Семинар "Педагогика ИИ"

Дата и время: 01.10.2025, 10.00

Место проведения: НИУ «МЭИ», г. Москва, ул. Красноказарменная, д. 13с3, аудитория М-101.

Участники: Олег Вадимович Глухов (НИУ «МЭИ»), Павел Юрьевич Анучин (НИУ «МЭИ»), Шамиль Алиевич Оцоков (НИУ «МЭИ»), Эдуард Артурович Челышев (НИУ «МЭИ»), Владимир Владимирович Чистяков (НИУ «МЭИ»), Роман Сергеевич Куликов (НИУ «МЭИ»), Даниил Алексеевич Денисов (НИУ «МЭИ»).

Повестка:

1) Тема основной дискуссии «Проектирование ИИ-ассистентов инженеров»:
- Доклад «Архитектура интеллектуальных решений на стыке ASR, TTS и LLM для задач унифицированных коммуникаций»;
2) Выбор тематики следующего семинара.

Олег Вадимович Глухов:
Коллеги, добрый день. Сегодня у нас уже седьмой семинар, плавно приближаемся к десятому. Здесь собрались те, кто уже присутствовал на встречах, то есть новых участников нет. Я предлагаю для стенограммы просто кратко представиться — быстро пробежаться по именам — и перейти к основной теме обсуждения. Напомню, что это цикл семинаров по проектированию ассистентов. Сегодня будет доклад об архитектуре интеллектуальных решений на стыке ASR, TTS и LLM для задач унифицированных коммуникаций. Меня зовут Олег Вадимович Глухов, я представляю кафедру радиотехнических систем Московского энергетического института. Давайте познакомимся по часовой стрелке: сначала те, кто присутствует очно, затем — те, кто онлайн.

Владимир Владимирович Чистяков:
Здравствуйте, коллеги. Меня зовут Чистяков Владимир. Я сотрудник кафедры РТС НИУ МЭИ. Занимаюсь развитием, внедрением и интегрированием искусственного интеллекта в рабочие процессы.

Эдуард Артурович Челышев:
Добрый день, Челышев Эдуард Артурович, ассистент кафедры ВМСС НИУ МЭИ. Интересуюсь искусственным интеллектом во всех его проявлениях.

Роман Сергеевич Куликов:
Роман Куликов, директор Института радиотехники и электроники (ИРЭ) НИУ МЭИ. Я очень хочу, чтобы у нас появилось образовательное направление в этой области, и хотел бы внедрить современные инструменты искусственного интеллекта в сферу разработок в области радиотехники и электроники.

Павел Юрьевич Анучин:
Анучин Павел Юрьевич. Представляю кафедру РТС НИУ МЭИ. Занимаюсь эксплуатацией, внедрением и решением прикладных задач для инженеров.

Даниил Алексеевич Денисов: Денисов Даниил Алексеевич, магистрант первого курса института ИВТИ НИУ МЭИ.

Олег Вадимович Глухов:
Спасибо. Собственно, я уже анонсировал тему дискуссии. Доклад у нас на тему: «Архитектура интеллектуальных решений на стыке трёх технологий для задач унифицированных коммуникаций». Даниил Алексеевич, вам слово.

Даниил Алексеевич Денисов:
Да. Только, коллеги, будьте добры, подскажите: я вижу презентацию на экране, но как мы договоримся о переключении слайдов? Я буду говорить «дальше», а вы будете переключать? Всё, хорошо.
Итак, коллеги, добрый день, ещё раз представлюсь. Меня зовут Даниил Денисов. Тема моего доклада — архитектура интеллектуальных решений на стыке ASR, TTS и LLM для задач унифицированных коммуникаций. Я работаю под руководством доцента Шамиля Алиевича Оцокова.
Сегодня мы вкратце рассмотрим путь от проблем, возникающих в унифицированных коммуникациях, до конкретных архитектурных решений. Цель моего доклада — показать, где и как в задачах унифицированных коммуникаций может пригодиться искусственный интеллект. Поэтому для начала предлагаю разобрать основные проблемы, с которыми мы сталкиваемся.
Унифицированные коммуникации — это сфера, в которой создаются решения для различных способов общения: как для систем видеоконференцсвязи, так и для различных решений, связанных с телефонией. Основным направлением нашей работы является разработка контакт-центров, о чём я и хотел бы сегодня рассказать.

Олег Вадимович Глухов:
Давайте попробуем так: чтобы разбавить доклад дискуссией, мы будем периодически останавливать вас, если у вас будут вопросы, — например, после каждого слайда. Чтобы встреча не превратилась просто в доклад о результатах работы, а мы могли подискутировать.
У нас сразу возник вопрос, так как мы немного не в курсе самой темы унифицированных коммуникаций. Если объяснять простыми словами: правильно ли

я понимаю, что любые типы сигналов, которые агрегируются в контакт-центрах (текстовые, голосовые и так далее) и которые необходимо обрабатывать, называются унифицированными коммуникациями?

Даниил Алексеевич Денисов:
Да, всё верно. Сам термин говорит за себя: «коммуникации» понятны, а «унифицированные» означает, что наша задача — объединить различные каналы связи — текстовые, голосовые, видео — желательно в одном месте, в каком-то одном сервисе. Именно такими задачами мы и занимаемся.

Роман Сергеевич Куликов:
Например, планшет командира на поле боя в тактическом звене — это тоже сюда относится?

Даниил Алексеевич Денисов:
Здесь речь скорее идёт не о конкретных устройствах (планшете или телефоне), а именно о способах коммуникации. Если бы командир с тактическим планшетом общался с оперативным штабом по одной линии связи, а этот же оперативный штаб мог бы вести диалог по другой линии, то это уже можно было бы назвать унифицированной коммуникацией.

Роман Сергеевич Куликов:
Ну, например: есть сеть датчиков, которые срабатывают на сигналы, если кто-то пролетел или прошёл; есть голосовые радиостанции; есть видеопоток с беспилотника-разведчика; есть команды из вышестоящего штаба, от соседей. Всё это поступает к командиру тактического звена, и он должен принять решение.
Сейчас он делает это в уме: на одной рации он связывается с одним, на телефоне — с другим и так далее. А если всё это автоматизировать — это и есть пример унифицированной коммуникации, я правильно понимаю?

Даниил Алексеевич Денисов:
Да, всё верно. Как раз то, о чём вы сказали — «по телефону с одним, по рации с другим» — это не унифицированная коммуникация, это просто коммуникация. Унифицированная коммуникация была бы в том случае, если бы у командира было одно устройство, с помощью которого он мог бы общаться по всем протоколам связи со всеми участниками.

