Архитектура LLM-агентов

Д.Е. Намиот, Е.А. Ильюшин

***Аннотация*—В работе рассматриваются архитектуры агентов для систем Искусственного интеллекта. Слово "агент" стало темой 2024 года для такого рода систем. Агентный Искусственный интеллект рассматривается как следующий шаг в развитии генеративных моделей. Соответственно, агенты для больших языковых моделей (базовых моделей) представляют собой одно из наиболее бурно развивающихся направлений исследований. Агент искусственного интеллекта представляет собой программу (программное обеспечение), которое умеет собирать данные и на их основе самостоятельно и выполнять задачи, позволяющие добиться заранее определенных целей. Для выполнения задач агенты задействуют одну или несколько языковых моделей. Агенты можно рассматривать как логичный шаг в помощи разработчикам создавать рабочие процессы (реализовывать бизнес-модели) с помощью больших языковых (базовых) моделей. С точки зрения сбора (анализа) информации агенты можно сравнить с давно известными мэшапами (веб-мэшапами, например), с точки зрения выполнения каких-либо действий – с программными роботами. Большие языковые модели сегодня могут содержать встроенную поддержку некоторых рабочих процессов. Альтернатива – разного рода фреймворки, которые призваны упростить процесс создания агентов.**

***Ключевые слова* — искусственный интеллект, агенты, генеративные модели.**

I. Введение

Слово агент (агентный), применительно к системам Искусственного интеллекта (ИИ) стало по-настоящему модным в 2024 году. Как обычно, в роли систем ИИ выступают большие языковые модели (LLM или уже большие многомодальные модели, которые работают не только с текстом). 2024 год во множестве аналитических обзоров назван годом агентов ИИ.

В рисках, ассоциируемых с использованием генеративных моделей ИИ [1] присутствует позиция, связанная с излишним "очеловечиванием" таких моделей, наделением их реально отсутствующими характеристиками. Это во многом относится и к агентам, которым часто приписывают многие "сверхспособности". Настоящая статья – это попытка описать архитектуру LLM агентов с более приземленной, технической стороны.

Статья получена 9 ноября 2024.
Д.Е. Намиот – МГУ имени М.В. Ломоносова (email: dnamiot@gmail.com)
Е.А. Ильюшин – МГУ имени М.В. Ломоносова (john.ilyushin@gmail.com).

Как отмечено в [2], одной из самых значимых тенденций разработки в области генеративного ИИ является переход от монолитных моделей к сложным, составным системам ИИ (рис.1).

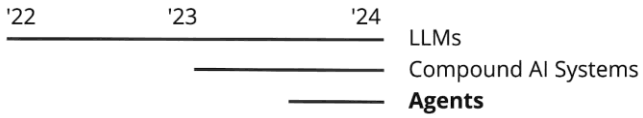

Рис.1. "Составной ИИ" [2]

Причины – возможность повысить эффективность работы за счет оптимизации (оптимального использования) отдельных компонент. Есть очень мало возможностей влиять на работу монолитной системы. Если же появляются отдельные компоненты, то можно уже влиять на процесс (качество) работы системы.

Действительно, идеи улучшения взаимодействия с LLM появились практически сразу же после запуска ChatGPT. Вся тема Prompt Engineering (инженерия подсказок) [3] – это о выстраивании процесса работы. Далее появились пре-промпты (системные подсказки для конкретных запросов) и RAG – обогащение подсказок результатами запроса (запросов) к базе (базам) документов [4]. Все это относится именно к выстраиванию рабочих процессов для решения задач с помощью LLM. Агенты – это просто следующий логичный шаг в помощи разработчикам создавать рабочие процессы (реализовывать бизнес-модели) с помощью LLM.

Как аналог такого подхода можно упомянуть, например, мэшапы, которые как раз и занимались тем, что собирали данные из различных источников. Итак – агенты LLM представляют собой программы, которые используют LLM (одну или несколько) в процессе работы. В любом случае, агент – это некоторый посредник, который скрывает непосредственный доступ к LLM от конечных пользователей (которыми, кстати, могут быть и другие агенты). Агент (программа агента, агентская программа) может (как и любая другая программа) получать данные из своего окружения (ввод данных пользователем/агентом/другой программой), данные от сенсоров и т.п., использовать одну или несколько LMM для обработки (интерпретации, уточнений, обогащения и т.п.), обращаться к другим программным системам (агентам или нет), выстраивать процессы (конвейеры) обработки (не обязательно синхронные), заниматься поиском и оркестровкой, запоминать (сохранять) данные для последующего использования и т.д. Основой всего, как и в любой

программной интеграции, являются различные программные интерфейсы (API).

Именно программные интерфейсы обеспечили возможность итеративного взаимодействия с LLM, что сразу привело к гораздо более высокой производительности в ряде приложений.

Что произошло: ИИ получил новое модное слово — агентный — поскольку исследователи, поставщики инструментов и разработчики моделей оснастили большие языковые модели (LLM) возможностью делать выбор и предпринимать действия для достижения целей. Появилось несколько инструментов, помогающих разработчикам создавать агентские рабочие процессы. Появились фреймворки, такие как Autogen [5], его наследник AG2 [6]. Причем фреймворки изначально ориентировались на многоагентные системы: рабочие процессы строились как взаимодействие разных агентов. На рисунке 2 изображен пример от Autogen: генерация дополнение) кода и его проверка – это разные агенты.

