроса → перехват инструмента», известный как Prompt Injection to Tool Hijacking.

Анализ спецификации MCP выявил три архитектурные уязвимости:

1. **Отсутствие аттестации возможностей (Capability Attestation Absence).** Серверы MCP самостоятельно декларируют свои возможности при инициализации соединения, и ни клиент, ни третья сторона не проводят верификацию. Протокол допускает изменение capabilities в ходе сессии без валидации. Скомпрометированный сервер может эскалировать привилегии от ограниченных к неограниченным уже после установления соединения.
2. **Неаутентифицированный сэмплинг (Unauthenticated Sampling).** Механизм сэмплинга позволяет серверам запрашивать генерацию LLM-ответов, присваивая сообщениям роль «user». Протокол не аутентифицирует источник сообщений, и пользователь не может визуально отличить сообщения, инжектированные сервером, от собственных запросов. Это открывает канал для прямой протокольной инъекции запроса.
3. **Неявное распространение доверия (Implicit Trust Propagation).** В конфигурациях с несколькими MCP-серверами границы изоляции не определены. Ответы инструментов одного сервера влияют на вызовы инструментов другого без отслеживания происхождения данных. Компрометация одного сервера каскадно распространяет привилегии через все подключенные сервисы.

Контекст LLM-агента складывается из системного промпта, пользовательского запроса и результатов предыдущих вызовов инструментов. Злоумышленник, контролируя результаты вызовов через скомпрометированный MCP-сервер, внедряет вредоносную инструкцию в возвращаемые данные. LLM-агент не различает инструкции и данные, поэтому модифицированный контекст вынуждает его вызвать произвольный инструмент с вредоносными аргументами. Формальное описание этой атаки как косвенной инъекции запроса приведено в Разделе 1 Главы 1.

Этот механизм является частным случаем косвенной инъекции запроса. Однако в контексте MCP последствия серьезнее: вызовы инструментов приводят к выполнению произвольного кода, модификации файловой системы и отправке сетевых запросов, а побочные эффекты затрагивают реальную среду исполнения.

В программном комплексе реализован модуль автоматизированного тестирования MCP-серверов на устойчивость к этому классу атак. Модуль

генерирует вредоносные ответы инструментов, отслеживает последующие вызовы агента и оценивает успешность перехвата.

5.3.4 Экспериментальная оценка уязвимостей протокола: MCPBench

Для количественной оценки влияния архитектурных уязвимостей MCP на безопасность агентских систем разработан исследовательский тестовый набор MCPBench [3]. Набор объединяет сценарии атак на агентские LLM-системы с инфраструктурой MCP. Тестирование выполнено на 847 сценариях с привлечением моделей Claude-3.5-Sonnet, GPT-4o и Llama-3.1-70B.

Для изоляции вклада протокольных уязвимостей от общих уязвимостей LLM проведено сравнение MCP-реализаций с базовыми не-MCP интеграциями инструментов. Результаты сведены в Таблице 10.

Таблица 10 — Сравнение успешности атак: MCP vs. базовая (не-MCP) интеграция инструментов.

Тип атаки	**Базовая ASR**	**MCP ASR**	**Δ**
Косвенная инъекция	31,2%	47,8%	+16,6 п.п.
Манипуляция ответами инструментов	28,4%	52,1%	+23,7 п.п.
Межсерверное распространение	19,7%	61,3%	+41,6 п.п.
Среднее	**26,4%**	**53,7%**	**+27,3 п.п.**

Архитектурные решения MCP усиливают уязвимость агентских систем на 16,6–41,6 процентного пункта относительно не-MCP интеграций. Для косвенных инъекций прирост составил +16,6 п.п. Межсерверные атаки дали наибольшее увеличение ASR, составившее +41,6 п.п. Разница в последнем случае объясняется отсутствием изоляции между серверами и неявным распространением доверия.

Полученные значения ASR относятся к исследованным реализациям агентов, выбранным сценариям инструментов и конкретным конфигурациям MCP-взаимодействия. Они подтверждают наличие и практическую значимость

данного класса уязвимостей, но не покрывают все пространство возможных агентских архитектур.

5.3.5 Алгоритмическое обеспечение безопасности агентского взаимодействия

На основе уязвимостей протокола MCP, выявленных в Разделе 5.3.3, в модуль MCPSec включены два алгоритма программной защиты агентских систем. Первый, AttestMCP, реализует криптографическую аттестацию сообщений. Второй, паттерн Commit Boundary, изолирует исполнение кода.

Алгоритм криптографической аттестации и маршрутизации сообщений (AttestMCP)

Алгоритм AttestMCP ограничивает множество вызовов инструментов, доступных LLM-агенту, заранее зафиксированной таблицей полномочий и заверяет целостность каждого RPC-пакета HMAC-подписью. Защита построена на двух разнородных уровнях. Авторизацию задает таблица полномочий: она определяет, какие инструменты и с какими аргументами допустимы в сессии. Целостность обеспечивает HMAC-подпись, которая аутентифицирует источник пакета на транспортном слое и защищает его от модификации при передаче.

Пусть $\mathcal{P} = \{(s_i, t_j, a_k)\}$ – таблица полномочий, где s_i – идентификатор сессии, t_j – идентификатор инструмента, a_k – спецификация допустимых аргументов. Таблица формируется на фазе инициализации сессии, до поступления в контекст агента данных из недоверенных источников. В фазе исполнения $\mathcal{P}$ остается неизменной, и у LLM-агента нет операции, расширяющей ее. Ограничение множества достижимых действий агента таблицей $\mathcal{P}$ образует основной механизм защиты, не зависящий от содержимого контекста.

Каждый RPC-пакет сопровождается подписью $\sigma = \mathrm{HMAC}(K_{session}, s_i \| t_j \| \mathrm{hash}($
где $K_{session}$ – сессионный ключ, установленный при инициализации MCP-соединения, a args – фактические аргументы вызова. Подпись связывает пакет с

сессией и инструментом и фиксирует целостность переданных аргументов через их хеш.

Алгоритм 3: AttestMCP – криптографическая аттестация вызовов инструментов

Вход: RPC-пакет $\text{pkt} = (\text{method}, \text{params}, \sigma)$, таблица полномочий $\mathcal{P}$, сессионный ключ $K_{session}$.
Выход: Результат выполнения или отказ.
Извлечь $(\text{session\_id}, \text{tool\_id}, \text{args}) \leftarrow \text{ParseRPC}(\text{pkt})$.
Вычислить $\sigma' \leftarrow \text{HMAC}(K_{session}, \text{session\_id} \| \text{tool\_id} \| \text{hash}(\text{args}))$.
if $\sigma \neq \sigma'$ **then**
    Вернуть Error(INVALID_SIGNATURE). ▷ Подпись недействительна
end if
Проверить $(s_i, t_j, a_k) \in \mathcal{P}$ для $s_i = \text{session\_id}$, $t_j = \text{tool\_id}$.
if запись не найдена **then**
    Вернуть Error(PERMISSION_DENIED). ▷ Нет полномочий
end if
if $\text{args} \notin \text{AllowedArgs}(a_k)$ **then**
    Вернуть Error(ARGS_VIOLATION). ▷ Недопустимые аргументы
end if
$\text{result} \leftarrow \text{ExecuteTool}(\text{tool\_id}, \text{args})$. ▷ Маршрутизировать к серверу
Записать $(\text{pkt}, \text{result}, \text{timestamp})$ в журнал аудита.
Вернуть CallToolResult(result).

Анализ сложности. Вычисление HMAC-SHA256 занимает $O(|\text{args}|)$. Поиск записи в таблице полномочий, реализованный через хеш-таблицу, выполняется за $O(1)$ амортизированно, а валидация аргументов по JSON-схеме требует $O(|\text{schema}|)$ операций сравнения.

На практике суммарные накладные расходы не превысили 0,1 мс на один вызов при $|\text{args}| \leqslant 4096$ байт. Замеры проводились на CPU Intel Xeon Gold 6248R с тактовой частотой 3,0 ГГц, инференс LLM выполнялся на NVIDIA A100. Время аттестации оказалось на три порядка меньше типичной задержки инференса, составляющей 300–800 мс для моделей класса LLaMA-2-13B.

Границы гарантий. Уровни авторизации и целостности решают разные задачи, и их функции не взаимозаменяемы. HMAC-подпись подтверждает источник и целостность пакета, но не намерение вызова. Вызов инструмента, сформированный LLM-агентом под действием инъекции, проходит через того

же легитимного клиента, что и вызов по запросу пользователя, и получает корректную подпись. Подпись сама по себе не отделяет санкционированный пользователем вызов от навязанного инъекцией. Это разграничение обеспечивает таблица полномочий: инъекция не выводит агента за пределы $\mathcal{P}$, поскольку таблица зафиксирована до поступления недоверенных данных и в сессии не изменяется.

Криптографический уровень закрывает отдельный вектор: межсерверное распространение доверия из Раздела 5.3.3. Скомпрометированный MCP-сервер не владеет ключом $K_{session}$, который хранится на транспортном слое за пределами контекстного окна модели. Такой сервер не может подделать пакет от имени клиента или другого сервера, что блокирует каскадную эскалацию привилегий.

Часть риска AttestMCP не устраняет. Если инъекция вынуждает агента вызвать разрешенный инструмент с разрешенными аргументами для вредоносной цели, пакет проходит обе проверки. Сужение этого остаточного риска возложено на спецификацию аргументов a_k и на паттерн Commit Boundary из Раздела 5.3.5: чем уже допустимое множество аргументов, тем меньше пространство злоупотребления внутри $\mathcal{P}$.

Формализация AttestMCP как конечного автомата. Протокол описывается конечным автоматом $\mathcal{M} = (S, \Sigma, \delta, s_{init}, F)$ с пятью состояниями $S = \{s_{init}, s_{auth}, s_{exec}, s_{return}, s_{error}\}$. Входной алфавит Σ включает RPC-пакеты, результаты верификации и результаты исполнения. Терминальными являются состояния $F = \{s_{return}, s_{error}\}$.

Переходы устроены так: из s_{init} по приему RPC-пакета автомат переходит в s_{auth}, то есть $s_{init} \xrightarrow{\text{pkt}} s_{auth}$. Переход $s_{auth} \to s_{exec}$ происходит только при совпадении подписи $\sigma = \sigma'$ и наличии записи $(s_i, t_j, a_k) \in \mathcal{P}$ с допустимыми аргументами. Несовпадение подписи, отсутствие записи или нарушение спецификации аргументов ведет в s_{error}. Завершение исполнения порождает $s_{exec} \xrightarrow{\text{ok}} s_{return}$ или $s_{exec} \xrightarrow{\text{fail}} s_{error}$.

Автомат обладает инвариантом: каждый исполненный вызов удовлетворяет таблице полномочий:

$$\forall \text{ путь } (s_{init}, \ldots, s_{exec}) : \quad \exists\, a_k : (\text{session\_id}, \text{tool\_id}, a_k) \in \mathcal{P} \wedge \text{args} \in \text{AllowedArgs}(a_k$$

Поскольку $\mathcal{P}$ фиксируется на фазе инициализации, инвариант ограничивает множество действий агента сверху независимо от инструкций, внедренных в его контекст. Инъекция запроса не расширяет полномочия за пределы $\mathcal{P}$, а

верификация HMAC не позволяет обойти это ограничение подменой пакета на транспортном уровне.

Алгоритм программной изоляции исполнения: паттерн Commit Boundary

Код, генерируемый LLM-агентом, не должен выполняться непосредственно в хост-среде. Паттерн Commit Boundary запускает его в контейнеризированном окружении и валидирует результат перед передачей обратно, реализуя принцип минимальных привилегий.

Алгоритм 4: Commit Boundary – изолированное исполнение кода

Вход: Код, сгенерированный LLM, code $\in \mathcal{X}$; лимит ресурсов limits $= (T_{max}, M_{max}, \text{net})$;
        флаг HITL (human-in-the-loop) $h \in \{0, 1\}$.
Выход: Результат выполнения или отказ.
ast $\leftarrow$ ParseAST(code). ▷ Разбор абстрактного синтаксического дерева
if ast = NULL **then**
    Вернуть Error(PARSE_FAILED). ▷ Синтаксическая ошибка
end if
risk $\leftarrow$ StaticAnalysis(ast). ▷ Статический анализ: syscall, network, fs
if risk > THRESHOLD_HIGH **then**
    if $h = 0$ **then**
        Вернуть Error(HIGH_RISK_BLOCKED).
    else
        approved $\leftarrow$ RequestHumanApproval(code, risk).
        if $\neg$approved **then**
            Вернуть Error(HUMAN_REJECTED).
        end if
    end if
end if
container $\leftarrow$ AllocateContainer(limits). ▷ Docker с ограничениями cgroups
Настроить container: таймаут T_{max}, память M_{max}, сеть net.

21: (result, exit_code) ← RunInContainer(container, code).
22: ReleaseContainer(container). ▷ Освободить ресурсы
23: **if** exit_code ≠ 0 или таймаут **then**
24: **Вернуть** Error(EXECUTION_FAILED, exit_code).
25: **end if**
26: validated ← ValidateOutput(result). ▷ Проверка выходных данных
27: Записать (code, result, risk, timestamp) в журнал аудита.
28: **Вернуть** CommitResult(result). ▷ Зафиксировать результат

Анализ сложности. Доминирует собственно выполнение кода в контейнере, ограниченное таймаутом T_{max}. Накладные расходы на изоляцию распределяются так:

- Разбор AST: $O(|\text{code}|)$, линейно по размеру кода.
- Статический анализ: $O(|\text{ast}| \cdot |\text{rules}|)$, где $|\text{rules}|$ – количество правил обнаружения, обычно 50–100.
- Аллокация контейнера Docker: 200–500 мс при холодном старте, около 50 мс при использовании пула предсозданных контейнеров.
- В сумме 0,3–0,8 с, что сопоставимо с временем генерации кода LLM и приемлемо для интерактивных сценариев.

Два эшелона защиты действуют последовательно: статический анализ на этапе разбора и контейнерная изоляция при выполнении. Для высокорисковых операций активируется режим HITL, реализующий принцип градуированного доверия из Главы 3.

Компоненты комплексной защиты модуля MCPSec

Модуль MCPSec [3] объединяет алгоритмы AttestMCP и Commit Boundary с четырьмя дополнительными механизмами защиты.

Сертификаты возможностей (Capability Certificates) ограничивают допустимый набор операций каждого сервера, исключая эскалацию привилегий. Для разграничения источников сообщений служит маркировка происхождения (Origin Tagging): визуальная и программная метка отделяет вывод механизма сэмплинга от пользовательских запросов. Без явного разрешения пользователя межсерверная передача данных блокируется механизмом принудительной изо-

ляции (Isolation Enforcement). Перехват сообщений нейтрализуется защитой от повторов (Replay Protection) на основе временных меток и одноразовых nonce.