Олег Вадимович Глухов:
Было бы очень полезно. Я внесу небольшое дополнение. То есть получается, что мы практически обрабатываем мультимодальные данные?

Даниил Алексеевич Денисов:

Да. Задача унифицированных коммуникаций состоит именно в этом. Но сегодня я хотел бы рассказать конкретно о способе работы с голосовыми коммуникациями как об одном из направлений развития.
Сегодня поговорим о контакт-центрах. Первое, о чём стоит сказать: зачем нам внедрять сюда искусственный интеллект? Причиной этого, естественно, являются проблемы. Наверное, все звонили в контакт-центры и долго ждали ответа. Далеко не всем это нравится, точнее, не нравится никому.
В связи с этим у контакт-центров возникают определённые проблемы — «операционные боли», связанные с перегрузкой линий, длительным ожиданием ответа и временем обработки звонка. Частая проблема — несогласованность работы операторов, когда у одних одни данные, а у других — другие. Вас как клиента передают от одного специалиста к другому, все дают разную информацию, и в итоге проблема не решается.
Следующая проблема, возникающая при создании контакт-центра, — это финансовые и регуляторные риски. Обработка звонков обходится довольно дорого. Кроме того, мы сталкиваемся с большими трудностями при соблюдении нормативных требований и защите персональных данных.
Например, недобросовестный оператор может начать раскрывать персональные данные клиента, которого он обслуживал до вас. С этим тоже нужно бороться.
Далее — технологические ограничения. То, о чём я говорил: разные операторы обладают разной степенью осведомлённости. Фрагментация данных и знаний — это в том числе технологическая проблема. Она связана с тем, что далеко не во всех контакт-центрах есть единая база знаний. А если она и есть, то зачастую операторы не могут оперативно к ней обратиться. Также очень ограничена гибкость IVR-сценариев.

Роман Сергеевич Куликов:
Мы уже сторонники автоматизации. Как её проводить?

Даниил Алексеевич Денисов:
Сейчас мы всё рассмотрим. Но нужно обозначить проблемы, потому что решать мы будем именно их. Из-за всего перечисленного качество взаимодействия с клиентами падает, и клиенты недовольны.

Даниил Алексеевич Денисов:
В общем, сегодня я хотел бы рассказать об архитектуре трёх решений, которые работают на стыке ASR (автоматического распознавания речи), TTS (преобразования текста в речь) и LLM (больших языковых моделей). Эта связка открывает действительно новые возможности для автоматизации и повышения качества работы в контакт-центрах.
Итак, первое решение — это Smart IVR. Это автоматизированная система обработки запросов с использованием больших языковых моделей и собственной базы знаний. Второе решение — это ИИ-суфлёр, то есть поддержка операторов в режиме реального времени. Он выводит на экран оператора гипотезы по поводу ответа клиенту, чтобы

ускорить работу. Третье решение относится к классу Quality Management (для контроля качества). Оно позволяет суммировать диалоги и проводить проверку по скриптам.

Олег Вадимович Глухов:
Мы можем переходить к следующему слайду.

Даниил Алексеевич Денисов:
Здесь очень кратко представлена концептуальная схема решения Smart IVR. Что мы видим? Человек звонит в контакт-центр. В моем решении в качестве АТС использовался Asterisk (конечно, не всегда будет использоваться именно он, но тем не менее). Asterisk передает весь аудиотрафик в скрипт Call Orchestration. Этот скрипт передает аудио в формате WAV в ASR-сервис — сервис распознавания речи. На входе мы получаем аудио, на выходе — текст.
Далее LLM работает с текстом. Следующий сервис — TTS — преобразует текст обратно в аудио. Таким образом, на схеме видно, что используются три модели: первая — преобразование аудио в текст, вторая — преобразование текста в текст, третья — преобразование текста в аудио. Предлагаю подробнее рассмотреть эти сервисы и обсудить проблемы, возникающие при разработке такой системы. Можно перейти к следующему слайду?

Олег Вадимович Глухов:
Подождите, Даниил Алексеевич, у меня вопрос. Если это будет рассмотрено в дальнейшем, скажите, в чём здесь идея оркестратора? Почему нельзя, допустим, сразу перевести аудио в текст и отправить в LLM-сервис?

Даниил Алексеевич Денисов:
Здесь всё просто. Дело в том, что Asterisk как АТС может передавать только потоковый аудиотрафик. Но для решения данной задачи потоковая ASR-модель нам не нужна, потому что для обращения к LLM нужны минимальные задержки. Если мы будем отправлять в LLM все промежуточные гипотезы распознавания, мы просядем по производительности. Поэтому Call Orchestration сначала превращает аудиотрафик в законченную аудиозапись (фразу) и только потом отправляет её в ASR-сервис.
И обратный процесс: когда TTS-сервис генерирует аудио (чанки в формате WAV), их нужно воспроизвести в телефоне абонента, то есть преобразовать обратно в потоковый трафик. За эти задачи и отвечают Call Orchestration и Asterisk.

Олег Вадимович Глухов:
То есть смысл Smart IVR в том, что это бот, который общается с тобой голосом: он тебя понимает и отвечает.

Даниил Алексеевич Денисов:

Да. Возможно, я дал мало контекста. История в чем: обычный IVR — это робот, который общается с вами при звонке в контакт-центр. Большинство современных роботов — это так называемые rule-based боты (основанные на правилах). Их поведение прописано вручную, жестко заскриптовано. Бот не скажет ничего, кроме того, что в него заложили.
Такое решение очень ограничено. Людям, отвечающим за IVR, приходится в бесконечном итеративном режиме его улучшать, и в какой-то момент он разрастается до таких размеров, что поддерживать его вручную становится невозможно. Smart IVR на основе LLM призван решить эту проблему: мы не прописываем конкретные ответы, а позволяем модели генерировать их на основе базы знаний.

Роман Сергеевич Куликов:
Но жесткость бота гарантирует, что он не скажет лишнего. Если это поддержка банка — это чувствительная вещь, когда речь идет о деньгах. Важно, чтобы бот отвечал в жестких нормативных рамках и не вводил никого в заблуждение. А здесь берется LLM, которая может выдать непредсказуемые вещи — так называемые галлюцинации.

Даниил Алексеевич Денисов:
Действительно, это проблема, с которой мы сейчас боремся. Но из опыта могу сказать: обычные IVR на правилах всегда дают одинаковые ответы, и зачастую клиентам этого недостаточно. Я сам, когда звоню даже в банк, часто прошу соединить с оператором в обход IVR-скриптов. Собственно, эту проблему мы и хотим решить.