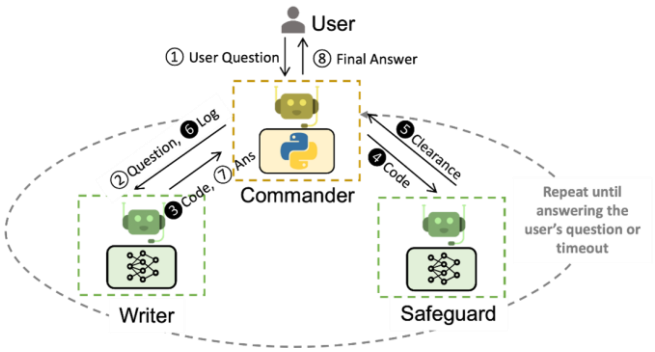

Рис. 2. Многоагентная система co-pilot [7].

Следующая цитата с сайта AG2 наилучшим образом объясняет, что делает этот фреймворк (выдеоение курсивом – наше): “AG2 — это программная среда с открытым исходным кодом для создания агентов ИИ и поддержки взаимодействия между несколькими агентами для решения задач. AG2 нацелен на оптимизацию разработки и исследования агентного ИИ, *подобно тому, как это делает PyTorch для глубокого обучения*. Он предлагает такие возможности, как агенты, способные взаимодействовать друг с другом, облегчает использование различных больших языковых моделей (LLM) и поддержку использования инструментов, поддерживает автономные и управляемые человеком рабочие процессы, а также предоставляет шаблоны многоагентных обменов”.

В октябре 2023 года CrewAI [8] выпустила свою среду с открытым исходным кодом для Python, предназначенную для создания и управления многоагентными системами. Агентам можно назначать роли и цели, они могут получать доступ к таким инструментам, как веб-поиск, и сотрудничать друг с другом. Интересно, что предоставляется множество шаблонов (подготовленных агентов, которые можно просто конфигурировать) для различных предметных областей.

Тем самым создается рынок компонент для разработки агентов. На рисунке 3 представлены основные концепции CrewAI

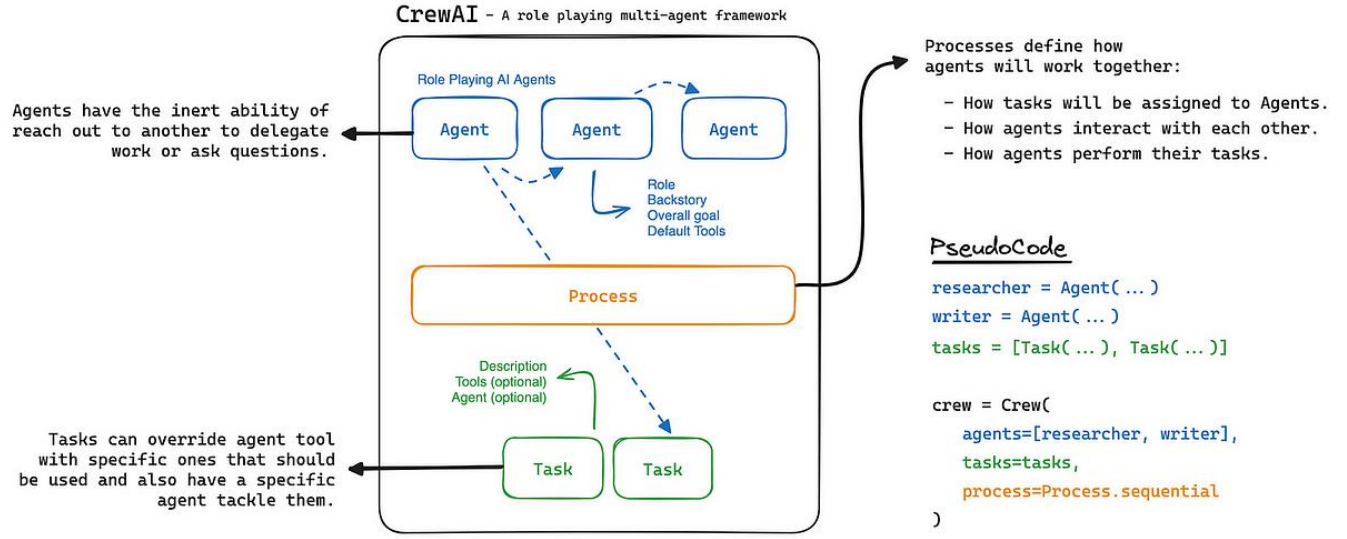

Рис. 3 CrewAI [9].

Интересно, что именно CrewAI широко представлен в учебных курсах на Coursera и deeplearning.ai. Материалы последнего сайта использовались при подготовке данного обзора.

В январе 2024 года LangChain [10], поставщик инструментов разработки (фреймворк для работы с LLM), представил LangGraph - это фреймворк оркестровки для управляемых агентских рабочих процессов, который организует поведение агентов с помощью циклических графов. Эта среда позволяет агентам, управляемым LLM, получать входные данные, строить рассуждения, принимать решения о действиях, использовать различные инструменты, оценивать

результаты и повторять эти шаги для улучшения результатов.