На тестовом наборе MCPBench фреймворк снизил среднюю успешность атак с 53,7% до 12,4%, то есть на 41,3 п.п. Дополнительная задержка обработки одного сообщения в установившемся режиме при использовании пула предсозданных контейнеров не превысила 8,3 мс, что пренебрежимо мало относительно времени инференса LLM, составляющего 300–800 мс для моделей класса LLaMA-2-13B.

5.4 Оценка вычислительной сложности и производительности программного комплекса

В разделе приведена количественная оценка производительности программного комплекса по двум показателям: зависимость задержки от длины входного контекста при активных защитных фильтрах и пропускная способность при разных конфигурациях инференса.

5.4.1 Анализ задержки в зависимости от длины контекста

Время обработки одного запроса складывается из четырех компонентов: предобработка, включающая токенизацию и форматирование промпта, инференс модели, работа защитных фильтров и вычисление метрик. Через $n = |x|$ обозначена длина входной последовательности в токенах.

Для Transformer-моделей время инференса растет квадратично – $O(n^2 \cdot d)$, где d – размерность модели. Защитные фильтры, а именно перплексийный фильтр и контент-модерация, работают за $O(n)$ каждый. На Рис. 5.3 представлена зависимость суммарной задержки от длины контекста при различных конфигурациях.

Рис. 5.3 показывает: накладные расходы на защитные компоненты составляют около 28% при типичной длине контекста $n = 2048$, что согласуется с декомпозицией CO в Таблице 11, и убывают до примерно 12% при $n = 8192$.

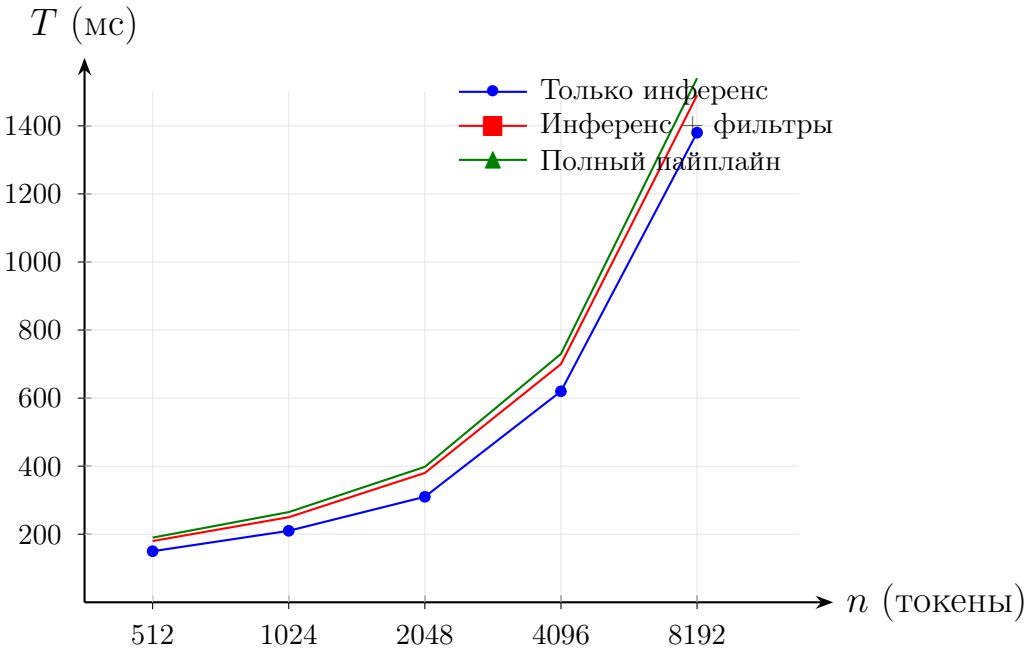


Рисунок 5.3 — Зависимость задержки обработки запроса от длины входного контекста (модель LLaMA-2-13B, vLLM, GPU: NVIDIA A100).

Фильтры работают за $O(n)$, инференс за $O(n^2)$, поэтому с ростом n доля фильтров в суммарном времени убывает. Защитные механизмы не становятся узким местом.

5.4.2 Сравнение пропускной способности при различных конфигурациях

Пропускная способность замерялась на четырех конфигурациях инференса. Модель – LLaMA-2-13B, длина генерации – 256 токенов, объем тестовой выборки – 1000 запросов:

1. **HF-seq** – HuggingFace pipeline, последовательная обработка;
2. **HF-batch** – HuggingFace pipeline, пакетная обработка (batch size = 8);
3. **vLLM-base** – vLLM без защитных фильтров;
4. **vLLM-full** – vLLM с активированными защитными фильтрами и вычислением метрик.

Результаты представлены на Рис. 5.4.

vLLM-base достигает 14,1 запр./сек, превышая HF-seq (1,4 запр./сек) в 10 раз и HF-batch (2,8 запр./сек) в 5 раз. При включении полного набора за-

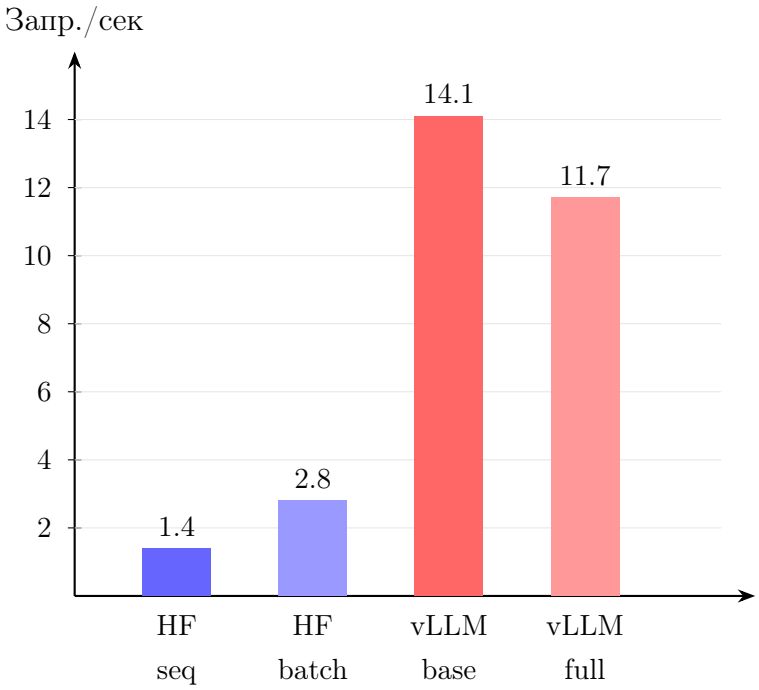


Рисунок 5.4 — Сравнение пропускной способности при различных конфигурациях инференса (модель LLaMA-2-13B, GPU: NVIDIA A100).

щитных фильтров и вычисления метрик (vLLM-full) пропускная способность снижается до 11,7 запр./сек, что составляет потерю 17% от базовой конфигурации. Механизм PagedAttention сохраняет высокую производительность даже при активных компонентах безопасности.

5.4.3 Масштабируемость при многомодельной оценке

При оценке ансамблей моделей, описанных как комитеты в Разделе 3.4.1 Главы 3, критическим параметром становится масштабируемость системы при увеличении числа моделей k. Маршрутизатор MRAG реализует два режима работы:

– **Параллельный инференс (GPU-sharing):** При размещении нескольких моделей на одном GPU с использованием механизма Multi-Process Service (MPS) NVIDIA. Пропускная способность масштабируется сублинейно: $\Theta_k \approx \Theta_1 \cdot k^{0.7}$, где Θ_k – суммарная пропускная способность для k моделей, Θ_1 – пропускная способность для одной модели.

– **Распределенный инференс (multi-GPU):** При размещении каждой модели на отдельном GPU. Пропускная способность масштабируется линейно: $\Theta_k \approx k \cdot \Theta_1$, с учетом накладных расходов на коммуникацию через сеть (∼2–5% потерь при использовании NVLink).

Для экспериментов, описанных в Главе 4, использовалась конфигурация с 4 GPU NVIDIA A100 80GB, позволяющая одновременно обслуживать до 4 моделей размером 13B параметров или 2 модели размером 70B параметров при использовании tensor parallelism. В распределенном режиме, когда каждая модель размещена на отдельном GPU, линейная оценка дает $\Theta_4 \approx 4 \cdot 14{,}1 \approx 54$ запр./сек. Измеренная суммарная пропускная способность составила до 45 модельных запросов в секунду (model inference requests/sec) для 4 моделей 13B. Разрыв с линейной оценкой объясняется накладными расходами оркестрации запросов и синхронизации вердиктов комитета. При оценке датасета из 10 000 промптов с использованием комитета из k моделей генерируется $10\,000 \cdot k$ модельных запросов, в частности 30 000 при $k = 3$, что при измеренной пропускной способности позволяло завершить полную оценку за ∼11 минут при $k = 3$ или ∼15 минут при $k = 4$.

5.4.4 Вычислительная нагрузка алгоритмов безопасности

Суммарная вычислительная нагрузка (Computational Overhead, CO) программного комплекса определяется как отношение дополнительного времени, затрачиваемого на компоненты безопасности, к базовому времени инференса:

$$\mathrm{CO} = \frac{T_{\text{total}} - T_{\text{inference}}}{T_{\text{inference}}} \times 100\%.$$

Измерения проводились для типичной конфигурации: модель LLaMA-2-13B, длина контекста 2048 токенов, бэкенд vLLM. Результаты декомпозиции вычислительной нагрузки приведены ниже:

Согласно Таблице 11, суммарная вычислительная нагрузка равна 28,4%. Перплексийный фильтр вносит наибольшую долю – 13,5%, а AttestMCP добавляет лишь 0,03%, что делает его пригодным для высоконагруженных систем

Таблица 11 — Декомпозиция вычислительной нагрузки компонентов безопасности.

Компонент	**Задержка, мс**	**Доля CO, %**
Базовый инференс (vLLM)	310	–
Перплексийный фильтр	42	13,5
Контент-модерация	28	9,0
Аттестация AttestMCP	0,08	0,03
Вычисление метрик	15	4,8
Логирование и хранение	3	1,0
Итого CO	**88,08**	**28,4**

реального времени. Суммарные накладные расходы составляют 28,4% от базового инференса, что подтверждает работоспособность комплекса с приемлемыми затратами при корректной реализации методов из Глав 2–3.

Выводы по главе

Методы Глав 2–3 реализованы в двух зарегистрированных программных комплексах: JudgeGuard, свидетельство № 2025687702, и TrojanArmor, свидетельство № 2025684247.

Единая четырехкомпонентная архитектура лежит в основе обоих комплексов. Генератор атак AGM, маршрутизатор моделей MRAG, модуль метрик EE и хранилище SL взаимодействуют через определенные интерфейсы, что позволяет добавлять новые модели, метрики и типы атак без модификации ядра. Замена HuggingFace pipeline на vLLM с PagedAttention увеличила пропускную способность в пять раз на NVIDIA A100. Для итеративных алгоритмов вроде ASA, порождающих тысячи модельных запросов на раунд, такой прирост критичен.

Модуль MCPSec расширяет оба комплекса на агентские LLM-системы с протоколом MCP. В его состав входят тестовый набор MCPBench из 847 сценариев атак, криптографическая аттестация AttestMCP, Алгоритм 3, и изолированное исполнение Commit Boundary, Алгоритм 4. На MCPBench средняя

ASR снизилась на 41,3 п.п., с 53,7% до 12,4%, по сравнению с незащищенной конфигурацией, как показано в Разделе 5.3.5.

Совокупная вычислительная нагрузка компонентов безопасности не превышает 30% базового инференса согласно Таблице 11. Фильтры работают за $O(n)$, тогда как инференс требует $O(n^2)$. С ростом длины контекста относительная доля защитных компонентов, следовательно, убывает. Текущая реализация привязана к конкретным API-интерфейсам провайдеров: при смене версий API адаптеры MRAG потребуют обновления. Замеры производительности выполнены на одной аппаратной конфигурации NVIDIA A100, и экстраполяция на другие ускорители нуждается в верификации.

JudgeGuard и TrojanArmor покрывают полный цикл оценки безопасности LLM. Практическое применение комплексов описано в Главе 6.

Глава 6. Практическое применение методов повышения устойчивости

Введение к главе

Лабораторные эксперименты и эксплуатируемые системы разделяет ряд практических задач: выбор метрик эффективности внедрения, согласование требований безопасности с вычислительным бюджетом, адаптация защит к специфике предметной области. Методология внедрения защитных механизмов опирается на технические и экономические метрики оценки, а практические примеры для NLP-систем и критически значимых приложений дополнены анализом тенденций развития технологий.

6.1 Методология внедрения методов повышения устойчивости

Внедрение методов повышения устойчивости LLM в реальные системы не сводится к применению отдельных алгоритмов. Необходим систематический подход, учитывающий специфику задачи, доступные ресурсы и требования к безопасности. Дискретное пространство токенов LLM накладывает дополнительные ограничения на выбор и адаптацию методов.

6.1.1 Этапы внедрения методов повышения устойчивости

Предлагается методология внедрения, основанная на жизненном цикле управления рисками и разработки безопасного программного обеспечения [46; 146]. Методология включает пять последовательных этапов.

1. Анализ требований и оценка рисков. Требования к устойчивости и безопасности системы определяются на основе ее назначения и потенциальных последствий сбоев. Оценка рисков проводится с применением стандартных

методологий, в частности NIST AI Risk Management Framework [46] и OWASP Top 10 для LLM [45], описанных в Главе 1. На этом же этапе идентифицируются релевантные угрозы, перечисленные в Главе 1, Раздел 1.2.2: prompt injection, включая атаки на LLM-as-a-Judge и защищенные системы, а также троянские атаки с учетом сложности их обнаружения. Результат этого этапа записывается как приоритизированный список рисков:

$$\mathcal{R}_{\text{priority}} = \text{Prioritize}(\{(a_i, v_j, P_{\text{exploit}}(v_j|a_i), \text{ImpactValue}(v_j))\}),$$

где $a_i \in \mathcal{A}$ – тип атаки, $v_j \in \mathcal{V}$ – уязвимость, P_{exploit} – вероятность эксплуатации, ImpactValue – количественная оценка ущерба, как определено в Определениях 1.3 и 1.4 Главы 1. Критичность каждой пары атака-уязвимость вычисляется как $c_{ij} = P_{\text{exploit}}(v_j|a_i) \cdot \text{ImpactValue}(v_j)$ и используется для приоритизации.

Замечание 6.1. Эмпирическое наблюдение: ограниченная эффективность защиты от неизвестных атак. Эффективность методов защиты и повышения устойчивости, разработанных и протестированных на известном наборе атак $\mathcal{A}_{\text{known}}$, может снижаться при столкновении с новыми, непредвиденными атаками $\mathcal{A}_{\text{new}}$ в реальных условиях. Степень снижения эффективности зависит от способности защитного механизма обобщаться на новые типы атак [147]. Даже общие принципы защиты имеют пределы обобщения, если атака нацелена на уязвимости, не учтенные при проектировании защитного механизма. Формально, если $R_{\mathcal{A}}(f)$ – метрика робастности/безопасности относительно атак из $\mathcal{A}$, то эмпирически наблюдается тенденция $R_{\mathcal{A}_{\text{new}}}(f) \leqslant R_{\mathcal{A}_{\text{known}}}(f)$. Снижение $(R_{\mathcal{A}_{\text{known}}}(f) - R_{\mathcal{A}_{\text{new}}}(f))$ зависит от «расстояния» между $\mathcal{A}_{\text{new}}$ и $\mathcal{A}_{\text{known}}$ и обобщающей способности защиты.