Роман Сергеевич Куликов:
А есть ли какие-то количественные исследования? Что вот эта гибкость LLM и имитация человеческого разговора оказываются более выигрышными, чем риск ошибки (галлюцинации)? Производились ли оценки, что ошибки не так критичны по сравнению с преимуществом человекоподобного диалога?

Даниил Алексеевич Денисов:
Таких оценок, к сожалению, не проводилось по простой причине: решения на основе LLM — это новая сфера, особенно в российских унифицированных коммуникациях, она только зарождается.

Роман Сергеевич Куликов:
А откуда берется LLM? Какую модель нужно использовать, чтобы люди не звонили просто поболтать, как по телефону доверия, занимая трафик?

Даниил Алексеевич Денисов:
Мы к этому еще придем, но вкратце: нам подойдет любая LLM из популярных бенчмарков, которую можно запустить локально. Облачные модели не подходят, так как контакт-центры работают с персональными данными, которые нельзя отправлять сторонним сервисам.

Основная проблема — ограниченность локальных моделей. Мы можем использовать только относительно небольшие модели (в районе 20–25 миллиардов параметров), которые зачастую уступают облачным гигантам в «интеллекте». Вся структура (ASR + LLM + TTS) занимает около 32 ГБ видеопамяти.

Владимир Владимирович Чистяков:
Она в целом нормально отрабатывает с такими параметрами? Понятно, что есть ошибки, но справляется хорошо?

Даниил Алексеевич Денисов:
Да, модель справляется хорошо. Во многом этому способствует «обвязка», построенная вокруг нее — мы увидим это дальше.

Олег Вадимович Глухов:
Даниил Алексеевич, еще один вопрос. Исследований по галлюцинациям не было, но в вашем решении есть какие-то фильтры, чтобы их избежать? Или это пока не прорабатывалось?

Даниил Алексеевич Денисов:
Специальных фильтров именно по галлюцинациям сейчас нет, но есть фильтр для отсечения лишнего трафика. Это как раз к вопросу о том, чтобы люди не использовали бота просто для разговоров. Обвязка LLM позволяет определить интент (намерение) клиента. Если тема не входит в нашу зону ответственности, бот выдает стандартную фразу: «Я не могу помочь с этим вопросом, пожалуйста, задайте другой».

Олег Вадимович Глухов:
Интент — это внутренний смысл фразы?

Даниил Алексеевич Денисов:
Да, интент — это намерение пользователя. Если вы звоните в банк и говорите: «Я хочу оформить дебетовую карту», то интент будет таким: «оформление дебетовой карты».

Олег Вадимович Глухов:
Хорошо, давайте двигаться дальше.

Даниил Алексеевич Денисов:
Давайте быстро разберём все три представленных модуля. Начнём с ASR-сервиса. Он отвечает за автоматическое распознавание речи (Automatic Speech Recognition). Это критически важное звено, которое преобразует речь в текст и позволяет нам в дальнейшем работать с моделями формата «текст-текст».
Что здесь важно? Во-первых, в этом решении речь анализируется целиком. Мы не можем позволить себе отправлять промежуточные гипотезы в LLM. Если бы мы генерировали гипотезы каждые 20–40 миллисекунд и отправляли их в модель, она бы

отвечала очень долго. Поэтому мы анализируем речь полностью только после того, как клиент закончит фразу.
При этом скорость работы ASR-модели должна быть высокой, поскольку общий цикл ответа не должен превышать двух секунд — иначе клиент начнёт нервничать. Речь идёт о времени от конца фразы пользователя до начала ответа робота.
Второе важное требование — определение конца фразы (endpointing). Все ASR-модели в решении имеют открытый исходный код. Мы пробовали разные варианты: и от NVIDIA, и российские модели от Vosk, и другие. Но остановились на решении от NVIDIA. В них встроено определение конца фразы: обычно 200–300 миллисекунд тишины определяются как конец высказывания.
Следующая важная функция, которая должна присутствовать в ASR-моделях, — это VAD (обнаружение голосовой активности). Это механизм, который позволяет определять тишину и не распознавать её. Дело в том, что ASR-модели обучены всегда находить речь. Даже если во входном сигнале тишина, модель пытается что-то «услышать» и зачастую вставляет наиболее часто встречающиеся слова из обучающей выборки. VAD помогает распознавать тишину и игнорировать её.
И последнее — дополнительные функции нормализации текста (приведение в порядок грамматики и пунктуации). Это очень помогает при запросах к LLM, так как модель гораздо корректнее обрабатывает грамотный текст. Мы проводим пунктуацию и нормализацию, возможно, даже классическими средствами машинного обучения, но в некоторых моделях эти функции уже заложены. Вот такие требования предъявляются к ASR-сервису в данном решении.

Олег Вадимович Глухов:
А мы рассмотрим сами модели, которые здесь применяются? Я хотел уточнить: что это? Это какие-то трансформеры? Что это за архитектура?

Даниил Алексеевич Денисов:
Это модели типа «последовательность-последовательность». Модель анализирует спектрограмму для распознавания речи и преобразует эту «энергетическую карту» в текст.

Олег Вадимович Глухов:
Но они гораздо меньше больших языковых моделей (БЯМ) и не требуют таких мощностей, верно? Это просто вспомогательный инструмент, адаптер?

Даниил Алексеевич Денисов:
Да, эти модели требуют гораздо меньше ресурсов. Например, модель, на которой я остановился, — NVIDIA Canary с 1 миллиардом параметров. Ей хватает 7 гигабайт видеопамяти, чтобы в параллельном режиме распознавать 2–3 дорожки.

Олег Вадимович Глухов:
Они требуют обучения или работают «из коробки»?

Даниил Алексеевич Денисов:
Мы используем готовые модели. Дообучение возможно, но для этого требуются большие мощности и объёмы данных, которыми я не располагаю.

Олег Вадимович Глухов:
С русской речью у них всё в порядке?

Даниил Алексеевич Денисов:
Да-да. Модели специально тестировались и подбирались именно для русской речи. Конечно, моделей много — тот же Whisper от OpenAI, уверен, вы слышали. Но на русском языке он работает хуже, чем выбранные мной модели.

Олег Вадимович Глухов:
Понятно. Хорошо. Можем двигаться дальше, если у коллег нет вопросов.

Даниил Алексеевич Денисов:
Соответственно, мы перешли к LLM-сервису. Этот сервис имеет обвязку, написанную, как правило, на Python (может быть и на других языках), но наибольший интерес представляет именно способ реализации этой обвязки.