На рисунке 4 представлен пример интеграции LangGraph и упоминавшегося выше CrewAI

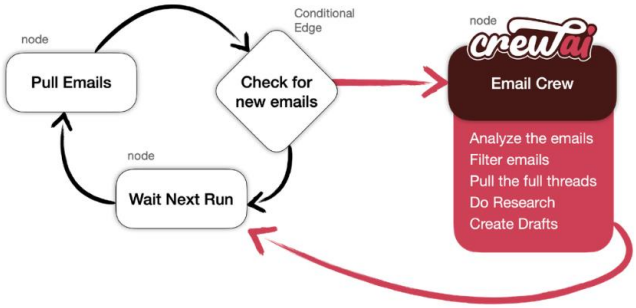

Рис.4. Агент, готовящий ответы на входящие письма [11]

Первые аналоги, которые подходят для сравнения – это Robotic Process Automation (RPA) [12, 13]. Собственно говоря, это вполне корректно представлять LLM агенты именно такими программными роботами.

В сентябре 2024 Meta представила Llama Stack [14] для создания агентских приложений на основе моделей Llama. Llama Stack определяет и стандартизирует набор основных строительных блоков (компонент), необходимых для вывода на рынок приложений генеративного ИИ. Эти компоненты представлены в виде совместимых API (REST API) с широким набором провайдеров услуг, предоставляющих свои реализации (рис. 5).

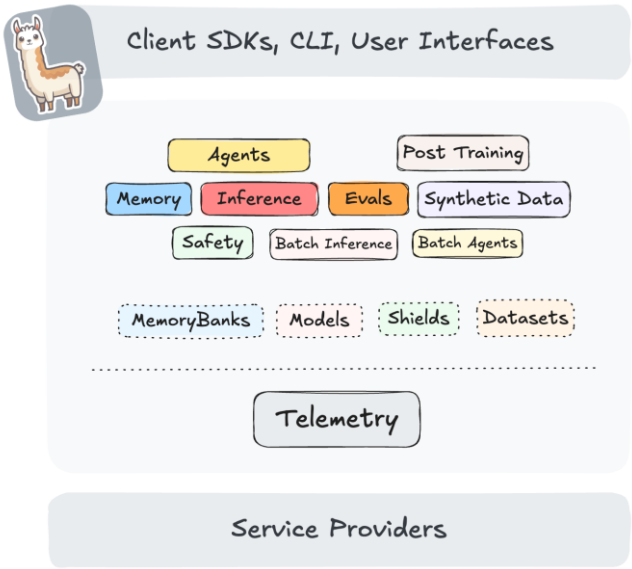

Рис.5 Llama stack [14]

Появились специализированные ИИ-агенты. Например, агентные рабочие процессы для генерации кода в интегрированных средах разработки. В частности, Devin (AI Software Developer) [15] и OpenHands [16] принимают инструкции на естественном языке для генерации прототипов программ. Агенты OpenHands могут делать все то же, что и разработчик-человек: изменять код, запускать команды, просматривать веб-страницы, вызывать API и даже копировать фрагменты кода из StackOverflow.

Другие примеры из той же области: Replit Agent [17], Vercel's V0 [18] и Bolt [19] оптимизируют проекты, автоматически записывая код, исправляя ошибки и управляя зависимостями.

Агентные рабочие процессы могут непосредственно поддерживаться в LLM. Пример – Anthropic Claude 3.5 Sonnet [20] и ее возможности напрямую управлять компьютерами пользователей. Внутренняя агентная оценка кодирования [21] разрешает модели писать и запускать код в агентном цикле и итеративно самокорректироваться во время оценки.

В конце 2024 года OpenAI выпустила свои модели o1 и режим o1 pro [22] с интенсивной обработкой данных, которые используют агентные циклы для пошаговой работы с подсказками (сейчас есть уже o3 и o3-preview). Большая коллекция агентов содержится в репозитории [23].

В декабре 2024 года появился и Google Gemini 2.0 [24]. Собственные слова от Google: “За последний год мы инвестировали в разработку большего количества агентных моделей, то есть они могут лучше понимать мир вокруг вас, думать на несколько шагов вперед и действовать от вашего имени под вашим руководством.

Сегодня мы рады представить нашу следующую эру моделей, созданных для этой новой агентной эры: представляем Gemini 2.0, нашу самую мощную модель на сегодняшний день. Благодаря новым достижениям в мультимодальности — таким как собственный вывод изображений и звука — и использованию собственных инструментов, это позволит нам создавать новых агентов ИИ, которые приближают нас к нашему видению универсального помощника.” Иными словами, агенты в LLM – это встроенная цепочка обработки (тестирования, улучшения) результата.

Google также анонсировал агент Jules — экспериментальный агент для кодирования на базе ИИ, который напрямую интегрируется в рабочий процесс GitHub. Он может решать проблему, разрабатывать план и выполнять его, все под руководством и контролем разработчика. Эти усилия являются частью долгосрочной цели Google по созданию агентов ИИ, которые будут полезны во всех областях, включая кодирование.