Обоснование. Защитные механизмы часто оптимизируются для противодействия конкретным паттернам или характеристикам известных атак. Новые атаки могут использовать иные векторы или обходить специфические предположения, заложенные при разработке защиты. Обобщающая способность защиты зависит от заложенных в нее принципов. Так, ограничение Липшицевости обобщается лучше, чем фильтры на сигнатурах известных атак.

2. Выбор методов повышения устойчивости. На основе результатов анализа рисков выбираются методы защиты и повышения устойчивости, описанные в Главе 3. При выборе учитываются следующие факторы:

- Эффективность метода против приоритизированных рисков. Экспериментальные данные Главы 4 подтверждают результативность комитетов моделей против атак на LLM-as-a-Judge.
- Совместимость с архитектурой и размером целевой модели.
- Соотношение вычислительных требований, выраженных через метрику CO из Главы 4, с доступными ресурсами.
- Влияние на полезность модели. Приемлемые границы определяются по метрике RUT из Главы 4.

Экспериментальные данные Главы 4 и принцип многоуровневой защиты из Раздела 3.2.2 Главы 3 свидетельствуют в пользу комбинирования методов. Формально выбор сводится к задаче оптимизации: максимизировать снижение суммарного риска при ограниченном бюджете. На практике решение нередко требует экспертной оценки.

3. Реализация и интеграция. Выбранные методы реализуются и встраиваются в систему. На этом этапе при необходимости модифицируется архитектура модели согласно Разделу 3.2 Главы 3, проводится робастное обучение или дообучение по Разделу 3.3, создаются модули пред- и постобработки данных: входные фильтры и санитизаторы, выходные верификаторы по Разделу 3.2.2. Параллельно настраиваются параметры генерации и подключаются системы мониторинга.

4. Тестирование и валидация. Внедренные защитные механизмы оцениваются по метрикам из Главы 4: устойчивость к известным атакам по показателю ASR, полнота обнаружения троянов по Recall и REASR, полезность на легитимных запросах по U-метрикам и компромисс робастности с полезностью по RUT. Параллельно проводится Red Teaming [93; 94] для поиска новых уязвимостей, включая автоматизированные подходы, апробированные на соревнованиях Kaggle и SaTML CTF. Результаты тестирования направляют итеративную доработку защиты.

5. Мониторинг и обновление. После развертывания ведется непрерывный сбор данных о запросах, ответах и срабатываниях защит. Профиль угроз отслеживается по новым публикациям и результатам соревнований. Периодическое повторное тестирование и Red Teaming выявляют деградацию защиты. Быстрая эволюция LLM и методов атак делает этот этап определяющим для долгосрочной безопасности системы.

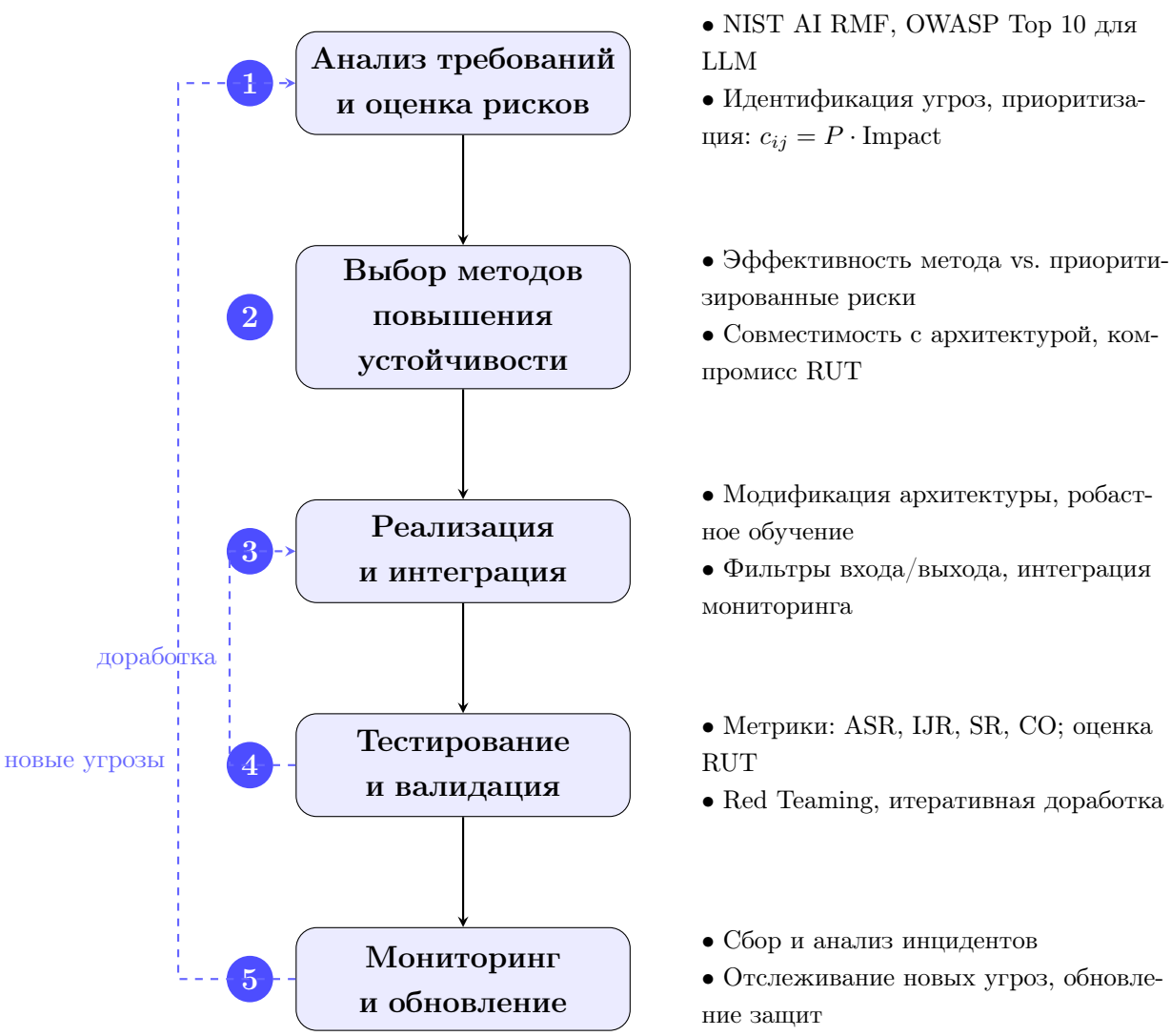


Рисунок 6.1 — Жизненный цикл внедрения методов повышения устойчивости LLM.

Замечание 6.2. Эмпирическое наблюдение: необходимость адаптации защиты. Новые типы атак ($\mathcal{A}_{\text{new}}$) появляются постоянно, а характеристики существующих атак меняются. Защитные механизмы требуют непрерывного мониторинга и обновления. Статические защиты со временем устаревают. Скорость адаптации защиты должна быть сопоставима со скоростью эволюции угроз [148]. Для LLM это особенно актуально: новые векторы атак появляются регулярно.

Методология задает порядок внедрения и удерживает баланс между безопасностью, функциональностью и вычислительной эффективностью. Цикл повторяется на протяжении всей эксплуатации системы.

6.1.2 Оценка эффективности внедрения

Оценка эффективности внедрения не ограничивается техническими показателями безопасности, а охватывает также влияние на производительность и затраты.

1. Технические метрики эффективности. Устойчивость и безопасность оцениваются через ASR, Recall и REASR (Раздел 4.1.1 Главы 4). Разница с базовой моделью фиксируется как $\Delta C = C(f_{\text{защищ}}) - C(f_{\text{base}})$. Полезность контролируется по BLEU, ROUGE, Accuracy и BERTScore на чистых данных: $\Delta U = U(f_{\text{защищ}}) - U(f_{\text{base}})$. Производительность характеризуется вычислительной нагрузкой CO, включающей время инференса, использование памяти и пропускную способность системы.

2. Экономические метрики эффективности. Стоимость внедрения C_I охватывает разработку, интеграцию и тестирование. Операционные расходы C_O складываются из дополнительных вычислений, обусловленных ростом CO, и затрат на мониторинг. Снижение риска $\Delta\text{Risk}_{\text{val}} = \sum_i(\text{Risk}_i(f_{\text{base}}) - \text{Risk}_i(f_{\text{защищ}}))$ оценивается по Определению 1.4 Главы 1. Возврат инвестиций вычисляется как $\text{ROI} = (\Delta\text{Risk}_{\text{val}} - (C_I + C_O))/(C_I + C_O)$. К неколичественным выгодам B_S относятся повышение доверия пользователей, репутационные эффекты и соответствие регуляторным требованиям.

3. Интегральные метрики эффективности. Коэффициент компромисса устойчивость–полезность RUT характеризует соотношение улучшения устойчивости и потерь полезности (Раздел 4.2.4 Главы 4). Все компоненты RUT нормированы в [0,1] методом min-max: $x_{\text{norm}} = (x - x_{\min})/(x_{\max} - x_{\min})$, где границы определяются по валидационной выборке. Эффективность затрат на защиту определяется как отношение прироста устойчивости ΔR_{stab} к совокупным расходам $C_I + C_O$, а совокупная стоимость владения TCO включает все затраты на разработку, внедрение и эксплуатацию.

Лабораторного тестирования недостаточно для оценки готовности к production-среде. Пилотные проекты и прототипы Proof of Concept фиксируют реальные издержки: рост задержки ответа, стоимость API-запросов, энергозатраты. Одновременно контролируется, не деградировало ли базовое качество на целевых задачах. Акты внедрения и отчеты по пилотным проектам подтверждают практическую значимость разработанных методов.

Оценку требуется повторять регулярно, поскольку ландшафт угроз меняется и без пересмотра защитные механизмы устаревают.

6.2 Примеры практического применения

Практическое применение результатов работы реализовано в двух формах. Реально выполненные элементы внедрения подтверждают работоспособность методов. Типовые референтные сценарии применения предложенных методов в прикладных системах носят рекомендательный характер и иллюстрируют переносимость разработанных подходов на другие предметные области.

К фактически реализованным результатам внедрения относятся: использование разработанных методов и программных средств в учебном процессе факультета ВМК МГУ имени М. В. Ломоносова, а также применение фреймворка оценки уязвимостей, метода комитетов моделей и методологии многошаговых атак в производственной среде компании ВИАСАТ TEX для обеспечения безопасности модуля ранжирования рекомендаций на архитектуре LLM-as-a-Judge. В рамках внедрения в ВИАСАТ TEX развернут комплекс JudgeGuard для тестирования модуля ранжирования рекомендаций методами prompt injection, включая разработанный автором алгоритм ASA и атаки класса CUA. Комитет из трех моделей (GPT-4, Claude-3-Opus, Llama 3) голосует мажоритарно для вынесения финальных вердиктов ранжирования. Нагрузочное тестирование подтвердило приемлемость увеличения латентности: не более 15% по сравнению с одиночной моделью. По данным внутреннего тестирования ВИАСАТ TEX, ASR снизился на 42 процентных пункта. Результат получен на внутреннем наборе данных компании, отличающемся от экспериментальной выборки Главы 4. Дополнительно внедрены входные фильтры на основе Python-санитизатора для блокировки инъекций в пользовательских отзывах, поступающих в модуль оценки. Апробация комплекса TrojanArmor и модуля MCPSec на момент написания работы ограничена бенчмарками TDC2023 и MCPBench соответственно. Производственное внедрение этих средств, в отличие от JudgeGuard, не проводилось, поскольку обнаружение троянских закладок и защита агентских MCP-систем не входили в профиль рисков рассмотренной производственной площадки.

Разработанные методы применимы в разных предметных областях. Типовые сценарии и референтные архитектуры, приведенные далее, носят рекомендательный характер и описывают схемы внедрения.

6.2.1 Применение в системах обработки естественного языка

Примеры данного и следующего подразделов носят рекомендательный характер и описывают типовые референтные архитектуры, а не фактически развернутые системы. Реально выполненные внедрения перечислены в начале данного раздела.

Системы обработки естественного языка различаются по профилю угроз. Три типа таких систем и подходы к обеспечению их безопасности рассмотрены с учетом специфики каждого.

Чат-боты и виртуальные ассистенты

В качестве референтного сценария рассмотрена система поддержки клиентов финансовой организации. Пользователи задают вопросы о продуктах, тарифах и операциях по счетам через текстовый интерфейс. Профиль угроз включает инъекции запросов (LLM01 по OWASP, Угроза 1), обход ограничений через jailbreaking (Угроза 14), утечку конфиденциальных данных из контекста диалога (LLM06, Угроза 11) и целенаправленную манипуляцию поведением.

Центральный элемент защиты: контекстная изоляция [88], запрещающая модели интерпретировать пользовательские данные как инструкции. До передачи в модель входные фильтры, лексические и семантические на базе LLM, отсеивают опасные паттерны. Модельный уровень опирается на Safety Alignment через RLHF/RLAIF [36; 85], описанный в Разделе 3.3 Главы 3, и явные ограничения в системном промпте. PII удаляется до передачи в LLM. Модель дообучена через RLHF, чтобы не генерировать непроверенные финансовые рекомендации. Логирование диалогов обеспечивает выявление новых векторов атак.

Генерация контента, оценка и рекомендации

Выходы генеративных LLM-приложений и систем оценки/рекомендаций напрямую влияют на решения пользователей или других систем, что создает общую плоскость угроз. Вредоносный код (LLM02 по OWASP), дезинформация и манипуляция ранжированием через состязательные запросы образуют основные векторы атак. В системах LLM-as-a-Judge атаки типа CUA и JMA, исследованные в Главе 4, искажают вердикты напрямую. Рекомендательные системы уязвимы к утечке пользовательских предпочтений (LLM06 по OWASP) и усилению информационных пузырей.

Генерируемый контент проверяется внешними верификаторами [149; 150] и статическим анализом кода [151]. Ограниченное декодирование [152] сужает множество допустимых выходов. Робастное обучение (Раздел 3.3 Главы 3) снижает чувствительность модели к провокационным запросам. В системах ранжирования робастные эмбеддинги [153] совместно с детекторами аномалий образуют первый эшелон. Его дополняют алгоритмы, нечувствительные к выбросам [154], техники Fair ML [155] и дифференциальная приватность [156]. Против атак CUA и JMA на LLM-as-a-Judge эффективны комитеты из нескольких моделей и сравнительный формат оценки.