Олег Вадимович Глухов:
Здесь представлены блок-схемы работы?

Даниил Алексеевич Денисов:
Да. Эти скрипты написаны с помощью фреймворка LangGraph. Этот фреймворк позволяет создавать конечные автоматы (state machines), по которым «ходит» клиент во время разговора. Он позволяет определить интент (намерение) и в зависимости от этого выдать ответ.
Что здесь возможно? Во-первых, запрос не по теме. Если мы звоним в контакт-центр банка и просим записать нас на приём к врачу, то ответ должен быть примерно таким: «Я не знаю ответа на ваш вопрос». Далее — RAG-поиск (Retrieval-Augmented Generation) с векторной базой знаний. Векторная база наполняется в зависимости от конкретного решения. Например, если нам нужна база знаний для медицинского центра, вполне возможно, что для её наполнения нам будет достаточно просто спарсить его сайт. Далее — механизм вызова функций. Он позволяет расширить систему до уровня, связанного, например, с денежными переводами или запросом баланса. Вызов функций — это механизм, который позволяет отправлять запросы в базу данных, API или куда-то ещё. Далее — перевод на оператора. Здесь всё просто.
И последнее — завершение разговора.
На схеме мы видим, что есть первый узел (приветственная фраза), а дальше мы перемещаемся по узлам графа и общаемся с ботом. Но здесь есть существенное ограничение: мы по-прежнему сильно зависим от скриптов. Возникает вопрос: зачем мы это сделали, если могли описать то же самое с помощью правил? Эту проблему

решают агентские системы, о которых я расскажу на следующем слайде. Но пока предлагаю остановиться на этом. Есть вопросы?

Олег Вадимович Глухов:
Коллеги, прошу, если есть вопросы.
А как определяется, какой из пяти вариантов выбрать?

Даниил Алексеевич Денисов:
Как раз по определению интента (намерения) пользователя. В конкретном решении это делается с помощью большой языковой модели. На вход подается текст и вопрос: «Что имел в виду пользователь? У меня есть такие варианты ответов/действий, скажи, в какой мне идти». Естественно, промпт структурирован.
Для подобных задач есть модели классического NLP, которые умеют определять интент по набору обучающих данных. Но как только количество интентов становится больше, скажем, 10, такой способ начинает терять в качестве и требует огромного количества данных. Поэтому здесь применяется большая языковая модель.

Роман Сергеевич Куликов:
И она лучше справляется с большой библиотекой намерений?

Даниил Алексеевич Денисов:
Намерений здесь немного, но по моим тестам даже на 80 намерениях модель «Вихрь» (Vikhrmodels) или Nemo отвечала грамотно.

Олег Вадимович Глухов:
А сколько сил надо, чтобы научить ее различать эти намерения?

Даниил Алексеевич Денисов:
Нисколько. Модели open-source. Нужна видеопамять, нужно завести модель и написать ей грамотный промпт.

Роман Сергеевич Куликов:
То есть модель взята «из коробки», из открытого источника. И она достаточно интеллектуальна, чтобы на естественном языке различать большое количество намерений?

Даниил Алексеевич Денисов:
Да. Все модели в представленных решениях — open-source (под лицензией Apache 2.0), скачаны с Hugging Face. Как правило, используются квантизованные версии.

Олег Вадимович Глухов:
Соответственно, всё работает без дообучения.

Роман Сергеевич Куликов:
А вот эти «коробочки» — RAG и вызов функции (Function Calling). У нас, конечно, мало русских терминов, много иностранных, но будем стараться переводить. Что это за блоки, кто их делает и как?

Даниил Алексеевич Денисов:
RAG (Retrieval-Augmented Generation) — это поиск по контексту. У нас есть векторная база данных (в конкретном решении использовалась российская Qdrant), заранее наполненная данными. Мы предварительно делим данные на чанки (фрагменты) и загружаем их туда. При запросе пользователя по векторному поиску находятся, скажем, топ-5 совпадений.
Эти топ-5 ответов соединяются в одну «простыню» текста, которая семантически связана с вопросом пользователя. Этим текстом мы обогащаем запрос в LLM. В системном промпте мы пишем: «При ответе учитывай следующие данные: ...» — и вставляем найденные результаты. Модель их анализирует и выдает осмысленный ответ.

Роман Сергеевич Куликов:
С этим понятно, это относительно понятная операция: главное — хорошую базу релевантно разбить. А вызов функции — это что?

Даниил Алексеевич Денисов:
Function Calling — это механизм, который встречается во всё большем количестве LLM. Он позволяет, помимо системного и пользовательского промпта, указать: «При необходимости ты можешь использовать такие-то функции». Функции заранее определены (например, на Python, как REST API или скрипт).
LLM сама решает, использовать ли инструмент. Пример: я спрашиваю баланс карты. LLM не знает, сколько у меня денег, но в списке инструментов у неё есть функция «возвратить количество денег на карте».

Роман Сергеевич Куликов:
То есть языковая модель должна разобраться, что есть набор данных, который она не знает, и вызвать соответствующую функцию?

Даниил Алексеевич Денисов:
Да, всё верно. Функция описывается контекстно: название, что она делает, входные и выходные параметры. Модели умеют грамотно составлять JSON-тела для вызова и обрабатывать ответы.

Роман Сергеевич Куликов:
Этот набор исполняемых функций должны определить специалисты или заказчик? LLM ведь не может знать априорно, как обратиться к банковской системе.

Даниил Алексеевич Денисов:
Да, набор инструментов заранее определяется и программируется, от этого никуда не уйти.

Роман Сергеевич Куликов:
Понятно. Ну и интент «перевод на оператора», думаю, всем понятен: клиент говорит «эти нейросетки — ужас какой-то, ничего не могут», и мы переводим его на человека.

Олег Вадимович Глухов:
Даниил Алексеевич, у меня вопрос. В конце шага диалога тоже сидит еще одна LLM?

Даниил Алексеевич Денисов:
Нет, «конец шага диалога» — это просто узел нашего графа в LangGraph, по которому мы возвращаемся в начало цикла обработки.

Олег Вадимович Глухов:
Я правильно понимаю, что на этой схеме LLM как функциональный блок опущена? Потому что при RAG предполагается, что контекст дополнен и идет в LLM, но на схеме этого блока нет.

Роман Сергеевич Куликов:
В каком квадратике сидит LLM?

Даниил Алексеевич Денисов:
В данных квадратиках она не нарисована, но она присутствует как в блоке RAG, так и в Function Calling. LLM используется для генерации ответа на основе входных данных (из базы знаний или результата функции).