Компания Anthropic [25] отмечает, что понятие «Агент» можно определить несколькими способами. Некоторые клиенты определяют агентов как полностью автономные системы, которые работают независимо в течение длительных периодов времени, используя различные инструменты для выполнения сложных задач. Другие используют этот термин для описания более предписывающих реализаций, которые следуют предопределенным рабочим процессам. Anthropic

классифицирует все эти вариации как агентские системы, но проводит важное архитектурное различие между рабочими процессами и агентами:

Рабочие процессы — это системы, в которых LLM и инструменты организованы посредством предопределенных последовательностей (путей) кода.

Агенты, с другой стороны, — это системы, в которых LLM динамически направляют свои собственные процессы и использование инструментов, сохраняя контроль над тем, как они выполняют задачи [25].

Общие черты для всех агентов:

- Пошаговое исполнение (подсказывание цепочки мыслей - chain of thoughts), которое просит LLM думать шаг за шагом
- Самосогласованность, которая побуждает модель генерировать несколько ответов и выбирать тот, который наиболее согласуется с другими
- Чередование шагов рассуждения и действия для достижения цели
- Самоуточнение, которое позволяет агенту размышлять над собственным результатом
- Рефлексия, которая позволяет модели действовать, оценивать, размышлять и повторять.

II Структура и задачи LLM агентов

A. *Ключевые компоненты*

К ключевым компонентам архитектур LLL-агентов можно отнести следующие модули.

Модуль восприятия: этот модуль отвечает за сбор информации из окружающей среды. Он может включать датчики, камеры или другие устройства ввода. Затем воспринимаемая информация обрабатывается и подается в LLM.

Большая языковая модель (LLM): ядро LLL-агента, LLM обрабатывает воспринимаемую информацию, генерирует действия и планирует будущие шаги. Он действует как «мозг» агента, обеспечивая сложные рассуждения и принятие решений.

Модуль действий: этот модуль преобразует сгенерированные действия LLM в команды, которые могут быть выполнены в реальном мире. Он может управлять актуаторами (роботизированными конечностями), отправлять сигналы другим устройствам или взаимодействовать с программными системами.

Модуль памяти: этот компонент хранит прошлый опыт, наблюдения и полученные знания. Он позволяет LLM сохранять информацию с течением времени, улучшая его способность принимать решения и приспосабливаться.

B. *Популярные типы LLL-агентов*

Рефлексные агенты: эти агенты напрямую реагируют на окружающую среду на основе простых правил. Они подходят для задач, требующих немедленного реагирования, но не имеющих сложного планирования или долгосрочных целей.

Агенты на основе моделей: эти агенты создают внутреннюю модель своей среды для прогнозирования будущих состояний. Они могут планировать заранее и принимать более обоснованные решения.

Целевые агенты: эти агенты имеют определенные цели и используют свои знания для их достижения. Они могут рассуждать о наилучшем курсе действий для достижения своих целей.

Агенты, основанные на полезности: эти агенты учитывают ожидаемую полезность или ценность различных действий и выбирают тот, который максимизирует их общее вознаграждение.

C. *Архитектурные решения*

Компания Anthropic в своей публикации о построении агентов (декабрь 2024) проиллюстрировала архитектурные шаблоны [25]. Например:

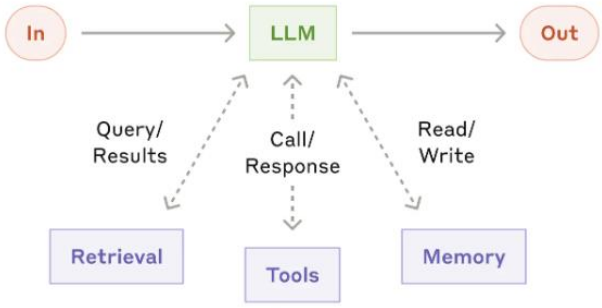

Рис.6. Дополненная LLM [25]

Базовый строительный блок агентных систем — это LLM, улучшенный такими дополнениями, как поиск, инструменты и память. Текущие модели Anthropic могут активно использовать эти возможности — генерировать собственные поисковые запросы, выбирать соответствующие инструменты и определять, какую информацию сохранять.

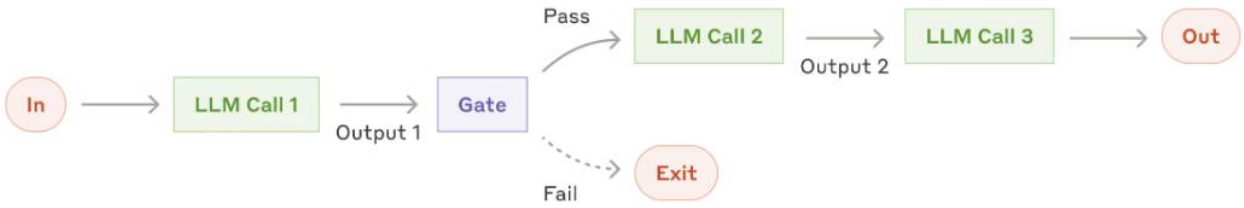

Рис. 7 Цепочка подсказок [25].

Цепочка подсказок разбивает задачу на последовательность шагов, где каждый вызов LLM обрабатывает вывод предыдущего. Вы можете добавить программные проверки (см. «gate» на схеме выше) на любых промежуточных шагах, чтобы убедиться, что процесс все еще идет по плану.