Два референтных сценария показывают, как различается расстановка акцентов. Платформа оценки студенческих эссе использует комитет из трех LLM: эссе проходят Perplexity Check и Instruction Filtering, после чего оцениваются в сравнительном формате относительно эталонных работ. Рекомендательная система новостного портала устроена иначе: ансамбль независимых моделей сочетается с фильтрами состязательных паттернов на входе и алгоритмами обеспечения разнообразия контента.

Выбор конкретных методов защиты определяется спецификой приложения: избыточная защита снижает функциональность, а недостаточная создает уязвимости.

6.2.2 Применение в критических системах

Критические системы предъявляют повышенные требования к надежности. Уязвимости к prompt injection и скрытым троянам, описанные в предыдущих главах, делают применение LLM в таких системах рискованным без специальных мер.

Медицинские системы поддержки принятия решений

Референтный сценарий: LLM анализирует рентгеновские снимки, опираясь на верифицированную базу знаний о патологиях через RAG [157]. Второй независимый алгоритм проверяет выводы до их представления врачу, а результаты сопровождаются оценкой уверенности и ссылками на источники. Архитектура противодействует основным рискам медицинских LLM-приложений: неверным диагнозам, утечке данных пациентов (PHI), отравлению обучающих данных [158] и демографической предвзятости.

Цена ошибки в медицинском домене выше, чем в большинстве других. Защита поэтому опирается на несколько принципов одновременно. Где возможно, применяются методы сертифицированной робастности с математическими гарантиями [159]. Врач сохраняет роль финального звена в принятии решений (Human-in-the-loop). Объяснимость выводов достигается методами Explainable AI [160]. Аудит на предвзятость [161] и защита данных по требованиям HIPAA дополняют технические меры.

Финансовые системы и управление инфраструктурой

Общий принцип для финансовых и инфраструктурных приложений LLM: модель изолируется от принятия автономных решений с необратимыми последствиями. Характер угроз и регуляторный контекст при этом различаются.

В финансах LLM-приложения подвержены генерации ошибочных прогнозов с прямыми убытками, манипуляции входными данными через новости и социальные сети, утечке инсайдерской информации. Ансамблирование разнородных моделей с диверсификацией источников, детекторы аномалий [162] и робастные методы прогнозирования временных рядов [163] составляют техническую основу. Регуляторный мониторинг ведется в соответствии с SOX, GDPR и отраслевыми нормативными актами. В качестве референтного сценария рассмотрена система алгоритмической торговли. Многоуровневые фильтры отсеивают недостоверные источники, решения о сделках принимаются на основе консенсуса нескольких моделей с подтверждением классическими количественными индикаторами, вся система работает в изолированной среде.

В управлении энергосетями, транспортом и промышленным оборудованием характер угроз иной: некорректные управляющие команды и целенаправленные атаки создают риски физической безопасности. LLM ограничивается ролью советника и архитектурно отделяется от прямого управления критическими компонентами. Рекомендации модели проходят формальную верификацию [164] и утверждаются оператором. Сегментация сети, многоуровневое резервирование, обнаружение вторжений и механизмы безопасного отключения при аномалиях образуют дополнительные рубежи защиты.

Во всех критических системах технические меры эффективны лишь в связке с организационными процессами, периодическим аудитом и соблюдением регуляторных норм.

6.3 Перспективы развития методов повышения устойчивости

Область безопасности LLM меняется быстро: за 2023–2025 годы опубликовано свыше 400 работ по атакам на языковые модели и более 200 по защитным механизмам. Новые техники атак появляются чаще, чем выходят обновления защит.

6.3.1 Тенденции и направления исследований

Теоретические основы и формальная верификация

Формальные модели не успевают за практикой атак: разрыв между экспериментальными результатами и теоретическим пониманием уязвимостей LLM сохраняется. Формализация робастности в дискретных пространствах высокой размерности остается нерешенной. Математические модели, адекватно описывающие пространство токенов и семантических преобразований, пока не разработаны. Доказательные гарантии для методов защиты, включая alignment-техники вроде RLHF, отсутствуют. Количественный анализ компромиссов robustness-utility, privacy-utility и fairness-robustness далек от завершения, а результаты по метрике RUT из Главы 4 лишь подтверждают существенность этих компромиссов.

Формальная верификация свойств LLM [164; 165] представляет отдельную задачу. SMT-решатели и абстрактная интерпретация пока применимы лишь к сетям малого размера. Прогресс в масштабировании абстрактной интерпретации, тем не менее, позволяет ожидать, что верификация отдельных свойств станет доступна и для моделей с миллиардами параметров.

От реактивной защиты к адаптивной

Существующие защиты создаются под известные классы атак и остаются реактивными. Опыт SaTML CTF и TDC2023 демонстрирует: атакующие систематически обходят фиксированные фильтры. Защиты нового поколения должны перестраиваться при встрече с zero-day атаками. Онлайн-обучение защитных модулей на потоке новых атак выглядит перспективным направлением. Самовосстанавливающиеся архитектуры развивают эту идею, автоматически закрывая обнаруженные уязвимости. Red teaming с адаптивными атаками типа ASA уже позволяет автоматизировать тестирование безопасности. Теоретико-игровое моделирование [65] дает инструмент оптимизации стратегий защиты.

Все перечисленные подходы, однако, предъявляют высокие требования к вычислительным ресурсам.

Адаптивность и интерпретируемость связаны напрямую. Без понимания причин уязвимости защита не способна обобщаться на новые атаки. Проблема «непреднамеренных триггеров» в троянских атаках (Глава 2) иллюстрирует дефицит интерпретируемости. Локализация уязвимых компонентов позволит определять конкретные слои и головы внимания, ответственные за backdoor-поведение [166]. Объяснимые механизмы защиты, сообщающие причину блокировки запроса, и визуализация атак через JailbreakLens [167] дополняют этот подход.

Мультимодальная устойчивость и новые поверхности атак

GPT-4V, Gemini 1.5 и LLaVA уже развернуты в продуктовых системах и принимают мультимодальный вход [168; 169]. Новые модальности открывают новые поверхности атак. Кросс-модальное воздействие позволяет, к примеру, через специально сконструированное изображение в документе изменить текстовый вывод модели. Защитные механизмы для мультимодальных систем как класс пока не сформированы. Разработка метрик робастности, учитывающих возмущения в каждой модальности по отдельности и в их комбинации, остается открытой задачей.

Контекст мультимодальной устойчивости расширяют два смежных направления. Федеративное обучение LLM [170] порождает угрозу отравления через агрегацию градиентов. Участник сети способен внедрить backdoor в глобальную модель, и протоколы вроде FedAvg к такому воздействию не устойчивы [171]. Дифференциальная приватность усугубляет положение: добавляемый шум может маскировать сигналы от backdoor-компонентов [172]. Иной характер угрозы несут квантовые вычислители. Алгоритм Гровера сокращает стойкость HMAC-SHA256 с 2^{256} до 2^{128} операций [173], что потребует адаптации криптографических протоколов, включая HMAC в AttestMCP из Главы 5. Вопрос о способности квантовых алгоритмов ускорить поиск состязательных примеров для нейросетей пока остается без теоретических оценок.

6.3.2 Прогнозы развития технологий

На основе текущих тенденций можно выделить несколько вероятных направлений развития.

Наиболее очевидным является эскалация автоматизированных атак и защит. LLM-агенты уже применяются для автоматического red teaming [44]. Адаптивные атаки типа ASA из Главы 3 демонстрируют, как эволюционный поиск обходит статические фильтры. Защиты в ответ будут использовать сами языковые модели для обнаружения угроз [174]. Платформы типа Gandalf [175] уже реализуют LLM-генерацию атакующих сценариев. Результаты SaTML CTF подтверждают нарастание такой «гонки».

Параллельно требования к робастности сместятся на ранние этапы разработки LLM по принципу «security by design», а постфактумное добавление фильтров уступит место фреймворкам с гарантиями безопасности на этапе обучения. HELM [14], JailbreakBench [176] и MT-Bench формируют базу для стандартизированной сравнительной оценки. Соревнования TDC2023 и Kaggle ускоряют этот процесс. Для критических приложений вероятна сертификация моделей по аналогии с Common Criteria. В этом контексте RLHF/RLAIF [36; 85] и Adversarial Training из Главы 3 решают смежные задачи разными средствами, и их интеграция даст модели, одновременно выровненные с человеческими предпочтениями и устойчивые к манипуляциям.

Отдельного внимания заслуживают отраслевая специализация и формальная верификация. Универсальные методы уступят специализированным решениям: архитектуры Mixture-of-Experts [177] потребуют защиты маршрутизатора экспертов, финансовые [178] и медицинские [161] приложения диктуют собственные требования к верификации, а защита LLM-as-a-Judge выделится в самостоятельное направление. Параллельно будет нарастать потребность в математических гарантиях безопасности [165]. Масштабирование формальных методов на модели с миллиардами параметров пока не решено, однако прогресс в абстрактной интерпретации нейросетей дает основания для оптимизма.

Общая тенденция: безопасность и устойчивость постепенно встраиваются в основной цикл разработки LLM, а реактивные меры уступают место проактивным.

Выводы по главе

Перенос результатов в эксплуатацию выявил ограничение, не видимое на бенчмарках: эффект защиты зависит от данных конкретного развертывания. Внедрение комитета из трех моделей в ВИАСАТ TEX дало снижение ASR на 42 п.п. на внутреннем наборе данных, отличном от экспериментальной выборки Главы 4, при росте латентности не более 15%. Это подтверждает переносимость метода, но показывает, что числовые показатели защиты не переносятся между доменами без повторной калибровки.

Оценка готовности к production требует метрик трех групп: технических, экономических и интегральной RUT. Интегральная метрика связывает разнородные критерии и позволяет пересчитывать оптимальную конфигурацию защиты при изменении ландшафта угроз, тогда как технические и экономические метрики по отдельности такого пересчета не дают.

Экономика комитета линейна по числу моделей, но параллельный опрос участников делает реальный прирост задержки сублинейным. Для дальнейшего снижения расходов перспективна каскадная схема: одиночная модель обрабатывает запрос, а комитет активируется только при низком уровне согласия, что выявляется по уверенности c из Раздела 3.4.1. Количественная оценка экономии остается открытой задачей.

Глава 7. Заключение

Диссертация посвящена разработке методов оценки и повышения устойчивости больших языковых моделей к вариациям входных последовательностей. Получены следующие результаты.

1. Анализ данных Trojan Detection Challenge 2023 обнаружил асимметрию между генерацией и обнаружением троянских триггеров. Суррогатные триггеры активируют закладку почти гарантированно: REASR $\approx 0{,}99$. Восстановить исходный триггер не удается: BLEU-подобная суррогатная метрика Recall для лучшего метода GCG не превысила $\approx 0{,}17$. Из этого разрыва следует приоритет превентивных мер: аудита данных обучения и контроля цепочки поставок. Результат относится к Задачам 1–2.
2. Метрика генеративной устойчивости $R_{stab}(f)$ учитывает дискретность пространства токенов, фиксируя пошаговое расхождение выходных распределений при малых возмущениях входа. Для локализованных атак доказано $V(h) \leqslant 1 - R_{class}(h)$. Для нелокализованных воздействий предложено калибруемое параметрическое семейство $V(h) \approx g(1 - R_{class}(h))$ со статусом эмпирической гипотезы. Результат относится к Задаче 2.
3. Уязвимость систем LLM-as-a-Judge к инъекциям запроса исследована в рамках разработанного фреймворка, разделяющего атаки на контентные и инструктивные. Алгоритм адаптивной генерации ASA достигает ASR до 73,8% на моделях Gemma, Llama, GPT-4 и Claude-3-Opus. Уязвимость зависит от архитектурного семейства и процедуры выравнивания, а не только от масштаба модели. Атаки обладают высокой переносимостью между открытыми архитектурами. Результат относится к Задаче 3.
4. На материале SaTML CTF 2024 систематизированы многошаговые атаки на LLM с трехуровневой защитой. Классифицированы четыре вектора обхода: код-ориентированные атаки, ASCII-кодирование, разделение слов и контекстное перенаправление. Первые три нейтрализуются нормализацией входных данных, четвертый требует контекстной изоляции системных инструкций. В качестве защитного механизма разработаны комитеты из 5–7 гетерогенных моделей. Для наиболее уязвимой мо-

дели Gemma-3-4B снижение ASR составило 47–55 п.п. Теоретическим основанием служит теорема Кондорсе о жюри: эффективность комитета обеспечивается тем, что большинство участников обладают базовой точностью $r > 0{,}5$. Для Gemma-3-4B с $r \approx 0{,}26$ снижение ASR достигается за счет гетерогенности состава комитета, декоррелирующей ошибки моделей разных архитектур. Метрика RUT для ансамбля из 7 моделей равна $\approx 5{,}5$. Результат относится к Задаче 4.

5. Обнаружены архитектурные уязвимости протокола MCP, открывающие путь к атакам класса *prompt injection to tool hijacking*. Для противодействия им предложены два алгоритма: AttestMCP ограничивает вызовы инструментов сессионной таблицей полномочий с HMAC-контролем целостности пакетов, Commit Boundary изолирует исполнение кода агента. Оценка на тестовом наборе MCPBench из 847 сценариев зафиксировала снижение средней ASR на 41,3 п.п., с 53,7% до 12,4%. Результат относится к Задаче 5.
6. Реализованы программные комплексы JudgeGuard и TrojanArmor, охватывающие полный цикл тестирования безопасности LLM: от генерации атакующих воздействий до вычисления метрик устойчивости. Модуль MCPSec дополняет комплекс средствами анализа агентских LLM-систем. Результат относится к Задаче 5.

Практическая значимость результатов подтверждена внедрением комплекса JudgeGuard и метода защиты на основе комитетов моделей в производственную среду компании ВИАСАТ TEX, а также использованием разработанных методов и программных средств в учебном процессе факультета ВМК МГУ.

Предложенные метрики, алгоритмы атак и защит, а также программные комплексы применимы к существующим и проектируемым LLM-системам. Перспективные направления развития работы включают распространение аппарата на мультимодальные модели, адаптацию алгоритмов для обнаружения атак в реальном времени и построение верифицируемых гарантий безопасности.