Олег Вадимович Глухов:
То есть она распределена. По сути, здесь мы объясняем информационное взаимодействие LLM с другими блоками.

Даниил Алексеевич Денисов:
В общем-то, да. Но стоит понимать, что без LLM эта схема неработоспособна, она является сердцем системы.

Олег Вадимович Глухов:
Понятно. Коллеги, есть вопросы? Пойдём дальше.

Владимир Владимирович Чистяков:
У меня вопрос. Одинакова ли скорость отклика у Function Calling и RAG?

Даниил Алексеевич Денисов:
Нет. Ответы от векторной базы (RAG) приходят очень быстро — за 100–200 миллисекунд.
При вызове функции всё зависит от того, к чему мы обращаемся. Если к локальной базе данных — тоже быстро (десятки миллисекунд). Но если мы обращаемся к API банка, который развёрнут в другом ЦОД (например, запрос из Москвы в Питер), возможны сетевые задержки, плюс сам API может отвечать не мгновенно.
Когда я говорил о пороге в 2 секунды, я имел в виду вариант, при котором все компоненты развернуты на одном сервере (или в одной локальной сети) с минимальными задержками.

Олег Вадимович Глухов:
Даниил Алексеевич, вы сказали, что RAG работает очень быстро. Какой объём векторной базы данных (в количестве чанков) использовался?

Даниил Алексеевич Денисов:
Порядка сотен тысяч. Я полностью пропарсил сайт одной компании — это несколько тысяч веб-страниц, разбитых на чанки.

Олег Вадимович Глухов:
Для достижения такого быстродействия при большом объёме, вероятно, используются какие-то механизмы оптимизации поиска (кластеризация и т. д.), чтобы не перебирать всю базу? Или просто сравниваются векторы?

Даниил Алексеевич Денисов:
С моей стороны никаких дополнительных действий не предпринималось. Вся оптимизация происходит на стороне самой векторной базы данных (Qdrant). К сожалению, я не очень хорошо разбираюсь в алгоритмах её работы, поэтому не могу подсказать, как именно реализован индекс.

Олег Вадимович Глухов:
Хорошо. Можем переходить к следующему слайду.

Даниил Алексеевич Денисов:
Как я уже говорил, жёсткая привязка графа состояний в LangGraph — это довольно серьёзная проблема. Если мы захотим расширить функционал, нам придётся написать много новых стейтов (состояний) и переходов. По сути, мы недалеко ушли от классических IVR-скриптов. Решить эту проблему призваны так называемые агентные системы (Agents) или агенты с гибкой логикой.

Олег Вадимович Глухов:
В чём их смысл?

Даниил Алексеевич Денисов:

Мы передаём задачу определения намерений пользователя и управления диалогом на откуп LLM. Модель сама решает:
- достаточно ли данных для выполнения действия;
- нужно ли завершить звонок;
- относится ли запрос к нашей теме или он пришел «в никуда».
Мы больше не строим жёсткий граф состояний. Мы просто описываем функции-инструменты, к которым может обращаться LLM: RAG-поиск, SQL-запрос к базе данных, API-запросы и так далее.

Олег Вадимович Глухов:
И LLM как агентная система сама определяет, когда и какие инструменты использовать. То есть мы отходим от концепции графа состояний.

Даниил Алексеевич Денисов:
Да. Такая система очень гибкая, поскольку в ней нет жёсткой структуры.
Здесь я привёл в пример библиотеку SmolAgents. Она определяет поведение агента, замыкает его логику внутри себя и обладает интересным функционалом. Например, при необходимости она создаёт «песочницу» в Docker-контейнере для выполнения Python-скриптов — агент может сам писать и выполнять код.
Пример: вы звоните в медицинский центр и говорите: «Хочу записаться к терапевту». Клиник много, терапевтов тоже, а дат приёма — тем более. Непонятно, к кому именно вас записать.
В старой системе мы бы жёстко прописывали ветвление вопросов. Здесь же мы просто даём оператору функцию «Запись к врачу», которая принимает на вход ФИО, дату и данные пациента.
LLM понимает, что для вызова этой функции ей не хватает данных (даты и фамилии врача). Она может либо сама запросить недостающую информацию у пользователя, либо, например, написать код на Python для поиска свободного слота в базе данных. Таким образом, у нас исчезает понятие «состояния» диалога. Есть просто набор функций и цель, а LLM сама прокладывает путь к решению.
Преимущество — значительное повышение гибкости. Можно вести любой диалог, и система будет отвечать в рамках дозволенного (определённого промптом).
Но есть и большие риски. Во-первых, непредвиденные побочные эффекты от выполнения кода. Конечно, есть «песочница», Docker-контейнеры, но это всё равно не на 100% безопасно. Модель может «заглючить» и решить, например, удалить базу данных.
Поэтому такие системы нужно тщательно проверять и аудировать, что пока не позволяет полноценно использовать их в продуктовой среде (особенно в банках). Но направление развивается очень быстро — подобные решения есть и у Microsoft. Думаю, в ближайшем будущем появятся более надёжные механизмы безопасности.

Роман Сергеевич Куликов:
Вопрос. Откуда LLM знает, что нужно программировать? Я так и не понял.

Даниил Алексеевич Денисов:
Библиотека представляет собой обёртку для LLM, которая превращает её в агента. Агент — это сущность, которая сама принимает решение: нужно ли что-то делать, и если да, то как.

Олег Вадимович Глухов:
Как она принимает решения?

Даниил Алексеевич Денисов:
Конкретную строку кода из библиотеки я не приведу, но общий принцип таков: модель строит цепочку размышлений (Chain of Thought) или план действий.
В системном запросе мы говорим ей: «Вот запрос пользователя. Пожалуйста, составь план действий для решения этой задачи. Вот список доступных инструментов (функций) с описанием того, что они делают». И LLM генерирует план.

Роман Сергеевич Куликов:
Например, «запросить пин-код»?

Даниил Алексеевич Денисов:
Нет, функцию «запросить пин-код» мы ей просто не дадим. Мы дадим ей функцию «вернуть баланс» или «вернуть последние 4 цифры карты». Агент оперирует только тем, что мы ему предоставили.

Олег Вадимович Глухов:
Коллеги, давайте попробуем ответить на вопрос Романа Сергеевича. Насколько я понял, если мы говорим о генерации кода, то LLM сначала анализирует запрос, генерирует возможные действия на основе того, чему она обучена, и применяет их.