Всего в публикации [25] описано 7 подобного рода шаблонов.

D. Открытые задачи и проблемы

Безопасность и надежность: обеспечение того, чтобы агенты LLL действовали этично и надежно в сложных и непредсказуемых средах.

Объясняемость: понимание обоснования решений LLL-агентов для повышения доверия и ответственности.

Смещение данных: смягчение смещений в обучающих данных для предотвращения несправедливого или дискриминационного поведения.

Ограничения оборудования: разработка оборудования, которое может поддерживать вычислительные требования продвинутых LLL-агентов.

III Практические реализации

Здесь можно сослаться, например, на открытый код упомянутой выше статьи Anthropic [26].

Исходный код агента для создания контента для социальных сетей представлен в данном ресурсе Github [27]. Используется CrewAI.

Примеры кода для LangGraph представлены здесь [28]. Rivet [29] представляет среду для визуального программирований агентов (рис. 8)

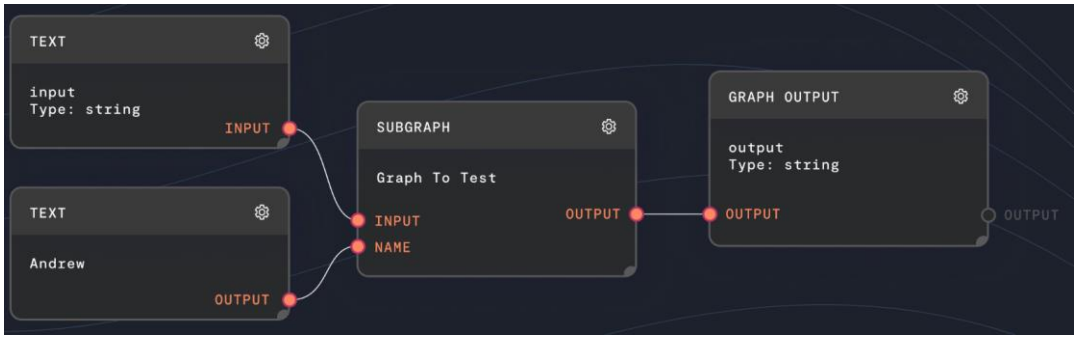

Рис. 8. Rivet [29]

Amazon Bedrock [30] — это полностью управляемый сервис, который предлагает выбор высокопроизводительных базовых моделей от ведущих компаний ИИ, таких как AI21 Labs, Anthropic, Cohere, , Meta, Mistral AI и др. через единый API, а также широкий набор возможностей, необходимых для создания приложений генеративного. Используя Amazon Bedrock, можно легко экспериментировать и оценивать лучшие базовые модели для вашего варианта использования, конфиденциально настраивать их с помощью собственных данных, используя такие методы, как тонкая настройка и расширенная генерация поиска (RAG), и создавать агентов, которые выполняют задачи, используя ваши корпоративные системы и источники данных. Поскольку Amazon Bedrock не требует серверов, то не нужно управлять какой-либо инфраструктурой.

AIDE — это агент LLM, который генерирует решения для задач машинного обучения исключительно на основе описаний задачи на естественном языке [31]. В собственном тесте, состоящем из более чем 60 соревнований Kaggle по анализу данных, AIDE продемонстрировал впечатляющие результаты, превзойдя в среднем 50% участников Kaggle.

Bee agent – интересный открытый фреймворк от IBM, написанный на TypeScript [39]. OpenAI представил текст так называемого ReAct агента [40]. Reason-then-Act (ReAct) агент итеративно рассуждает о том, как решить запрос, выполняет действие, наблюдает за результатом и определяет, следует ли предпринять другое действие или предоставить ответ.

IV Агенты и кибербезопасность

Здесь все относительно просто. Агенты скрывают прямой доступ к LLM. Для разработчиков это означает

возможность самостоятельно формулировать запросы к LLM, проверять вывод различными способами и т.д. То есть, для разработчиков, агенты – это способ повысить безопасность приложений.

Для конечных пользователей – все наоборот. Есть не очень прозрачные LLM (а при использовании агентов – еще и неизвестные LLM), для которых добавлена непрозрачная оболочка – агент. Для конечных пользователей ситуация с кибербезопасностью явно становится хуже.

Соответственно, развитие агентов ставит вопрос о расширении тестирования таких приложений. Также очевидно, что придется иметь дело с вредоносными агентами. Причем вредоносными они могут стать непосредственно в силу того, что так они проектировались специально, а также в силу того, что какие-то из используемых компонент (или отдельных агентов для многоагентных систем) стали таковыми (здесь мы будем видеть типичные атаки на цепочки поставок). Кроме того, агенты могут попросту выдавать неверные (как минимум, не оптимальные) решения. И отладка (проверка) будут сильно осложнены недетерминированностью результатов работы тех же LLM. Пример из сегодняшней практики – неверные (не оптимальные) решения по маршруту движения, которые может предлагать навигатор. Они ведь не каждый раз являются таковыми.