Список сокращений и условных обозначений

Аббревиатуры

AGM	Модуль Генерации Атакующих Воздействий (Attack Generator Module)
AI	Искусственный Интеллект (Artificial Intelligence)
AI RMF	Система Управления Рисками ИИ (NIST AI Risk Management Framework)
API	Интерфейс Прикладного Программирования (Application Programming Interface)
AR	Устойчивость к Атакам (Attack Resistance)
ASA	Адаптивная Атака на Основе Поиска (Adaptive Search-Based Attack)
ASR	Коэффициент Успешности Атаки (Attack Success Rate)
AT	Состязательное Обучение (Adversarial Training)
BI	Базовая Инъекция (Basic Injection)
BLEU	Оценка Качества Двуязычного Перевода (Bilingual Evaluation Understudy)
CM	Контекстуальное Перенаправление (Contextual Misdirection)
CO	Вычислительная Нагрузка (Computational Overhead)
CPU	Центральный процессор (Central Processing Unit)
CUA	Атака Сравнительного Подрыва (Comparative Undermining Attack)
CVSS	Общая Система Оценки Уязвимостей (Common Vulnerability Scoring System)
CTF	Соревнование по Захвату Флага (Capture The Flag)
CWB	Комплексная Словесная Бомбардировка (Complex Word Bombardment)
DAG	Направленный Ациклический Граф (Directed Acyclic Graph)
DoS	Отказ в обслуживании (Denial of Service)
DP	Дифференциальная Приватность (Differential Privacy)
EE	Модуль Вычисления Метрик (Evaluation Engine)

FFN	Полносвязная Нейронная Сеть (Feed-Forward Network)
GCG	Жадный Координатный Градиент (Greedy Coordinate Gradient)
GPU	Графический процессор (Graphics Processing Unit)
HITL	Человек в Контуре (Human-In-The-Loop)
HMAC	Код Аутентификации Сообщений на Основе Хеша (Hash-based Message Authentication Code)
JS(P,Q)	Дивергенция Йенсена–Шеннона (Jensen–Shannon divergence, JSD)
JMA	Атака Манипуляции Обоснованием (Justification Manipulation Attack)
KD	Дистилляция Знаний (Knowledge Distillation)
LLM	Большая Языковая Модель (Large Language Model)
MCP	Протокол Контекста Модели (Model Context Protocol)
MCPSec	Безопасность Протокола Контекста Модели (Model Context Protocol Security)
MLD	Многоуровневая Защита (Multi-level Defense)
MoE	Смесь Экспертов (Mixture of Experts)
MRAG	Маршрутизатор Моделей / API-шлюз (Model Router / API Gateway)
NIST	Национальный Институт Стандартов и Технологий (США) (National Institute of Standards and Technology)
NLP	Обработка Естественного Языка (Natural Language Processing)
OWASP	Открытый Проект по Обеспечению Безопасности Веб-Приложений (Open Web Application Security Project)
PHI	Защищаемая Медицинская Информация (Protected Health Information)
PII	Персонально Идентифицируемая Информация (Personally Identifiable Information)
PPL	Перплексия (Perplexity)
RAG	Генерация, Дополненная Поиском (Retrieval-Augmented Generation)
RAM	Робастный Механизм Внимания (Robust Attention Mechanism)

REASR	Коэффициент Успешности Атаки с Обратной Разработкой (Reverse-Engineered Attack Success Rate)
ROUGE	Метрика Оценки на Основе Перекрытия (Recall-Oriented Understudy for Gisting Evaluation)
RLAIF	Обучение с Подкреплением на Основе Обратной Связи от ИИ (Reinforcement Learning from AI Feedback)
RLHF	Обучение с Подкреплением на Основе Обратной Связи от Человека (Reinforcement Learning from Human Feedback)
ROI	Возврат Инвестиций (Return on Investment)
RUT	Компромисс Устойчивость-Полезность (Robustness-Utility Trade-off)
SCC	Компонента Сильной Связности (Strongly Connected Component)
SL	Уровень Хранения Данных (Storage Layer)
SQL	Язык Структурированных Запросов (Structured Query Language)
TCO	Совокупная Стоимость Владения (Total Cost of Ownership)
TDC2023	Соревнование по Обнаружению Троянов 2023 (Trojan Detection Challenge 2023)
TEC	Коэффициент Эффективности Компромисса (Trade-off Efficacy Coefficient)
TSR	Коэффициент Успешности Переноса (Transfer Success Rate)
VAT	Виртуальное Состязательное Обучение (Virtual Adversarial Training)
XSS	Межсайтовый скриптинг (Cross-Site Scripting)

Условные обозначения

$\mathcal{V}$	Конечный словарь токенов
$\mathcal{X} = \mathcal{V}^*$	Пространство всех конечных последовательностей токенов (входных текстов)
$\mathcal{Y}$	Пространство возможных выходов модели (выходных текстов)

$\mathcal{P}(\mathcal{Y})$	Множество вероятностных распределений на пространстве $\mathcal{Y}$
f, f_θ	Большая языковая модель, параметризованная параметрами θ
θ, θ	Параметры модели
$x \in \mathcal{X}$	Входная последовательность токенов
$x' \in \mathcal{X}$	Возмущенная входная последовательность токенов
$y \in \mathcal{Y}$	Выходная последовательность токенов
$p(y\|x)$	Вероятностное распределение на выходах y при заданном входе x
$\|x\|$	Длина последовательности x
$d(x, x')$	Метрика (расстояние) между последовательностями x и x'
d_H	Расстояние Хэмминга
d_L	Расстояние Левенштейна
d_S	Семантическое расстояние
$\mathcal{B}_\varepsilon(x)$	ε-окрестность последовательности x
ε	Бюджет возмущений
$R_{stab}(f)$	Метрика устойчивости (стабильности) генеративной модели f
$R_{class}(h)$	Метрика устойчивости классификационной модели ($h = \mathrm{Dec} \circ f$)
$D(p_1, p_2)$	Обобщенная мера расхождения между распределениями p_1 и p_2 (например, JS); для несимметричных мер используется специальная запись, например $D_{KL}(P \parallel Q)$
$D_{KL}(P\|Q)$	Дивергенция Кульбака-Лейблера между распределениями P и Q
JS(P,Q)	Дивергенция Йенсена–Шеннона между распределениями P и Q
C_{max}	Нормализационная константа для меры расхождения D
$V_{\text{local}}(f)$	Метрика локальной уязвимости модели f
$V(f)$	Метрика глобальной уязвимости модели f
$\mathcal{Y}_{\text{bad}}$	Множество нежелательных или запрещенных выходов

$\mathcal{D}$, $P_{\mathcal{X}}$	Распределение входных данных
$\mathbb{E}[\cdot]$	Математическое ожидание
$\mathbb{I}\{\cdot\}$	Индикаторная функция
sup	Супремум (точная верхняя грань)
$\mathcal{L}$	Функция потерь
$\mathcal{A}^{\text{type}}$	Множество типов атак (Глава 1); $\mathcal{A}$ – множество атакующих преобразований (Глава 2)
$\mathcal{V}$	Множество уязвимостей
$\mathcal{I}_{AV}$	Отношение «атака использует уязвимость»
$\mathcal{C}_{VV}$	Отношение зависимости между уязвимостями
$\text{Risk}(a, f)$	Риск атаки a на модель f
$\text{Impact}(\cdot)$	Функция оценки воздействия или ущерба
$\text{AdvRobustness}(f)$	Состязательная устойчивость модели f

Словарь терминов

Большая языковая модель (LLM) Класс моделей машинного обучения, основанных на глубоких нейронных сетях, обученных на больших объемах текстовых данных для понимания, генерации и обработки естественного языка.

Токен (Token) Минимальная единица текста, с которой оперирует языковая модель, например слово, часть слова или символ. Процесс преобразования текста в последовательность токенов называется токенизацией.

Эмбеддинг (Embedding) Векторное представление токена, фразы или документа в многомерном пространстве, отражающее семантические свойства.

Устойчивость (Robustness) Способность модели или композиционной системы на ее основе сохранять корректность поведения и приемлемые характеристики качества при малых возмущениях входных данных, состязательных модификациях запросов и иных воздействиях, рассматриваемых в принятой модели угроз. В данной работе является основным термином. Термин «робастность» используется как синоним при отсылке к англоязычной литературе (adversarial robustness). Метрика $R_{stab}(f)$ (от англ. stability) количественно оценивает устойчивость через расхождение выходных распределений. Индекс «stab» отражает этимологию метрики и не подразумевает отдельного понятия «стабильность».

Уязвимость (Vulnerability) Свойство системы или модели, которое может быть использовано атакующим для нарушения ее нормального функционирования, компрометации данных или обхода механизмов защиты.

Атака (Attack) Целенаправленное воздействие на систему или модель с целью вызвать нежелательное поведение, получить несанкционированный доступ, извлечь информацию или нарушить работоспособность.

Состязательная атака (Adversarial Attack) Тип атаки, заключающийся в создании малых, часто незаметных для человека, возмущений во входных данных, которые приводят к ошибочному или нежелательному поведению модели.

Состязательный пример (Adversarial Example) Входные данные, модифицированные в результате состязательной атаки.

Jailbreaking Тип атаки, направленный на обход встроенных в LLM механизмов безопасности (safety alignment) с целью заставить модель генерировать запрещенный, вредоносный или неэтичный контент.

Инъекция запроса (*Prompt Injection*) Тип атаки, при котором в легитимный запрос пользователя внедряются скрытые или замаскированные инструкции, которые переопределяют исходные системные указания модели или ее механизмы безопасности. В тексте работы используется русскоязычный термин «инъекция запроса». Англоязычный эквивалент *prompt injection* приводится при первом упоминании и в контексте международных классификаций.

Отравление данных (Data Poisoning) Атака, при которой злоумышленник внедряет в обучающий набор данных небольшое количество специально сформированных («отравленных») примеров с целью нарушить процесс обучения, снизить качество модели, создать троянские закладки или усилить предвзятость.

Троянская закладка (Бэкдор, *Backdoor*) Скрытая функциональность, внедренная в модель, как правило через отравление данных. Закладка активируется специальным триггером во входных данных и вызывает предопределенное атакующим поведение. В тексте работы основным является термин «троянская закладка», а «бэкдор» используется как краткий синоним.

Выравнивание (Alignment) Процесс обучения или дообучения LLM с целью приведения ее поведения в соответствие с человеческими ценностями, этическими нормами и инструкциями, а также для предотвращения генерации вредоносного, предвзятого или нежелательного контента.

Дифференциальная приватность (Differential Privacy) Формальное определение приватности, гарантирующее, что результат анализа данных практически не изменится, если из набора данных удалить или изменить одну любую запись. Используется для защиты конфиденциальности индивидуальных данных в обучающих наборах.

Расстояние Левенштейна (Levenshtein Distance) Метрика, измеряющая минимальное количество односимвольных операций (вставка, удаление, замена), необходимых для преобразования одной строки (последовательности токенов) в другую.

Дивергенция Кульбака-Лейблера (KL-Divergence) Мера расхождения между двумя вероятностными распределениями.

Комитет моделей (Ensemble of models / Model Committee) Подход к повышению надежности и точности предсказаний, при котором используется несколько независимо обученных моделей, чьи выходы объединяются, например через голосование, усреднение или взвешенную комбинацию, для получения финального результата. В контексте безопасности LLM комитет моделей может использоваться для обнаружения аномальных или потенциально вредоносных запросов.

LLM-as-a-Judge Архитектура или методология автоматической оценки, при которой большая языковая модель выступает в роли оценщика и выносит дискретный или ранговый вердикт относительно качества, безопасности, соответствия инструкции или предпочтительности ответов другой модели либо нескольких альтернативных ответов.

Многоуровневая защита (Multi-level Defense) Стратегия обеспечения безопасности, основанная на применении нескольких независимых уровней защитных механизмов. В контексте LLM включает комбинацию различных методов: фильтрацию входных данных, мониторинг поведения модели, постобработку выходных данных, выравнивание модели и другие техники, работающие совместно для минимизации рисков.

Список литературы

1. *Maloyan, N.* Investigating the Vulnerability of LLM-as-a-Judge Architectures to Prompt-Injection Attacks [Текст] / N. Maloyan, B. Ashinov, D. Namiot // International Journal of Open Information Technologies. — 2025. — Т. 13, № 9. — URL: http://injoit.ru/index.php/j1/article/view/2261.
2. *Maloyan, N.* Prompt Injection Attacks on Agentic Coding Assistants: A Systematic Analysis of Vulnerabilities in Skills, Tools, and Protocol Ecosystems [Текст] / N. Maloyan, D. Namiot // Современные информационные технологии и ИТ-образование. — 2025. — Т. 21, № 3. — URL: http://sitito.cs.msu.ru/index.php/SITITO/article/view/1280.
3. *Maloyan, N.* Breaking the Protocol: Security Analysis of the Model Context Protocol Specification and Prompt Injection Vulnerabilities in Tool-Integrated LLM Agents [Текст] / N. Maloyan, D. Namiot // International Journal of Open Information Technologies. — 2026. — Т. 14, № 2. — URL: http://injoit.org/index.php/j1/article/view/2423.
4. *Maloyan, N.* Sleeper Channels and Provenance Gates: Persistent Prompt Injection in Always-on Autonomous AI Agents [Текст] / N. Maloyan, D. Namiot // International Journal of Open Information Technologies. — 2026. — Т. 14, № 6. — URL: https://injoit.org/index.php/j1/article/view/2621.
5. *Малоян, Н. Г.* JudgeGuard. Программа для экспериментального исследования уязвимостей и тестирования методов защиты систем автоматической оценки на базе больших языковых моделей [Текст] / Н. Г. Малоян. — 2025. — Рос. Федерация. Свидетельство о государственной регистрации программы для ЭВМ № 2025687702.
6. *Малоян, Н. Г.* TrojanArmor. Программа для экспериментального исследования троянских атак в нейронных сетях и методов защиты от них [Текст] / Н. Г. Малоян. — 2025. — Рос. Федерация. Свидетельство о государственной регистрации программы для ЭВМ № 2025684247.
7. *Khomsky, D.* Prompt Injection Attacks in Defended Systems [Текст] / D. Khomsky, N. Maloyan, B. Nutfullin // Distributed Computer and

Communication Networks. — Springer Nature Switzerland, 2025. — С. 404—416. — URL: http://dx.doi.org/10.1007/978-3-031-80853-1_30.

8. *Maloyan, N.* DIALOG-22 RuATD Generated Text Detection [Текст] / N. Maloyan, B. Nutfullin, E. Ilyshin // Computational Linguistics and Intellectual Technologies. — RSUH, 06.2022. — С. 394—401. — URL: http://dx.doi.org/10.28995/2075-7182-2022-21-394-401.

9. *Maloyan, N.* Adversarial Attacks on LLM-as-a-Judge Systems: Insights from Prompt Injections [Текст] / N. Maloyan, D. Namiot. — 2025. — arXiv: 2504.18333 [cs.CR]. — URL: https://arxiv.org/abs/2504.18333.

10. Attention is All you Need [Текст] / A. Vaswani [и др.] //. — 06.2017. — URL: https://www.semanticscholar.org/paper/Attention-is-All-you-Need-Vaswani-Shazeer/204e3073870fae3d05bcbc2f6a8e263d9b72e776.

11. Language Models are Few-Shot Learners [Текст] / T. B. Brown [и др.] // Advances in Neural Information Processing Systems. Т. 33 / под ред. H. Larochelle [и др.]. — Curran Associates, Inc., 2020. — С. 1877—1901. — URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

12. Intriguing properties of neural networks [Текст] / C. Szegedy [и др.]. — 2013.

13. *Goodfellow, I. J.* Explaining and harnessing adversarial examples [Текст] / I. J. Goodfellow, J. Shlens, C. Szegedy // arXiv preprint arXiv:1412.6572. — 2014. — URL: https://arxiv.org/abs/1412.6572.