Роман Сергеевич Куликов:
Допустим, я звоню в МФЦ и говорю: «Запиши меня к терапевту». Она знает, что я такой-то человек. Но она знает, что в Москве 5 тысяч терапевтов в 100 поликлиниках.

Олег Вадимович Глухов:
Что она делает дальше? Куда она применяет программный код? Давайте пофантазируем. Зачем нам вообще нужно, чтобы она писала код в типовой задаче?

Даниил Алексеевич Денисов:
Хорошо, давайте на примере. Запрос: «Хочу записаться к терапевту на Солянке, дом 1». Терапевтов там много, но LLM не знает их поимённо. Ей нужно как-то получить список. Для этого она «на лету» пишет код на Python (или SQL-запрос), который обращается к базе данных и извлекает список врачей, прикреплённых к адресу «Солянка, 1».

Олег Вадимович Глухов:
То есть, если вернуться к предыдущей схеме с жёстким пайплайном: там у нас был фиксированный набор из 5 инструментов. И если они не позволяют выполнить задачу (найти терапевта по адресу), то мы в тупике.
А здесь мы разрешаем модели написать, допустим, SQL-запрос, который выполнит эту операцию. Правильно?

Даниил Алексеевич Денисов:
Да, всё верно. В контексте старого пайплайна это было бы невозможно, так как инструмента не было.

Роман Сергеевич Куликов:
Правильно ли я понимаю, что это можно трактовать как динамическое расширение банка вызываемых функций?

Олег Вадимович Глухов:
Да, именно так.

Роман Сергеевич Куликов:
На предыдущем этапе этот банк был заранее написан разработчиком. А здесь мы даём LLM возможность самостоятельно дописывать функции. Но тогда мы не можем гарантировать, что она напишет их безопасно.

Даниил Алексеевич Денисов:
Да, всё верно.

Олег Вадимович Глухов: Нужна надёжность. Ага, так понятнее, спасибо. Переходим дальше.

Даниил Алексеевич Денисов:
Мы подошли к последнему компоненту нашей схемы — сервису TTS (Text-to-Speech, синтез речи).
Что здесь важно? Одним из главных критериев остаётся естественность звучания: сохранение интонации, пунктуации и так далее. Если голос будет звучать роботизированно, это никому не понравится.
Но есть и технические требования. Сервис должен поддерживать стриминг. Первый чанк (фрагмент) аудио должен передаваться менее чем за 150–200 миллисекунд. Как это работает? LLM генерирует токены в потоковом режиме, не выдавая весь ответ сразу. Как только мы получаем от LLM первые 2–3 слова, мы сразу же отправляем их на синтез.
Это позволяет быстро озвучить начало фразы. Пока клиент слушает первые слова, LLM успевает сгенерировать остальную часть ответа, которую мы также отправляем на синтез. Таким образом, диалог проходит с минимальными задержками.

Также важны интонация, управление паузами и произношение. Раньше в сервисах синтеза речи (особенно работающих на центральном процессоре) использовались SSML-теги. В текст добавлялись специальные метки, определяющие, где нужно поставить ударение, сделать паузу, ускорить или замедлить речь.
Современные модели, работающие на графическом процессоре (примеры приведены ниже), могут сами определять нужную интонацию и грамотно воспроизводить речь без ручной разметки.
И последнее — функция Barge-in (перебивание). Если вы начали слушать ответ бота, но передумали и сказали: «Мне нужен оператор», синтез речи должен мгновенно прекратиться, а бот должен начать обрабатывать новый запрос.
В качестве примеров я привёл модели Silero TTS V4 и Vosk TTS-RU. Это русскоязычные модели, обученные на русскоязычных датасетах, которые достаточно качественно синтезируют речь. Приставка «Multi» у Vosk означает, что модель мультиязычная или мультоголосовая (в ней представлено несколько мужских и женских голосов, которые можно переключать).
Готов ответить на вопросы.

Роман Сергеевич Куликов:
Маленький вопрос. Когда мы говорим, что модуль русскоязычный, возникает нюанс: айтишники постоянно пересыпают речь английскими терминами. На каком модуле надо работать с ними?

Даниил Алексеевич Денисов:
Есть так называемые мультилингвальные модели, как для распознавания речи, так и для синтеза. Просто следует подобрать подходящую.

Роман Сергеевич Куликов:
Да, это для расширения кругозора. А модели с матерным языком есть?

Даниил Алексеевич Денисов:
В контексте синтеза речи — да, есть. Синтез речи ведь не «обучается материться», он просто умеет воспроизводить то, что ему подали на вход.

Олег Вадимович Глухов:
А если клиент начинает ругаться?

Даниил Алексеевич Денисов:
Конечно. ASR-модели тоже умеют распознавать мат и бранную речь. В больших Enterprise-системах часто есть настройка: «выключить распознавание бранных слов». По умолчанию она может быть включена или выключена.

Роман Сергеевич Куликов:
То есть такие модули в принципе есть. Спасибо.

Даниил Алексеевич Денисов:
Кстати, это довольно полезно с точки зрения следующей системы, которую мы рассмотрим. Можно переключить слайд.

Олег Вадимович Глухов:
Последний вопрос. Эти модели синтеза тоже довольно легковесные относительно LLM?

Даниил Алексеевич Денисов:
Да. Silero вообще работает на CPU и не требует больших ресурсов. Vosk может работать на GPU. В целом всё, о чем я рассказываю, помещается на персональный компьютер с хорошей видеокартой.

Олег Вадимович Глухов:
Что за видеокарта?

Даниил Алексеевич Денисов:
Например, RTX 4090 на 24 ГБ. Разработка ведется на локальном компьютере: мы поднимаем там все окружение. Такая карта позволяет хранить модель с 15–16 миллиардами параметров плюс ASR (например, NVIDIA Canary на 1 миллиард).
Сейчас мы переходим на RTX 5090 (32 ГБ) или профессиональные карты серии RTX A-series/Quadro.
Для эксплуатации решение переносится на сервер Bare Metal с виртуализацией.

Олег Вадимович Глухов:
А чем вызвана необходимость эксплуатации именно на мощном сервере? Сама эксплуатация требует больше вычислительных ресурсов?

Даниил Алексеевич Денисов:
Во-первых, каждый параллельный запрос требует ресурсов. У нас контакт-центр, куда звонят не раз в сутки, а десятки или сотни тысяч раз. Нам нужна параллельная обработка.

Олег Вадимович Глухов:
Понятно. Если пользователь один, то ресурсов ПК достаточно.