Одна из первых работ по безопасности ИИ-агентов [32] отмечает, что агенты, работающие на основе больших языковых моделей (LLM), могут привносить критические уязвимости безопасности. Однако существующая литература не дает всесторонней оценки атак и защиты от агентов на основе LLM. Чтобы решить эту проблему, авторы представили Agent Security Bench (ASB), комплексную структуру, предназначенную для формализации, сравнительного анализа и оценки атак и защиты агентов на основе LLM, включая 10 сценариев (например, электронная коммерция, автономное вождение, финансы), 10 агентов, нацеленных на сценарии, более 400 инструментов, 23 различных типа методов атаки/защиты и 8 метрик оценки.

V Что будет?

Как хороший пример ближайшего будущего можно привести материал, который опубликовала компания Anthropic уже в момент редактирования данной статьи. Anthropic создает своего рода интернет для ИИ-универсальную инфраструктуру, которая позволит разным ИИ системам работать вместе. Роль, которую для построения Интернет сыграл HTTP протокол, Anthropic отводит Model Context Protocol (MCP). Это открытый стандарт, который позволяет разработчикам создавать безопасные двусторонние соединения между источниками данных и инструментами на базе ИИ. Архитектура проста: разработчики могут либо предоставлять свои данные через серверы MCP, либо создавать приложения ИИ (клиенты MCP), которые подключаются к этим серверам [33].

Компания Anthropic представила свою дорожную карту [34] по развитию протокола Model Context Protocol (MCP) на первую половину 2025 года. Почему мы считаем это важным:

- Сейчас для использования, например, ChatGPT нужно подключаться к серверам OpenAI. С MCP модели будут доступны на любых серверах
- Единый формат взаимодействия с моделями позволит создавать переиспользуемые сторонние компоненты.
- Появится инфраструктура для ИИ-систем, каталоги сервисов (приложений, агентов)
- Единое решение для безопасности соединений

Благодарности

Авторы благодарны сотрудникам лаборатории Открытых информационных технологий кафедры Информационной безопасности факультета ВМК МГУ имени М.В. Ломоносова за обсуждения и ценные замечания.

Статья написана в рамках развития направления «Искусственный интеллект в кибербезопасности» на факультете ВМК МГУ имени М.В. Ломоносова [35].

Традиционно отмечаем, что все публикации в журнале INJOIT, связанные с цифровой повесткой, начинались с работ В.П. Куприяновского и его многочисленных соавторов [36-38].

Библиография

[1] Namiot, Dmitry, and Eugene Ilyushin. "On Cyber Risks of Generative Artificial Intelligence." International Journal of Open Information Technologies 12.10 (2024): 109-119.
[2] 2024: The Year of AI Agents https://pcg.io/insights/2024-year-ai-agents/ Retrieved: Dec, 2024
[3] Mudarova, Ramina, and Dmitry Namiot. "Countering Prompt Injection attacks on large language models." International Journal of Open Information Technologies 12.5 (2024): 39-48.
[4] De Stefano, Gianluca, Lea Schönherr, and Giancarlo Pellegrino. "Rag and roll: An end-to-end evaluation of indirect prompt manipulations in llm-based application frameworks." arXiv preprint arXiv:2408.05025 (2024).
[5] Autogen https://github.com/microsoft/autogen Retrieved: Dec, 2024
[6] AG2 https://github.com/ag2ai/ag2 Retrieved: Dec, 2024
[7] AutoGen: Enabling next-generation large language model applications https://www.microsoft.com/en-us/research/blog/autogen-enabling-next-generation-large-language-model-applications/ Retrieved: Dec, 2024
[8] The Leading Multi-Agent Platform https://www.crewai.com/ Retrieved: Dec, 2024
[9] Introduction to CrewAI https://www.kaggle.com/code/amansherjadakhan/introduction-to-crewai Retrieved: Dec, 2024
[10] Langchain https://www.langchain.com/ Retrieved: Dec, 2024
[11] Langgraph: multi-agent workflows https://blog.langchain.dev/langgraph-multi-agent-workflows/ Retrieved: Dec, 2024
[12] Namiot, Dmitry, et al. "Information robots in enterprise management systems." International Journal of Open Information Technologies 5.4 (2017): 12-21.
[13] Namiot, Dmitry, Vladimir Sukhomlin, and Sergey Shargalin. "On Software Agents in ERP Systems." International Journal of Open Information Technologies 4.6 (2016): 49-54.
[14] Llama Stack https://github.com/meta-llama/llama-stack Retrieved: Dec, 2024