14. Holistic Evaluation of Language Models [Текст] / P. Liang [и др.]. — 10.2023. — URL: http://arxiv.org/abs/2211.09110 (дата обр. 25.01.2025).

15. *Jia, R.* Adversarial Examples for Evaluating Reading Comprehension Systems [Текст] / R. Jia, P. Liang // Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. — Copenhagen, Denmark : Association for Computational Linguistics, 09.2017. — С. 1868—1877. — URL: https://aclanthology.org/D17-1181.

16. *Levenshtein, V. I.* Binary codes capable of correcting deletions, insertions, and reversals [Текст] / V. I. Levenshtein // Soviet Physics Doklady. — 1966. — Т. 10, № 8. — С. 707—710.

17. BERT-ATTACK: Adversarial Attack Against BERT Using BERT [Текст] / L. Li [и др.] // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) / под ред. B. Webber [и др.]. — Online : Association for Computational Linguistics, 11.2020. — С. 6193—6202. — URL: https://aclanthology.org/2020.emnlp-main.500/ (дата обр. 30.03.2025).

18. Extracting Training Data from Large Language Models [Текст] / N. Carlini [и др.] // Proceedings of the 30th USENIX Security Symposium. — 2021. — С. 2633—2650. — URL: https://arxiv.org/abs/2012.07805.

19. *Perez, F.* Ignore Previous Prompt: Attack Techniques For Language Models [Текст] / F. Perez, I. Ribeiro. — 11/2022. — URL: http://arxiv.org/abs/2211.09527 (visited on 06/11/2024).

20. *Wei, A.* Jailbroken: How Does LLM Safety Training Fail? [Текст] / A. Wei, N. Haghtalab, J. Steinhardt // Advances in Neural Information Processing Systems (NeurIPS). — 2023. — URL: https://arxiv.org/abs/2307.02483.

21. Universal and Transferable Adversarial Attacks on Aligned Language Models [Текст] / A. Zou [и др.] // arXiv preprint arXiv:2307.15043. — 2023. — URL: https://arxiv.org/abs/2307.15043.

22. Universal Adversarial Triggers for Attacking and Analyzing NLP [Текст] / E. Wallace [и др.] // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). — Hong Kong, China : Association for Computational Linguistics, 11.2019. — С. 2153—2162. — URL: https://aclanthology.org/D19-1221 (дата обр. 09.10.2023).

23. Universal and Transferable Adversarial Attacks on Aligned Language Models [Текст] / A. Zou [и др.] // arXiv preprint arXiv:2307.15043. — 2023. — URL: https://arxiv.org/abs/2307.15043.

24. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP [Текст] / J. X. Morris [и др.]. — 10.2020. — URL: http://arxiv.org/abs/2005.05909 (дата обр. 30.03.2024).

25. Prompt Injection attack against LLM-integrated Applications [Текст] / K. Liu [и др.] // arXiv preprint arXiv:2306.05499. — 2023. — Июнь. — URL: http://arxiv.org/abs/2306.05499.

26. Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection [Текст] / K. Greshake [и др.] // Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec'23). — 2023. — С. 209—220.

27. *Khachaturov, D.* Adversarial Suffix Filtering: a Defense Pipeline for LLMs [Текст] / D. Khachaturov, R. Mullins // arXiv preprint arXiv:2505.09602. — 2025.

28. Jailbreak Attacks and Defenses Against Large Language Models: A Survey [Текст] / S. Yi [и др.] // arXiv:2407.04295v2 [cs.CR]. — 2024. — URL: https://arxiv.org/abs/2407.04295v2.

29. Characterizing and Evaluating the Reliability of LLMs against Jailbreak Attacks [Текст] / K. Chen [и др.]. — 2024. — Авг.

30. Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study [Текст] / Y. Liu [и др.]. — 03.2024. — URL: http://arxiv.org/abs/2305.13860 (дата обр. 20.04.2024).

31. *Gu, T.* BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain [Текст] / T. Gu, B. Dolan-Gavitt, S. Garg // arXiv preprint arXiv:1708.06733. — 2017. — URL: https://arxiv.org/abs/1708.06733.

32. Targeted backdoor attacks on deep learning systems using data poisoning [Текст] / X. Chen [и др.] // arXiv preprint arXiv:1712.05526. — 2017.

33. *Kurita, K.* Weight poisoning attacks on pre-trained models [Текст] / K. Kurita, P. Michel, G. Neubig // arXiv preprint arXiv:2004.00784. — 2020.

34. *Anthropic.* Model Context Protocol Specification [Текст] / Anthropic. — 2024. — Accessed: 2025-01-15. https://spec.modelcontextprotocol.io.

35. Towards deep learning models resistant to adversarial attacks [Текст] / A. Madry [и др.] // arXiv preprint arXiv:1706.06083. — 2017.

36. Training language models to follow instructions with human feedback [Текст] / L. Ouyang [и др.]. — 03.2022. — URL: http://arxiv.org/abs/2203.02155 (дата обр. 03.02.2024).

37. *Xu, Z.* Bag of Tricks: Benchmarking of Jailbreak Attacks on LLMs [Текст] / Z. Xu, F. Liu, H. Liu. — 2024. — arXiv: 2406.09324 [cs.CR]. — URL: https://arxiv.org/abs/2406.09324.

38. Adversarial GLUE: A Multi-Task Benchmark for Robustness Evaluation of Language Models [Текст] / B. Wang [и др.]. — 01.2022. — URL: http://arxiv.org/abs/2111.02840 (дата обр. 30.03.2024).

39. TF-Attack: Transfer Learning by Adversarial Fine-tuning for Attacking Large Language Models [Текст] / Z. Li [и др.] // arXiv preprint arXiv:2404.09404. — 2024. — URL: https://arxiv.org/abs/2404.09404.

40. JailbreakZoo: Survey, Landscapes, and Horizons in Jailbreaking Large Language and Vision-Language Models [Текст] / H. Jin [и др.] // arXiv. — 2024. — Июль. — Т. 2407, № 01599v2. — URL: https://chonghan-chen.com/llm-jailbreak-zoo-survey/.

41. *Andriushchenko, M.* JAILBREAKING LEADING SAFETY-ALIGNED LLMs WITH SIMPLE ADAPTIVE ATTACKS [Текст] / M. Andriushchenko, F. Croce, N. Flammarion // arXiv. — 2024. — Окт. — С. 1—14. — URL: https://arxiv.org/abs/2404.02151v3.

42. JAILJUDGE: A Comprehensive Jailbreak Judge Benchmark with Multi-Agent Enhanced Explanation Evaluation Framework [Текст] / F. Liu [и др.]. — 10.2024. — URL: https://arxiv.org/abs/2410.12855v2.

43. Empirical data drift detection experiments on real-world medical imaging data [Текст] / A. Kore [и др.] // Nature Communications. — 2024. — Февр. — Т. 15, № 1887. — URL: https://www.nature.com/articles/s41467-024-46142-w.

44. RedAgent: Red Teaming Large Language Models with Context-aware Autonomous Language Agent [Текст] / H. Xu [и др.] // arXiv preprint arXiv:2407.16667v1. — 07.2024. — URL: https://arxiv.org/abs/2407.16667v1.

45. *OWASP Foundation.* OWASP Top 10 for Large Language Model Applications [Текст] / OWASP Foundation. — 2023. — URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/.

46. *National Institute of Standards and Technology.* Artificial Intelligence Risk Management Framework (AI RMF 1.0) [Текст] : тех. отч. / National Institute of Standards and Technology ; U.S. Department of Commerce. — 01.2023. — NIST AI 100—1. — URL: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf.

47. Privacy Leakage on DNNs: A Survey of Model Inversion Attacks and Defenses [Текст] / H. Fang [и др.] // arXiv preprint arXiv:2402.04013. — 2024. — Февр. — URL: https://arxiv.org/abs/2402.04013.

48. A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly [Текст] / Y. Yifan [и др.] // High-Confidence Computing. — 2024. — Июнь. — Т. 4, № 2. — С. 100211. — URL: https://www.sciencedirect.com/science/article/pii/S266729522400014X.

49. Machine Learning Security against Data Poisoning: Are We There Yet? [Текст] / A. E. Cinà [и др.] // arXiv preprint arXiv:2204.05986. — 2024. — Март. — URL: https://arxiv.org/html/2204.05986v3.

50. Poison frogs! targeted clean-label poisoning attacks on neural networks [Текст] / A. Shafahi [и др.] // Advances in neural information processing systems. — 2018. — Т. 31.

51. A Comprehensive Overview of Backdoor Attacks in Large Language Models [Текст] / Y. Li [и др.] // arXiv preprint arXiv:2308.14367. — 2023. — URL: https://arxiv.org/abs/2308.14367.

52. *Veale, M.* Algorithms that remember: model inversion attacks and data protection law [Текст] / M. Veale, R. Binns, L. Edwards // Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. — 2018. — Окт. — Т. 376, № 2133. — С. 20180083. — URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC6191664/.

53. *Fredrikson, M.* Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures [Текст] / M. Fredrikson, S. Jha, T. Ristenpart // Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15). — 2015. — С. 1322—1333.

54. Membership Inference Attacks Against Machine Learning Models [Текст] / R. Shokri [и др.] // Proceedings of the IEEE Symposium on Security and Privacy (S&P). — 2017. — С. 3—18.

55. OverThink: Slowdown Attacks on Reasoning LLMs [Текст] / W. Zhang [и др.] // arXiv preprint arXiv:2502.02542. — 2024. — URL: https://arxiv.org/abs/2502.02542.

56. Model Evaluation for Extreme Risks [Текст] / R. Krishna [и др.] // arXiv preprint arXiv:2305.15324. — 2023. — URL: https://arxiv.org/abs/2305.15324.

57. Stealing Machine Learning Models via Prediction APIs [Текст] / F. Tramèr [и др.] // Proceedings of the 25th USENIX Security Symposium (USENIX Security '16). — 2016. — С. 601—618. — URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer.

58. *OWASP Foundation*. OWASP LLM02: Insecure Output Handling [Текст] / OWASP Foundation. — 2023. — URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/llm02/ ; Accessed: 2024-05-15.

59. Probing Privacy Leakage in Large Language Models [Текст] / X. Wang [и др.] // Proceedings of Neural Information Processing Systems (NeurIPS). — 2023. — С. 24631—24644. — URL: https://arxiv.org/abs/2307.01881.

60. Bias and Fairness in Large Language Models: A Survey [Текст] / I. O. Gallegos [и др.] // Computational Linguistics. — 2024. — Т. 50, № 3. — URL: https://arxiv.org/abs/2309.00770.

61. Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints [Текст] / J. Zhao [и др.] // Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP). — 2017. — С. 2979—2989.

62. S3C2: Secure Supply Chain Coding with Large Language Models [Текст] / M. R. I. Rahman [и др.] // arXiv preprint arXiv:2401.17125. — 2024. — URL: https://arxiv.org/abs/2401.17125.

63. *Joint Task Force*. Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations [Текст] : тех. отч. / Joint Task Force ; National Institute of Standards ; Technology. — 05.2022. — NIST Special Publication 800—161, Revision 1. — URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-161r1.pdf.

64. Systematic Evaluation of Neural Text Generation on 11 Languages and 16 Genres [Текст] / A. Belz [и др.] // Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics. — 2021. — С. 213—227. — URL: https://arxiv.org/abs/2109.06379.

65. Red Teaming Game: A Game-Theoretic Framework for Red Teaming Language Models [Текст] / C. Ma [и др.] // arXiv preprint arXiv:2310.00322. — 2023. — URL: https://arxiv.org/abs/2310.00322.

66. *Li, Y.* Large Language Models can be Guided to Evade AI-Generated Text Detection [Текст] / Y. Li, Y. Jiang, X. Ren // arXiv preprint arXiv:2305.10847. — 2023. — URL: https://arxiv.org/abs/2305.10847.

67. Practical Black-Box Attacks against Machine Learning [Текст] / N. Papernot [и др.] // Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIACCS). — 2017. — С. 506—519.

68. PLeak: Prompt Leaking Attacks against Large Language Model Applications [Текст] / B. Hui [и др.]. — 05.2024. — URL: http://arxiv.org/abs/2405.06823 (дата обр. 11.06.2024).

69. *Joint Task Force.* Security and Privacy Controls for Information Systems and Organizations [Текст] : тех. отч. / Joint Task Force ; National Institute of Standards ; Technology. — 09.2020. — NIST Special Publication 800—53, Revision 5. — URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf.

70. Can LLMs Keep a Secret? Testing Privacy Implications of Language Models via Contextual Integrity Theory [Текст] / N. Mireshghallah [et al.]. — 10/2023. — URL: http://arxiv.org/abs/2310.17884 (visited on 06/11/2024).

71. *Dwork, C.* The Algorithmic Foundations of Differential Privacy [Текст] / C. Dwork, A. Roth // Foundations and Trends in Theoretical Computer Science. — 2014. — Т. 9, № 3/4. — С. 211—407. — URL: https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf.

72. Deep Learning with Differential Privacy [Текст] / M. Abadi [и др.] // Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. — 2016. — С. 308—318. — URL: https://arxiv.org/abs/1607.00133.

73. Defending Against Alignment-Breaking Attacks via Robustly Aligned LLM [Текст] / B. Cao [и др.]. — 12.2023. — URL: http://arxiv.org/abs/2309.14348 (дата обр. 14.05.2024).

74. SmoothLLM: Defending Large Language Models Against Jailbreaking Attacks [Текст] / A. Robey [и др.]. — 11.2023. — URL: http://arxiv.org/abs/2310.03684 (дата обр. 14.05.2024).

75. *Spirling, A.* Replication for Language Models: Problems, Principles, and Best Practices [Текст] / A. Spirling, C. Barrie, J. Palmer // Working Paper. — 2023. — URL: https://arthurspirling.org/documents/BarriePalmerSpirling_TrustMeBro.pdf.

76. LLM Stability: A detailed analysis with some surprises [Текст] / T. Chen [и др.] // arXiv preprint arXiv:2408.04667. — 2024. — URL: https://arxiv.org/abs/2408.04667.

77. Language (Technology) is Power: A Critical Survey of "Bias"in NLP [Текст] / S. L. Blodgett [и др.] // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. — 2020. — С. 5454—5476. — URL: https://arxiv.org/abs/2005.14050.

78. A Survey on Bias and Fairness in Machine Learning [Текст] / N. Mehrabi [и др.] // ACM Computing Surveys. — 2021. — Т. 54, № 6. — С. 1—35. — URL: https://arxiv.org/abs/1908.09635.

79. *Shah, S.* Evaluating and Mitigating Bias in Image Classifiers: A Causal Perspective Using Counterfactuals [Текст] / S. Shah, A. G. Schwing, A. Galstyan // arXiv preprint arXiv:2009.08270. — 2020. — URL: https://arxiv.org/abs/2009.08270.