Даниил Алексеевич Денисов:
Да, всё верно. Если убрать специфику контакт-центра, то по этим схемам можно сделать персонального ассистента на домашнем компьютере.

Олег Вадимович Глухов:
Хорошо. Давайте дальше. Спасибо.

Даниил Алексеевич Денисов:
Да, перейдём к последнему осмысленному слайду.
Как я и сказал, мы рассматриваем три вида работы с LLM в контексте контакт-центров. Один из них мы разобрали подробно (Smart IVR), а два других рассмотрим поверхностно, просто чтобы показать различия в требованиях к ресурсам и моделям.
Сначала рассмотрим пример ИИ-суфлёра. Его структурная схема показана по центру и справа.
Что здесь происходит? Клиент разговаривает с оператором, идёт двухканальная запись. Мы снимаем трафик и по протоколу, например gRPC (для быстродействия), отправляем его чанками по 20–40 миллисекунд в сервис обработки.
Этот сервис взаимодействует со стриминговой ASR-моделью. В отличие от предыдущей схемы, где ASR работал в автономном режиме (анализировал всю запись целиком), здесь используется стриминговая модель, которая принимает чанки, склеивает их и для каждого нового фрагмента выдаёт новую гипотезу распознавания. Эти гипотезы отправляются в Retrieval-сервис, который (в зависимости от задачи) может быть связан как с LLM, так и напрямую с векторной базой данных.
При взаимодействии с LLM мы готовы выдать оператору полностью сгенерированный ответ, чтобы он просто зачитал его клиенту. Либо, чтобы сэкономить время и ресурсы (поскольку LLM требует дополнительных затрат), мы можем извлечь первую подсказку напрямую из векторной базы знаний и сразу отправить её оператору для уточнения контекста.
Для чего это нужно? Это ускоряет работу. Часто, когда мы звоним в контакт-центр, оператор говорит: «Подождите, я уточню информацию», — и это может занять десятки минут. Суфлер решает эту проблему, быстро предоставляя нужные данные.
Слева представлен сервис Quality Management (контроль качества). Мы отправляем зеркалированный трафик на сервер записи, который сохраняет его в виде WAV-файла и передает на сервер транскрибации. Здесь уже можно использовать офлайн-модель автоматического распознавания речи (стриминг не важен, транскрибация происходит в отложенном режиме).
Полученный текст передается в сервисы проверки качества и суммирования. Они обогащают текст промтами и отправляют в LLM.
Первый сервис проверяет работу оператора: следовал ли он сценарию, был ли доброжелателен, поздоровался ли и попрощался ли.
Второй сервис (суммирование) протоколирует звонок. Все выявленные ошибки вместе с кратким содержанием диалога отправляются менеджеру по качеству — сотруднику, который обучает операторов и улучшает клиентский сервис.
На этой схеме видно, что под разные задачи требуются разные решения: где-то нужна стриминговая ASR, где-то офлайн; где-то нужен TTS, где-то нет; где-то достаточно простого поиска по базе, а где-то нужен RAG.
Есть ли вопросы по этой схеме?

Роман Сергеевич Куликов:
А зачем тут нужен оператор?

Даниил Алексеевич Денисов:
Вопрос хороший. Ответ простой: к сожалению, клиенты (и мы с вами) пока не готовы общаться только с ботами.

Роман Сергеевич Куликов:
Если это заскриптованный «тупой» бот — я понимаю почему. Мы не хотим с ними общаться. А если бот человекоподобный?

Даниил Алексеевич Денисов:
Научного ответа у меня нет, есть только гипотеза. Когда вы звоните, вы заранее не знаете, какой там бот — старый скриптованный или новый умный на LLM. Поэтому по старой привычке люди сразу просят оператора.
А для бизнеса главная метрика — оценка клиентского опыта (CSAT). Если мы уберем операторов и заставим людей говорить с ботами (даже умными), оценки упадут, и контакт-центр пострадает.

Роман Сергеевич Куликов:
Зачем тогда вообще внедрять ботов, если людям нужны люди? Бизнес — это ведь оптимум между удовлетворенностью и затратами.

Даниил Алексеевич Денисов:
Согласен. Боты решают проблему высокой загруженности (о чем я говорил в начале). Держать огромный штат операторов дорого.

Роман Сергеевич Куликов:
То есть, если выбирать между ожиданием в 10 минут и общением с ботом, человек выберет бота?

Даниил Алексеевич Денисов:
Да.

Олег Вадимович Глухов:
Плюс бот всё-таки обрабатывает часть заявок. Даже простые боты на основе правил в банках обрабатывают около 20% обращений. Это статистика, которую я наблюдал.

Роман Сергеевич Куликов:
Думаю, если бы боту разрешили рассказывать анекдоты, процент бы сильно вырос. Я сейчас без шуток. Меня и моих знакомых раздражает рафинированность, механистичность. Люди сидят как биороботы и говорят по сценарию — иногда хочется просто... ну, очень эмоционально отреагировать.

Даниил Алексеевич Денисов:
Работа над эмоциональностью и эмпатией роботов ведётся. Но пока существенных результатов нет.
Кроме того, голос должен зависеть от задачи. Банковский помощник может быть весёлым и беззаботным. А вот робот коллекторского агентства с радостным голосом: «Сообщаем, что у вас есть задолженность!» — будет звучать странно. Для таких случаев подбираются специальные интонации.

Роман Сергеевич Куликов:
Интересно, что мы с лёгкостью общаемся с текстовыми поисковиками — в них нет ничего живого, и нас это не волнует. А голос вызывает раздражение.
С другой стороны, когда появился ChatGPT, многие мои друзья чуть с ума не сошли от общения с ним. А те, у кого есть голосовые помощники вроде «Алисы», быстро привыкают. У меня есть приятель, артист, который много ездит за рулём. Он постоянно материт «Алису» — у него такой способ социализации. Она просыпается от слова «кулисы» (он часто его употребляет), спрашивает: «Что вам нужно?», а он в ответ произносит тираду.
То есть ниши, в которых люди привыкли к роботам, уже есть. Почему в колл-центрах этот барьер до сих пор не преодолён?

Даниил Алексеевич Денисов:
Думаю, дело в изначально негативном опыте, который накопился у людей за годы общения с плохими телефонными ботами. Со временем мы справимся с этой проблемой.

Роман Сергеевич Куликов:
Нужно просто брать плату за общение с человеком. Хочешь оператора — плати.

Олег Вадимович Глухов:
Да, идея хорошая.

Роман Сергеевич Куликов:
Тогда они сразу начнут решать вопросы с роботом, скрепя сердце.