[15] Introducing Devin, the first AI software engineer https://www.cognition.ai/blog/introducing-devin Retrieved: Dec, 2024
[16] OpenHands: Code Less, Make More https://github.com/All-Hands-AI/OpenHands Retrieved: Dec, 2024
[17] Replit Agent https://docs.replit.com/replitai/agent Retrieved: Dec, 2024
[18] Announcing v0: Generative UI https://vercel.com/blog/announcing-v0-generative-ui Retrieved: Dec, 2024
[19] Bolt https://bolt.new/ Retrieved: Dec, 2024
[20] Anthropic Claude 3.5 Sonnet https://www.anthropic.com/news/claude-3-5-sonnet Retrieved: Dec, 2024
[21] Internal agentic coding evaluation https://www-cdn.anthropic.com/fed9cc193a14b84131812372d8d5857f8f304c52/Model_Card_Claude_3_Addendum.pdf Retrieved: Dec, 2024
[22] What OpenAI ChatGPT Pro Means for AI Agents and Agentic AI https://www.teneo.ai/blog/what-openai-chatgpt-pro-means-for-ai-agents-and-agentic-ai Retrieved: Dec, 2024
[23] Awesome AI agents https://github.com/e2b-dev/awesome-ai-agents Retrieved: Dec, 2024
[24] Gemini 2.0 https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/#ceo-message Retrieved: Dec, 2024
[25] Building effective agents https://www.anthropic.com/research/building-effective-agents Retrieved: Dec, 2024
[26] Building Effective Agents Cookbook https://github.com/anthropics/anthropic-cookbook/tree/main/patterns/agents Retrieved: Dec, 2024
[27] AI Agents Tutorial https://github.com/keitazoumana/LLMs/blob/main/Multi_Agents_For_Content_Creation.ipynb Retrieved: Dec, 2024
[28] Choosing Between LLM Agent Frameworks https://readmedium.com/en/https:/towardsdatascience.com/choosing-between-llm-agent-frameworks-69019493b259 Retrieved: Dec, 2024
[29] Rivet https://rivet.ironcladapp.com/ Retrieved: Dec, 2024
[30] Amazon BedRock https://aws.amazon.com/bedrock/agents/ Retrieved: Dec, 2024
[31] AIDE: the Machine Learning Engineer Agent https://github.com/WecoAI/aideml Retrieved: Dec, 2024
[32] Zhang, Hanrong, et al. "Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents." arXiv preprint arXiv:2410.02644 (2024).
[33] Introducing the Model Context Protocol https://www.anthropic.com/news/model-context-protocol Retrieved: Dec, 2024
[34] MCP roadmap https://modelcontextprotocol.io/development/roadmap Retrieved: Dec, 2024
[35] Намиот, Д. Е. Искусственный интеллект и кибербезопасность / Д. Е. Намиот, Е. А. Ильюшин, И. В. Чижов // International Journal of Open Information Technologies. – 2022. – Т. 10, № 9. – С. 135-147. – EDN DYQWEH.
[36] Розничная торговля в цифровой экономике / В. П. Куприяновский, С. А. Синягов, Д. Е. Намиот [и др.] // International Jou rnal of Open Information Technologies. – 2016. – Т. 4, № 7. – С. 1-12. – EDN WCMIWN.
[37] Развитие транспортно-логистических отраслей Европейского Союза: открытый BIM, Интернет Вещей и кибер-физические системы / В. П. Куприяновский, В. В. Аленьков, А. В. Степаненко [и др.] // International Journal of Open Information Technologies. – 2018. – Т. 6, № 2. – С. 54-100. – EDN YNIRFG.
[38] Умная инфраструктура, физические и информационные активы, Smart Cities, BIM, GIS и IoT / В. П. Куприяновский, В. В. Аленьков, И. А. Соколов [и др.] // International Journal of Open Information Technologies. – 2017. – Т. 5, № 10. – С. 55-86. – EDN ZISODV.
[39] Bee agent https://github.com/i-am-bee/bee-agent-framework Retrieved: Dec, 2024
[40] How to create a ReAct agent from scratch https://langchain-ai.github.io/langgraph/how-tos/react-agent-from-scratch/#define-nodes-and-edge Retrieved: Dec, 2024

On Architecture of LLM agents

Dmitry Namiot, Eugene Ilyushin

***Abstract*— The paper considers the architecture of agents for Artificial Intelligence systems. The word "agent" has become the topic of 2024 for such systems. Agent-based Artificial Intelligence is considered as the next step in the development of generative models. Accordingly, agents for large language models (base models) are one of the most rapidly developing research areas. An artificial intelligence agent is a program (software) that can collect data and, based on it, independently perform tasks to achieve predetermined goals. To perform tasks, agents use one or more language models. Agents can be considered as a logical step in helping developers create workflows (implement business models) using large language (base) models. In terms of collecting (analyzing) information, agents can be compared with long-known mashups (web mashups, for example), and in terms of performing any actions - with software robots. Large language models today can contain built-in support for some workflows. An alternative is various kinds of frameworks that are designed to simplify the process of creating agents.**