80. Backdoor Attacks in the Era of Large Language Models [Текст] / Y. Li [и др.] // arXiv preprint arXiv:2308.14367. — 2023. — URL: https://arxiv.org/abs/2308.14367.

81. Privacy and Utility Trade-offs in Large Language Models [Текст] / M. Du [и др.] // arXiv preprint arXiv:2402.01003. — 2024. — URL: https://arxiv.org/abs/2402.01003.

82. *Wang, T.* The Empirical Impact of Data Sanitization on Language Models [Текст] / T. Wang, R. Bommasani, P. Liang // OpenReview. — 2023. — URL: https://openreview.net/forum?id=Jron0NaeFt.

83. MBIAS: Mitigating Bias in Large Language Models While Retaining Context [Текст] / Y. Li [и др.] // arXiv preprint arXiv:2405.11290. — 2024. — URL: https://arxiv.org/abs/2405.11290.

84. Communication-Efficient Learning of Deep Networks from Decentralized Data [Текст] / B. McMahan [и др.] // Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. — 2017. — С. 1273—1282. — URL: https://arxiv.org/abs/1602.05629.

85. Constitutional AI: Harmlessness from AI Feedback [Текст] / Y. Bai [и др.]. — 12.2022. — URL: http://arxiv.org/abs/2212.08073 (дата обр. 03.02.2024).

86. Securing the AI Software Supply Chain [Текст] / A. Kumar [и др.] // Google Research. — 2023. — URL: https://research.google/pubs/securing-the-ai-software-supply-chain.

87. Prompt Injection Attacks and Defenses in LLM-Integrated Applications [Текст] / K. Liu [и др.] // arXiv preprint arXiv:2310.12815. — 2023. — URL: https://arxiv.org/abs/2310.12815.

88. StruQ: Defending Against Prompt Injection with Structured Queries [Текст] / S. Chen [и др.]. — 02.2024. — URL: https://doi.org/10.48550/arXiv.2402.06363.

89. *Wang, T.* Casper: Prompt Sanitization for Protecting User Privacy in Large Language Model Services [Текст] / T. Wang, Y. Jiang, X. Ren // arXiv preprint arXiv:2408.07004. — 2024. — URL: https://arxiv.org/html/2408.07004v1.

90. *Al-Khateeb, F.* Identifying and Mitigating API Misuse in Large Language Models [Текст] / F. Al-Khateeb, N. Dey, D. Soboleva // arXiv preprint arXiv:2503.22821. — 2023. — URL: https://arxiv.org/abs/2503.22821.

91. Crabs: Consuming Resource via Auto-generation for LLM-DoS Attack under Black-box Settings [Текст] / J. Zhang [и др.] // arXiv preprint arXiv:2412.13879. — 2023. — URL: https://arxiv.org/abs/2412.13879.

92. Trojaning Plugins of Large Language Models [Текст] / X. Wang [и др.] // NDSS Symposium. — 2023. — URL: https://www.ndss-symposium.org/wp-content/uploads/2025-s164-paper.pdf.

93. Red Teaming Language Models with Language Models [Текст] / E. Perez [и др.]. — 02.2022. — URL: http://arxiv.org/abs/2202.03286 (дата обр. 28.03.2024).

94. Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned [Текст] / D. Ganguli [и др.]. — 11.2022. — URL: http://arxiv.org/abs/2209.07858 (дата обр. 28.03.2024).

95. Auditing large language models: a three-layered approach [Текст] / I. D. Raji [и др.] // AI and Ethics. — 2023. — С. 1—15. — URL: https://link.springer.com/article/10.1007/s43681-023-00289-2.

96. *Melnick, T.* CVSS: Common Vulnerability Scoring System Version 4.0 Specification [Текст] / T. Melnick, J. Spring, A. Householder // Forum of Incident Response and Security Teams (FIRST). — 2023. — URL: https://www.first.org/cvss/v4.0/specification-document.

97. *Tarjan, R.* Depth-First Search and Linear Graph Algorithms [Текст] / R. Tarjan // SIAM Journal on Computing. — 1972. — Т. 1, № 2. — С. 146—160.

98. Introduction to Algorithms [Текст] / T. H. Cormen [и др.]. — 3-е изд. — Cambridge, MA : MIT Press, 2009.

99. *Ba, J. L.* Layer Normalization [Текст] / J. L. Ba, J. R. Kiros, G. E. Hinton // arXiv preprint arXiv:1607.06450. — 2016. — URL: https://arxiv.org/abs/1607.06450.

100. Improving Natural Language Understanding by Matching Sentence Representations Bidirectionally [Текст] / S. Wang [и др.] // arXiv preprint arXiv:1904.04733. — 2019. — URL: https://arxiv.org/abs/1904.04733.

101. Stabilizing Transformer Training by Preventing Attention Entropy Collapse [Текст] / S. Zhai [и др.] // Proceedings of the 40th International Conference on Machine Learning (ICML 2023). — 2023. — URL: https://arxiv.org/abs/2303.06296.

102. Certified Robustness for Large Language Models with Self-Denoising [Текст] / Z. Zhang [и др.] // International Conference on Machine Learning (ICML). — 2023. — URL: https://arxiv.org/abs/2307.07171.

103. *Kim, H.* The Lipschitz Constant of Self-Attention [Текст] / H. Kim, G. Papamakarios, A. Mnih // Proceedings of the 38th International Conference on Machine Learning. — 2021. — С. 5562—5571. — URL: https://arxiv.org/abs/2006.04710.

104. Improving the Robustness of Transformer-based Large Language Models with Dynamic Attention [Текст] / L. Shen [и др.] // arXiv preprint arXiv:2311.17400. — 2023. — URL: https://arxiv.org/abs/2311.17400.

105. Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Текст] / N. Srivastava [и др.] // Journal of Machine Learning Research. — 2014. — Т. 15, № 56. — С. 1929—1958. — URL: http://jmlr.org/papers/v15/srivastava14a.html.

106. Regularization of Neural Networks using DropConnect [Текст] / L. Wan [и др.] // Proceedings of the 30th International Conference on Machine Learning. — 2013. — С. 1058—1066. — URL: https://proceedings.mlr.press/v28/wan13.html.

107. Deep Networks with Stochastic Depth [Текст] / G. Huang [и др.] // European Conference on Computer Vision. — 2016. — С. 646—661. — URL: https://arxiv.org/abs/1603.09382.

108. *Gal, Y.* Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning [Текст] / Y. Gal, Z. Ghahramani // Proceedings of the 33rd International Conference on Machine Learning. — 2016. — Т. 48. — С. 1050—1059. — URL: https://arxiv.org/abs/1506.02142.

109. *Lakshminarayanan, B.* Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles [Текст] / B. Lakshminarayanan, A. Pritzel, C. Blundell // Advances in Neural Information Processing Systems. — 2017. — Т. 30. — URL: https://arxiv.org/abs/1612.01474.

110. *Cover, T. M.* Elements of Information Theory [Текст] / T. M. Cover, J. A. Thomas. — 2nd. — Wiley, 2006.

111. Film: Frame interpolation for large motion [Текст] / S. Perez [и др.] // arXiv preprint arXiv:2202.04901. — 2021. — URL: https://arxiv.org/abs/2202.04901.

112. Adversarial Distributional Training for Robust Deep Learning [Текст] / Y. Dong [и др.] // arXiv preprint arXiv:2002.05999. — 2020. — URL: https://arxiv.org/abs/2002.05999.

113. *Gao, T.* SimCSE: Simple Contrastive Learning of Sentence Embeddings [Текст] / T. Gao, X. Yao, D. Chen // Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. — 2021. — С. 6894—6910. — URL: https://arxiv.org/abs/2104.08821.

114. *Sennrich, R.* Neural Machine Translation of Rare Words with Subword Units [Текст] / R. Sennrich, B. Haddow, A. Birch // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. — 2016. — Т. 1. — С. 1715—1725. — URL: https://arxiv.org/abs/1508.07909.

115. *Oord, A. van den.* Representation Learning with Contrastive Predictive Coding [Текст] / A. van den Oord, Y. Li, O. Vinyals // arXiv preprint arXiv:1807.03748. — 2018. — URL: https://arxiv.org/abs/1807.03748.

116. A Simple Framework for Contrastive Learning of Visual Representations [Текст] / T. Chen [и др.] // Proceedings of the 37th International Conference on Machine Learning. — 2020. — Т. 119. — С. 1597—1607. — URL: https://arxiv.org/abs/2002.05709.

117. Momentum Contrast for Unsupervised Visual Representation Learning [Текст] / K. He [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2020. — С. 9729—9738. — URL: https://arxiv.org/abs/1911.05722.

118. Text-CRS: A Generalized Certified Robustness Framework against Textual Adversarial Attacks [Текст] / H. Ye [и др.] // arXiv preprint arXiv:2307.16630. — 2023. — URL: https://arxiv.org/abs/2307.16630.

119. From Robustness to Improved Generalization and Calibration in Pre-trained Language Models [Текст] / Y. Zhou [и др.] // arXiv preprint arXiv:2404.00758. — 2024. — URL: https://arxiv.org/abs/2404.00758.

120. On Calibration of Modern Neural Networks [Текст] / C. Guo [и др.] // Proceedings of the 34th International Conference on Machine Learning. — 2017. — Т. 70. — С. 1321—1330. — URL: https://arxiv.org/abs/1706.04599.

121. HotFlip: White-Box Adversarial Examples for Text Classification [Текст] / J. Ebrahimi [и др.]. — 05.2018. — URL: http://arxiv.org/abs/1712.06751 (дата обр. 23.03.2024).

122. FreeLB: Enhanced Adversarial Training for Natural Language Understanding [Текст] / C. Zhu [и др.] // arXiv preprint arXiv:1909.11764. — 2019. — URL: https://arxiv.org/abs/1909.11764.

123. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment [Текст] / D. Jin [и др.]. — 2020. — arXiv: 1907.11932 [cs.CL]. — URL: https://arxiv.org/abs/1907.11932.

124. BERT-EMD: Many-to-Many Layer Mapping for BERT Compression with Earth Mover's Distance [Текст] / J. Li [и др.] // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. — 2020. — С. 3009—3018. — URL: https://arxiv.org/abs/2010.06133.

125. Robustness May Be at Odds with Accuracy [Текст] / D. Tsipras [и др.]. — 2019. — arXiv: 1805.12152 [stat.ML]. — URL: https://arxiv.org/abs/1805.12152.

126. Theoretically Principled Trade-off between Robustness and Accuracy [Текст] / H. Zhang [и др.]. — 2019. — arXiv: 1901.08573 [cs.LG]. — URL: https://arxiv.org/abs/1901.08573.

127. *Hoffman, J.* Robust Learning with Jacobian Regularization [Текст] / J. Hoffman, D. A. Roberts, S. Yaida // arXiv preprint arXiv:1908.02729. — 2019. — URL: https://arxiv.org/abs/1908.02729.

128. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning [Текст] / T. Miyato [и др.] // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2018. — Т. 41, № 8. — С. 1979—1993. — URL: https://arxiv.org/abs/1704.03976.

129. *Wei, J.* EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks [Текст] / J. Wei, K. Zou // arXiv preprint arXiv:1901.11196. — 2019. — URL: https://arxiv.org/abs/1901.11196.

130. mixup: Beyond Empirical Risk Minimization [Текст] / H. Zhang [и др.] // arXiv preprint arXiv:1710.09412. — 2017. — URL: https://arxiv.org/abs/1710.09412.

131. *Chen, J.* MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification [Текст] / J. Chen, Z. Yang, D. Yang // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. — 2020. — С. 2147—2157. — URL: https://arxiv.org/abs/2004.12239.

132. Manifold Mixup: Better Representations by Interpolating Hidden States [Текст] / V. Verma [и др.] // arXiv preprint arXiv:1806.05236. — 2019. — URL: https://arxiv.org/abs/1806.05236.

133. *Hinton, G.* Distilling the Knowledge in a Neural Network [Текст] / G. Hinton, O. Vinyals, J. Dean // arXiv preprint arXiv:1503.02531. — 2015. — URL: https://arxiv.org/abs/1503.02531.

134. Robust Knowledge Distillation from RNN-T Models With Noisy Training Labels [Текст] / W. Wang [и др.] // arXiv preprint arXiv:2303.05958. — 2023. — URL: https://arxiv.org/abs/2303.05958.

135. Adversarially Robust Distillation [Текст] / M. Goldblum [и др.] // Proceedings of the AAAI Conference on Artificial Intelligence. — 2020. — Т. 34, № 04. — С. 3996—4003. — URL: https://arxiv.org/abs/1905.09747.

136. Ensemble Distillation for Robust Model Fusion in Federated Learning [Текст] / T. Lin [и др.] // Advances in Neural Information Processing Systems. — 2020. — С. 2351—2363. — URL: https://proceedings.neurips.cc/paper/2020/file/18df51b97ccd68128e994804f3eccc87-Paper.pdf.

137. Ensemble Adversarial Training: Attacks and Defenses [Текст] / F. Tramèr [и др.]. — 2020. — arXiv: 1705.07204 [stat.ML]. — URL: https://arxiv.org/abs/1705.07204.

138. Trading-Off Interpretability and Accuracy in Medical Applications: A Study Toward Optimal Explainability of Hoeffding Trees [Текст] / A. Sharma [и др.] // 2024 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). — 2024. — С. 1—10.

139. Motivating the Rules of the Game for Adversarial Example Research [Текст] / J. Gilmer [и др.]. — 2018. — arXiv: 1807.06732 [cs.LG]. — URL: https://arxiv.org/abs/1807.06732.

140. On Evaluating Adversarial Robustness [Текст] / N. Carlini [и др.]. — 2019. — arXiv: 1902.06705 [cs.LG]. — URL: https://arxiv.org/abs/1902.06705.

141. *Hoeffding, W.* Probability Inequalities for Sums of Bounded Random Variables [Текст] / W. Hoeffding // Journal of the American Statistical Association. — 1963. — Т. 58, № 301. — С. 13—30.

142. The Trojan Detection Challenge 2023 (LLM Edition) - The Trojan Detection Challenge [Текст]. — URL: https : / / trojandetection . ai/ (дата обр. 06.03.2024).

143. PyTorch: An Imperative Style, High-Performance Deep Learning Library [Текст] / A. Paszke [и др.] // Advances in Neural Information Processing Systems. Т. 32. — Curran Associates, Inc., 2019.

144. Transformers: State-of-the-Art Natural Language Processing [Текст] / T. Wolf [и др.] // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. — Association for Computational Linguistics, 2020. — С. 38—45.

145. Efficient Memory Management for Large Language Model Serving with PagedAttention [Текст] / W. Kwon [и др.] // Proceedings of the 29th Symposium on Operating Systems Principles. — ACM, 2023. — С. 611—626.