Олег Вадимович Глухов:
Итак, Даниил Алексеевич, давайте продолжим.

Даниил Алексеевич Денисов:
Да, можно переходить к заключению. В заключение хочу сказать, что большие языковые модели в унифицированных коммуникациях активно применяются, практики их использования развиваются с каждым днем.

Все представленные решения требуют различных наборов моделей, работающих с разными входными данными (аудио, текст) и генерирующих соответствующий выход. Есть модели, которые работают даже с видео.

Роман Сергеевич Куликов:
Исходя из комментариев, я так понял, что у вас уже есть какие-то прототипы или даже работающие сборки. А на них можно где-то поглядеть в действии, попробовать пообщаться?

Даниил Алексеевич Денисов:
В свободном доступе их, думаю, пока нет. Дело в том, что разработка ведется в закрытом контуре.

Олег Вадимович Глухов:
Это связано с законом?

Даниил Алексеевич Денисов:
Нет. Данные решения — это мои наработки, но я также работаю в этой сфере, и там решения покрыты NDA. А мои личные наработки пока никуда не выведены, они крутятся локально.

Роман Сергеевич Куликов:
А если мы зайдем в кабинет, сдав телефоны на входе, и просто посмотрим, как это выглядит вживую? В режиме «выставки» — не с собой забрать, а у вас на компьютере посмотреть?

Олег Вадимович Глухов:
Ну, если это интересно, можно подумать об этом.

Даниил Алексеевич Денисов:
Можно попробовать записать демо, которое я мог бы вам выслать, чтобы вы послушали, как это работает. Посмотреть вживую, боюсь, не получится, потому что стенд находится в контуре компании, и доступ туда сторонним лицам запрещен.

Роман Сергеевич Куликов:
А аналогичную сборку можно собрать на каком-то внешнем компьютере?

Даниил Алексеевич Денисов:
Физически можно. Но для обработки реального голосового трафика нам нужна АТС (автоматическая телефонная станция) и шлюзы.

Роман Сергеевич Куликов:
Зачем нужна АТС?

Олег Вадимович Глухов:
Ну, чтобы звонить.

Даниил Алексеевич Денисов:
Хотя, в принципе, я могу говорить в микрофон напрямую, без телефонии. Такое гипотетически можно сделать.

Роман Сергеевич Куликов:
Понятно. Я просто сразу начинаю фантазировать: такие штуки могли бы найти применение не только в колл-центре. Нам было бы интересно попробовать внедрить их экспериментально в учебный процесс и одновременно поучить студентов осваивать этот стек технологий.

Даниил Алексеевич Денисов:
В целом, да, задача интересная: сделать из этого интеллектуального ассистента-агента, который мог бы помогать в учебном процессе. Например, с контрольными работами.

Роман Сергеевич Куликов:
Некоторые хотят, чтобы роботы читали лекции, но я скептически к этому отношусь — контакт с учителем важен. Но попробовать автоматизировать рутину стоит. Сейчас остро стоит вопрос повышения производительности труда преподавателей.
Есть рутинные процедуры (например, промежуточная проверка сотен студентов), которые занимают треть времени и никакой творческой радости не приносят. Их можно было бы автоматизировать.

Даниил Алексеевич Денисов:
Мы с научным руководителем, Шамилем Алиевичем, рассматривали вариант продолжения работы именно в области создания учебного ассистента. Может быть, действительно, что-то такое выйдет.

Роман Сергеевич Куликов:
В некоторых дисциплинах сейчас экспериментируют с LLM для проверки текстовых заданий. А если бы LLM можно было подключить в режиме незаскриптованного голосового диалога — это было бы круто.

Олег Вадимович Глухов:
Плюс перспективным видится направление цифровых координаторов (мы вчера с Владимиром обсуждали). Представьте: руководитель проекта идет по улице и надиктовывает задачи голосовому ассистенту.

Роман Сергеевич Куликов:
Пока плохо себе это представляю... Но ладно.
Вопрос: можно ли обращаться к вам за консультациями, если мы со студентами попробуем соорудить такую штуку?

Даниил Алексеевич Денисов:
Вполне. Я, к сожалению, не оставил контактов на слайде, но могу поделиться ими через организаторов.

Олег Вадимович Глухов:
Мы добавим вас в группу семинара. Думаю, совместное участие в дальнейших встречах будет полезно всем.

Даниил Алексеевич Денисов:
Да, безусловно, это интересно.

Роман Сергеевич Куликов:
У нас большая цель — научиться применять современные методы ИИ в прикладных задачах, особенно в сфере радиотехники и электроники. Нас интересует стык информационных технологий и предметных знаний.
Хочется автоматизировать рутину, чтобы человек ставил задачи и контролировал результат, а вычисления делала машина. И возможность диалога с техникой на естественном языке была бы очень привлекательной.

Даниил Алексеевич Денисов:
Да, связать голосовое управление с любой предметной областью — это крутая задача.

Олег Вадимович Глухов:
Даниил Алексеевич, мы можем опубликовать презентацию на портале МЭИ на странице семинара?

Даниил Алексеевич Денисов:
Думаю, да.

Олег Вадимович Глухов:
В целом, хотел бы вынести на обсуждение тему семинаров. Мы ведем цикл по проектированию ИИ-ассистентов инженеров. Но данный доклад показал, что мы затрагиваем и вопросы автоматизации.
Я бы предложил расширить тематику: не просто «проектирование ассистентов», а «автоматизация с применением методов ИИ». Коллеги, есть мнения?

Роман Сергеевич Куликов:
Как сейчас называется семинар?

Олег Вадимович Глухов: «Педагогика искусственного интеллекта».

Роман Сергеевич Куликов:
Мы задумывали его как обсуждение современных подходов к повышению производительности труда. Для меня, как директора института, важна автоматизация труда и инженеров, и педагогов.

Олег Вадимович Глухов:
Я к тому, что ассистенты и автоматизация идут вплотную. В моем понимании, ассистент — это диалог и неопределенность входных данных (как говорил Даниил Алексеевич: множество смысловых подтекстов). А классическая автоматизация часто работает с заранее известным диапазоном значений.

Роман Сергеевич Куликов:
Не согласен. Автоматизация возможна там, где ты можешь описать контур задачи и подобрать инструмент. Это не обязательно жесткий алгоритм. Но творческие задачи действительно пока не автоматизируются.

Олег Вадимович Глухов:
Если у коллег есть предложения по темам следующих семинаров — сообщайте. Если нет — предложения поступят от меня.
Тогда давайте заканчивать. Всем спасибо за участие.