***Keywords* — artificial intelligence, agents, generative models**

REFERENCES

[1] Namiot, Dmitry, and Eugene Ilyushin. "On Cyber Risks of Generative Artificial Intelligence." International Journal of Open Information Technologies 12.10 (2024): 109-119.
[2] 2024: The Year of AI Agents https://pcg.io/insights/2024-year-ai-agents/ Retrieved: Dec, 2024
[3] Mudarova, Ramina, and Dmitry Namiot. "Countering Prompt Injection attacks on large language models." International Journal of Open Information Technologies 12.5 (2024): 39-48.
[4] De Stefano, Gianluca, Lea Schönherr, and Giancarlo Pellegrino. "Rag and roll: An end-to-end evaluation of indirect prompt manipulations in llm-based application frameworks." arXiv preprint arXiv:2408.05025 (2024).
[5] Autogen https://github.com/microsoft/autogen Retrieved: Dec, 2024
[6] AG2 https://github.com/ag2ai/ag2 Retrieved: Dec, 2024
[7] AutoGen: Enabling next-generation large language model applications https://www.microsoft.com/en-us/research/blog/autogen-enabling-next-generation-large-language-model-applications/ Retrieved: Dec, 2024
[8] The Leading Multi-Agent Platform https://www.crewai.com/ Retrieved: Dec, 2024
[9] Introduction to CrewAI https://www.kaggle.com/code/amansherjadakhan/introduction-to-crewai Retrieved: Dec, 2024
[10] Langchain https://www.langchain.com/ Retrieved: Dec, 2024
[11] Langgraph: multi-agent workflows https://blog.langchain.dev/langgraph-multi-agent-workflows/ Retrieved: Dec, 2024
[12] Namiot, Dmitry, et al. "Information robots in enterprise management systems." International Journal of Open Information Technologies 5.4 (2017): 12-21.
[13] Namiot, Dmitry, Vladimir Sukhomlin, and Sergey Shargalin. "On Software Agents in ERP Systems." International Journal of Open Information Technologies 4.6 (2016): 49-54.
[14] Llama Stack https://github.com/meta-llama/llama-stack Retrieved: Dec, 2024
[15] Introducing Devin, the first AI software engineer https://www.cognition.ai/blog/introducing-devin Retrieved: Dec, 2024
[16] OpenHands: Code Less, Make More https://github.com/All-Hands-AI/OpenHands Retrieved: Dec, 2024
[17] Replit Agent https://docs.replit.com/replitai/agent Retrieved: Dec, 2024
[18] Announcing v0: Generative UI https://vercel.com/blog/announcing-v0-generative-ui Retrieved: Dec, 2024
[19] Bolt https://bolt.new/ Retrieved: Dec, 2024
[20] Anthropic Claude 3.5 Sonnet https://www.anthropic.com/news/claude-3-5-sonnet Retrieved: Dec, 2024
[21] Internal agentic coding evaluation https://www-cdn.anthropic.com/fed9cc193a14b84131812372d8d5857f8f304c52/Model_Card_Claude_3_Addendum.pdf Retrieved: Dec, 2024
[22] What OpenAI ChatGPT Pro Means for AI Agents and Agentic AI https://www.teneo.ai/blog/what-openai-chatgpt-pro-means-for-ai-agents-and-agentic-ai Retrieved: Dec, 2024
[23] Awesome AI agents https://github.com/e2b-dev/awesome-ai-agents Retrieved: Dec, 2024
[24] Gemini 2.0 https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/#ceo-message Retrieved: Dec, 2024
[25] Building effective agents https://www.anthropic.com/research/building-effective-agents Retrieved: Dec, 2024
[26] Building Effective Agents Cookbook https://github.com/anthropics/anthropic-cookbook/tree/main/patterns/agents Retrieved: Dec, 2024
[27] AI Agents Tutorial https://github.com/keitazoumana/LLMs/blob/main/Multi_Agents_For_Content_Creation.ipynb Retrieved: Dec, 2024
[28] Choosing Between LLM Agent Frameworks https://readmedium.com/en/https:/towardsdatascience.com/choosing-between-llm-agent-frameworks-69019493b259 Retrieved: Dec, 2024
[29] Rivet https://rivet.ironcladapp.com/ Retrieved: Dec, 2024
[30] Amazon BedRock https://aws.amazon.com/bedrock/agents/ Retrieved: Dec, 2024
[31] AIDE: the Machine Learning Engineer Agent https://github.com/WecoAI/aideml Retrieved: Dec, 2024
[32] Zhang, Hanrong, et al. "Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents." arXiv preprint arXiv:2410.02644 (2024).
[33] Introducing the Model Context Protocol https://www.anthropic.com/news/model-context-protocol Retrieved: Dec, 2024
[34] MCP roadmap https://modelcontextprotocol.io/development/roadmap Retrieved: Dec, 2024
[35] Namiot, D. E. Iskusstvennyj intellekt i kiberbezopasnost' / D. E. Namiot, E. A. Il'jushin, I. V. Chizhov // International Journal of Open Information Technologies. – 2022. – T. 10, # 9. – S. 135-147. – EDN DYQWEH.
[36] Roznichnaja torgovlja v cifrovoj jekonomike / V. P. Kuprijanovskij, S. A. Sinjagov, D. E. Namiot [i dr.] // International Jou rnal of Open Information Technologies. – 2016. – T. 4, # 7. – S. 1-12. – EDN WCMIWN.
[37] Razvitie transportno-logisticheskih otraslej Evropejskogo Sojuza: otkrytyj BIM, Internet Veshhej i kiber-fizicheskie sistemy / V. P. Kuprijanovskij, V. V. Alen'kov, A. V. Stepanenko [i dr.] // International Journal of Open Information Technologies. – 2018. – T. 6, # 2. – S. 54-100. – EDN YNIRFG.
[38] Umnaja infrastruktura, fizicheskie i informacionnye aktivy, Smart Cities, BIM, GIS i IoT / V. P. Kuprijanovskij, V. V. Alen'kov, I. A. Sokolov [i dr.] // International Journal of Open Information Technologies. – 2017. – T. 5, # 10. – S. 55-86. – EDN ZISODV.
[39] Bee agent https://github.com/i-am-bee/bee-agent-framework Retrieved: Dec, 2024
[40] How to create a ReAct agent from scratch https://langchain-ai.github.io/langgraph/how-tos/react-agent-from-scratch/#define-nodes-and-edge Retrieved: Dec, 2024