146. *Dodson, D.* Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities [Текст] : tech. rep. / D. Dodson, M. Souppaya, K. Scarfone ; National Institute of Standards ; Technology. — 02/2022. — NIST SP 800—218. — URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf.

147. *Athalye, A.* Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples [Текст] / A. Athalye, N. Carlini, D. Wagner // Proceedings of the 35th International Conference on Machine Learning. — 2018. — Т. 80. — С. 274—283. — URL: https://arxiv.org/abs/1802.00420.

148. *Salehie, M.* Self-adaptive software: Landscape and research challenges [Текст] / M. Salehie, L. Tahvildari // ACM Transactions on Autonomous and Adaptive Systems. — 2009. — Май. — Т. 4, № 2. — С. 1—42. — URL: https://doi.org/10.1145/1516533.1516538.

149. Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback [Текст] / B. Peng [и др.]. — 2023. — URL: https://arxiv.org/abs/2302.12813 (дата обр. 25.01.2025).

150. FacTool: Factuality Detection in Generative AI – A Tool Augmented Framework for Multi-Task and Multi-Domain Scenarios [Текст] / I.-C. Chern [и др.]. — 2023. — URL: https://arxiv.org/abs/2307.13528 (дата обр. 25.01.2025).

151. IRIS: LLM-Assisted Static Analysis for Detecting Security Vulnerabilities [Текст] / P. Pearce [и др.]. — 2024. — URL: https://openreview.net/forum?id=9LdJDU7E91.

152. NEUROLOGIC A*esque Decoding: Constrained Text Generation with Lookahead Heuristics [Текст] / X. Lu [и др.] // arXiv preprint arXiv:2112.08726. — 2022. — URL: http://arxiv.org/abs/2112.08726.

153. Robust Neural Information Retrieval: An Adversarial and Out-of-distribution Perspective [Текст] / H. Zhang [и др.] // arXiv preprint arXiv:2407.06992. — 2024. — URL: http://arxiv.org/abs/2407.06992.

154. *Yun, H.* Ranking via Robust Binary Classification [Текст] / H. Yun, P. Raman, S. V. N. Vishwanathan // Advances in Neural Information Processing Systems. — 2014. — С. 2582—2590. — URL: http://papers.nips.cc/paper/5363-ranking-via-robust-binary-classification.pdf.

155. FairRoad: Achieving Fairness for Recommender Systems with Optimized Antidote Data [Текст] / A. Ferraro [и др.] // arXiv preprint arXiv:2212.06750. — 2022. — URL: http://arxiv.org/abs/2212.06750.

156. *McSherry, F.* Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders [Текст] / F. McSherry, I. Mironov // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. — 2009. — С. 627—636. — URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2009/06/NetflixPrivacy.pdf.

157. Retrieval-Augmented Generation for Large Language Models: A Survey [Текст] / Y. Gao [и др.]. — 01.2024. — URL: http://arxiv.org/abs/2312.10997 (дата обр. 03.02.2024).

158. Medical large language models are vulnerable to data-poisoning attacks [Текст] / D. A. Alber [et al.] // Nature Medicine. — 2025. — Jan. — P. 1—9. — URL: https://www.nature.com/articles/s41591-024-03445-1 (visited on 01/26/2025).

159. *Cohen, J.* Certified adversarial robustness via randomized smoothing [Текст] / J. Cohen, E. Rosenfeld, Z. Kolter // International Conference on Machine Learning. — PMLR, 2019. — С. 1310—1320.

160. Explainability for Large Language Models: A Survey [Текст] / W. X. Zhao [и др.] // ACM Computing Surveys. — 2023. — Т. 56, № 4. — С. 1—38. — URL: http://arxiv.org/abs/2309.01029.

161. Addressing fairness in artificial intelligence for medical imaging [Текст] / I. Y. Chen [и др.] // Nature Communications. — 2022. — Т. 13, № 1. — С. 4453. — URL: https://www.nature.com/articles/s41467-022-32186-3.

162. *Abdallah, A.* Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances [Текст] / A. Abdallah, M. A. Maarof, A. Zainal // Expert Systems with Applications. — 2022. — Т. 193. — С. 116421. — URL: https://www.sciencedirect.com/science/article/pii/S0957417421017164.

163. RobustTSF: Towards Theory and Design of Robust Time Series Forecasting with Anomalies [Текст] / H. Cheng [и др.] // Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024). — 2024. — URL: https://arxiv.org/abs/2402.02032.

164. The Fusion of Large Language Models and Formal Methods for Verified Code Generation [Текст] / H. Zhang [и др.] // arXiv preprint arXiv:2412.06512. — 2023. — URL: https://arxiv.org/abs/2412.06512.

165. A Survey of Safety and Trustworthiness of Large Language Models through the Lens of Verification and Validation [Текст] / X. Huang [и др.] // arXiv preprint arXiv:2305.11391. — 2023. — URL: https://arxiv.org/abs/2305.11391.

166. *García-Carrasco, J.* Detecting and Understanding Vulnerabilities in Language Models via Mechanistic Interpretability [Текст] / J. García-Carrasco, A. Maté, J. Trujillo // Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24). — 2024. — С. 385—393. — URL: https://arxiv.org/abs/2407.19842.

167. JailbreakLens: Visual Analysis of Jailbreak Attacks Against Large Language Models [Текст] / Y. Feng [и др.]. — 04.2024. — URL: http://arxiv.org/abs/2404.08793 (дата обр. 26.01.2025).

168. From LLMs to MLLMs: Exploring the Landscape of Multimodal Jailbreaking [Текст] / S. Wang [и др.]. — 2024. — Июнь.

169. Agent Smith: A Single Image Can Jailbreak One Million Multimodal LLM Agents Exponentially Fast [Текст] / X. Gu [и др.] // Proceedings of the 41st International Conference on Machine Learning. Т. 235. — 06.2024. — URL: https://github.com/sail-sg/Agent-Smith.

170. Exploring the Robustness of Decentralized Training for Large Language Models [Текст] / L. Lu [и др.] // arXiv preprint arXiv:2312.00843. — 2023. — URL: https://arxiv.org/abs/2312.00843.

171. Robust Federated Learning against both Data Heterogeneity and Poisoning Attack via Aggregation Optimization [Текст] / Y. Xie [и др.] // arXiv preprint arXiv:2211.05554. — 2022. — URL: https://arxiv.org/abs/2211.05554.

172. *Bagdasaryan, E.* Differential Privacy Has Disparate Impact on Model Accuracy [Текст] / E. Bagdasaryan, O. Poursaeed, V. Shmatikov // Advances in Neural Information Processing Systems. — 2020. — Т. 33. — С. 15479—15488. — URL: https://arxiv.org/abs/1905.12101.

173. The Predominant Vulnerability of LLM-Integrated Applications: Unauthorized Data Access via Prompt Injection [Текст] / A. Kumar [и др.] // arXiv preprint arXiv:2402.09717. — 2024. — URL: https://arxiv.org/abs/2402.09717.

174. SELFDEFEND : LLMs Can Defend Themselves against Jailbreaking in a Practical Manner [Текст] / X. Wang [и др.] // arXiv:2406.05498v2. — 09.2024.

175. Gandalf the Red: Adaptive Security for LLMs [Текст] / N. Pfister [и др.]. — 01.2025. — URL: http://arxiv.org/abs/2501.07927 (дата обр. 25.01.2025).

176. JailbreakBench : An Open Robustness Benchmark for Jailbreaking Large Language Models [Текст] / P. Chao [и др.] // 38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks. — 10.2024. — URL: https://arxiv.org/abs/2404.01318v5.

177. Buffer Memory Transformer for Efficient Long Context Processing [Текст] / Z. Wang [и др.] // arXiv preprint arXiv:2402.17764. — 2024. — URL: https://arxiv.org/abs/2402.17764.

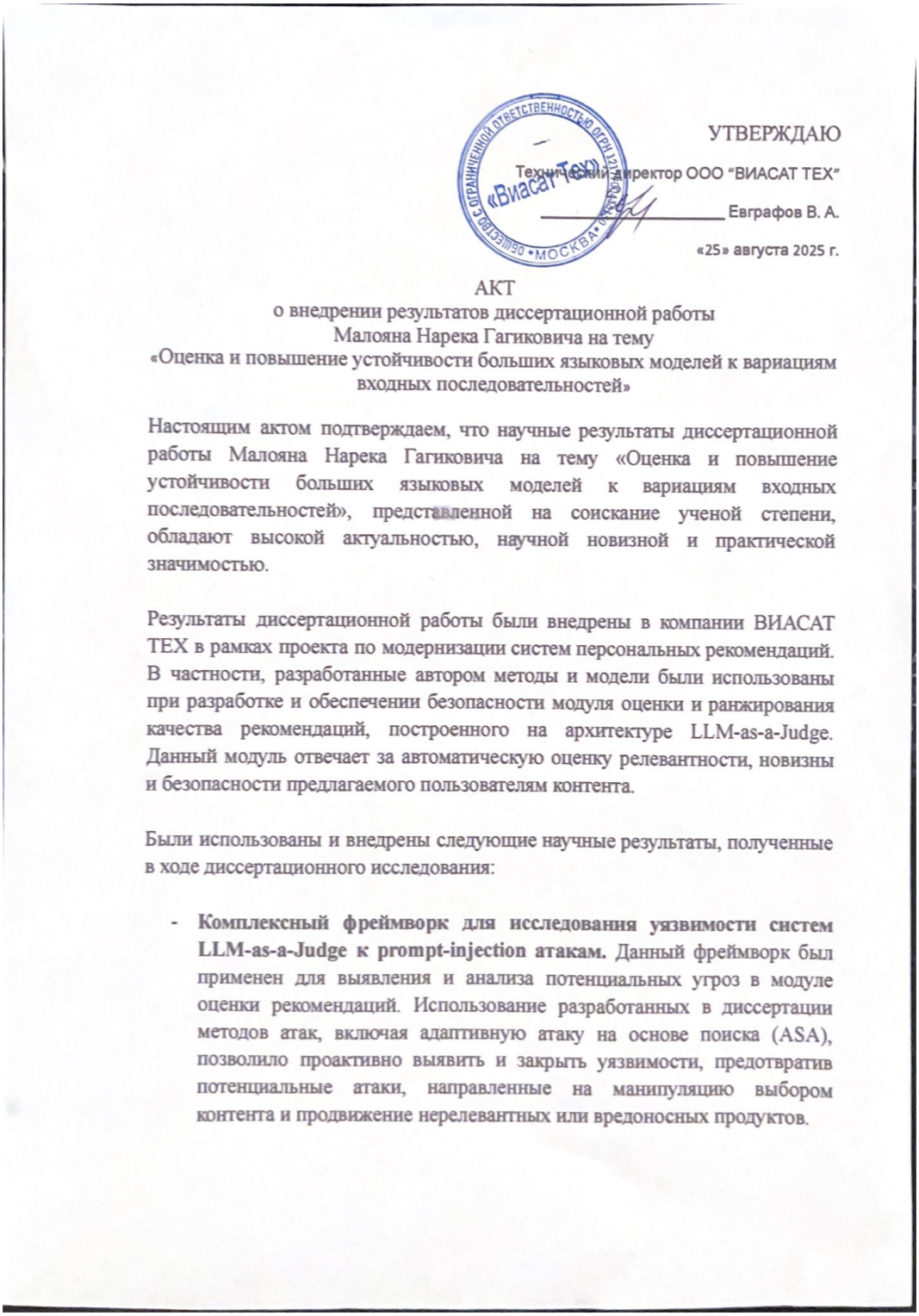
178. *Lee, J.* Large Language Models in Finance (FinLLMs) [Текст] / J. Lee, N. Stevens, S. C. Han // Neural Computing and Applications. — 2025. — Jan. — URL: https://link.springer.com/10.1007/s00521-024-10495-6 (visited on 01/25/2025).

Список рисунков

Список таблиц

Приложение А

Акт о внедрении



УТВЕРЖДАЮ

Технический директор ООО "ВИАСАТ ТЕХ"

_____ Евграфов В. А.

«25» августа 2025 г.

АКТ

о внедрении результатов диссертационной работы
Малояна Нарека Гагиковича на тему
«Оценка и повышение устойчивости больших языковых моделей к вариациям входных последовательностей»

Настоящим актом подтверждаем, что научные результаты диссертационной работы Малояна Нарека Гагиковича на тему «Оценка и повышение устойчивости больших языковых моделей к вариациям входных последовательностей», представленной на соискание ученой степени, обладают высокой актуальностью, научной новизной и практической значимостью.

Результаты диссертационной работы были внедрены в компании ВИАСАТ ТЕХ в рамках проекта по модернизации систем персональных рекомендаций. В частности, разработанные автором методы и модели были использованы при разработке и обеспечении безопасности модуля оценки и ранжирования качества рекомендаций, построенного на архитектуре LLM-as-a-Judge. Данный модуль отвечает за автоматическую оценку релевантности, новизны и безопасности предлагаемого пользователям контента.

Были использованы и внедрены следующие научные результаты, полученные в ходе диссертационного исследования:

- **Комплексный фреймворк для исследования уязвимости систем LLM-as-a-Judge к prompt-injection атакам.** Данный фреймворк был применен для выявления и анализа потенциальных угроз в модуле оценки рекомендаций. Использование разработанных в диссертации методов атак, включая адаптивную атаку на основе поиска (ASA), позволило проактивно выявить и закрыть уязвимости, предотвратив потенциальные атаки, направленные на манипуляцию выбором контента и продвижение нерелевантных или вредоносных продуктов.

- **Методика использования комитетов из нескольких LLM с разнообразными архитектурами для защиты от состязательных атак.** Этот результат был реализован в качестве основного механизма защиты модуля оценки. Внедрение ансамбля из нескольких моделей-судей позволило существенно повысить устойчивость системы к prompt-injection атакам, обеспечив снижение их потенциальной успешности (ASR) на десятки процентных пунктов и гарантировав стабильность и надежность процесса ранжирования рекомендаций.

- **Разработанная методология генерации многошаговых prompt-injection атак на LLM с многоуровневой защитой.** Данная методология использовалась в рамках внутреннего тестирования (Red Teaming) для проведения стресс-тестов разработанной системы защиты рекомендательного сервиса. Это позволило убедиться в эффективности implemented защитных барьеров, включая системные инструкции и внешние фильтры, против целенаправленных и сложных атак.

- **Предложенная формальная математическая модель для оценки устойчивости LLM.** Модель была адаптирована и использована в качестве инструмента для количественного мониторинга стабильности работы моделей-оценщиков в продуктивной среде. Это позволило отслеживать деградацию или улучшение робастности моделей при их обновлении и проводить сравнительный анализ различных архитектур защиты на этапе их выбора для внедрения.

Внедрение указанных результатов позволило существенно повысить надежность, безопасность и предсказуемость работы рекомендательных систем Viasat, основанных на больших языковых моделях, а также защитить пользователей от потенциальных информационных угроз и манипуляций.

Руководитель отдела машинного обучения /Тонких А.А. /

ООО "ВИАСАТ ТЕХ"