ЛЕКЦИЯ 12
БОЛЬШИЕ ЯЗЫКОВЫЕ МОДЕЛИ (LLM)

Цель лекции

Изучить понятие и принципы работы больших языковых моделей, архитектуру Transformer и механизм внимания, этапы токенизации, обучения и генерации текста; рассмотреть семейства современных LLM, локальные модели, основные сценарии применения, ограничения, риски и требования безопасного использования больших языковых моделей.

Понятие большой языковой модели

Large Language Model (LLM) — большая языковая модель, обученная на значительных объёмах текстовых и других данных и способная работать с последовательностями естественного языка. В загруженном учебном пособии тема LLM выделена в отдельную главу после генеративного искусственного интеллекта и включает Transformer, GPT, Gemini, Claude и локальные LLM.

LLM могут использоваться для генерации и преобразования текста, ответов на вопросы, объяснения понятий, перевода, анализа документов и поддержки программирования. При этом связность и уверенный стиль ответа сами по себе не гарантируют фактическую правильность результата.

Упрощённая схема обработки текста в LLM

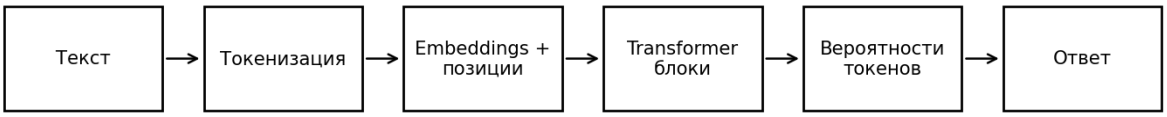


Рисунок 1 – Упрощённый процесс работы большой языковой модели

Связь NLP, генеративного ИИ и LLM

Понятие	**Назначение**	**Пример**
NLP	Обработка, анализ и понимание естественного языка	Классификация текста, NER, перевод
Generative AI	Создание нового контента	Текст, изображения, аудио, видео
LLM	Большая модель для работы с языком и последовательностями	Диалог, суммаризация, анализ документов, код

LLM находятся на пересечении NLP, глубокого обучения и генеративного ИИ. Однако не всякая NLP-модель является LLM, и не всякий генеративный ИИ является языковой моделью.

Языковое моделирование

Основная идея языкового моделирования состоит в оценке вероятности последовательностей языка. В авторегрессионной постановке модель по предыдущему контексту оценивает вероятность следующего токена.

Например, после фразы «Машинное обучение является» модель вычисляет распределение вероятностей возможных продолжений. После выбора следующего токена он добавляется к контексту, и процедура повторяется.

Таким образом, длинный ответ создаётся последовательной генерацией токенов, а не извлечением целого готового абзаца из базы заранее написанных ответов.

Токенизация

Перед обработкой текст преобразуется в последовательность токенов. Токен может соответствовать слову, части слова, знаку пунктуации или другому фрагменту. Затем каждому токену сопоставляется числовой идентификатор.

Уровень	**Пример**	**Особенность**
Слово	искусственный | интеллект	Просто, но большой словарь и проблема неизвестных слов
Подслово	искусств | енный	Позволяет работать с редкими и новыми словами
Символ	и | н | т | е | л | л | е | к | т	Малый словарь, но длинные последовательности

Современные LLM обычно используют подсловную токенизацию. Поэтому число токенов не совпадает строго с числом слов и может различаться между языками.

Embeddings

После токенизации дискретные идентификаторы преобразуются в векторные представления — embeddings. Вектор позволяет нейронной сети работать с токеном как с набором числовых признаков.

В отличие от классического TF-IDF, embedding является плотным представлением. В Transformer итоговое представление токена изменяется с учётом контекста, поэтому смысл одного и того же слова может зависеть от соседних элементов.

Почему нужен порядок токенов

Механизм внимания сам по себе не задаёт порядок элементов последовательности. Поэтому Transformer должен получать информацию о позиции токена. Для этого используются позиционные представления или другие механизмы кодирования позиции.

Это принципиально важно: предложения «модель анализирует данные» и «данные анализируют модель» содержат сходный набор слов, но имеют разную структуру и смысл.

Архитектура Transformer

Transformer — архитектура нейронной сети, основанная на механизме внимания. Она стала фундаментом многих современных языковых моделей. В отличие от классических RNN, Transformer способен непосредственно учитывать связи между разными позициями последовательности.

Ключевые элементы Transformer включают представления токенов, позиционную информацию, механизм внимания, полносвязные преобразования, нормализацию и остаточные соединения.

Компонент	**Назначение**
Token embedding	Числовое представление токена

Position information	Учет порядка элементов
Self-Attention	Связывание токена с другими токенами контекста
Feed-Forward Network	Нелинейное преобразование представлений
Residual connection	Передача информации через глубокую сеть
Normalization	Стабилизация вычислений и обучения

Механизм Self-Attention

Self-Attention позволяет модели определить, какие элементы текущего контекста наиболее важны для формирования представления конкретного токена. Для вычисления внимания используются представления Query (Q), Key (K) и Value (V).



Рисунок 2 – Концептуальная схема механизма внимания

Упрощённо соответствие Query и Key определяет вес внимания, а затем значения Value объединяются с учётом полученных весов. Масштабированное скалярное внимание обычно записывается как:

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k})V$$

Формула показывает общий принцип: сначала оценивается сходство запросов и ключей, затем веса нормализуются функцией softmax и используются для взвешивания значений.

Multi-Head Attention

Вместо одного механизма внимания Transformer использует несколько голов внимания. Каждая голова может обучаться выделять различные отношения между элементами последовательности.

Например, одна голова может сильнее учитывать локальные связи, другая — дальние зависимости, третья — определённые структурные отношения. Результаты нескольких голов объединяются и преобразуются далее.

Encoder и Decoder

Архитектурный вариант	**Особенность**	**Типичные задачи**
Encoder-only	Формирует контекстные представления входа	Классификация, извлечение признаков, NER
Decoder-only	Авторегрессионно генерирует продолжение	Генерация текста, диалог, код
Encoder–Decoder	Отдельно кодирует вход и генерирует выход	Перевод, преобразование последовательностей

Конкретная большая языковая модель может использовать один из этих вариантов или модифицированную архитектуру. Поэтому термин LLM не означает одну единственную схему сети.

Обучение больших языковых моделей

Обучение LLM обычно включает предварительное обучение на больших массивах данных. Модель решает задачу предсказания или восстановления элементов текста и постепенно формирует статистические и семантические представления языка.

После предварительного обучения модель может дополнительно адаптироваться к выполнению инструкций и диалоговым сценариям. Конкретные процедуры различаются между разработчиками и моделями.

Этап	**Смысл**
Pretraining	Обучение общим закономерностям на больших данных
Instruction tuning	Адаптация к выполнению инструкций
Preference alignment	Настройка поведения по предпочтениям и требованиям
Domain adaptation	Адаптация к специализированной предметной области

Параметры модели

Параметры нейронной сети — числовые коэффициенты, которые изменяются в процессе обучения. В большой языковой модели число параметров может быть очень большим, однако размер модели сам по себе не определяет качество.

На итоговые возможности влияют архитектура, данные, процедура обучения, качество настройки, длина контекста, вычислительная инфраструктура и методы применения модели.

Контекстное окно

Контекстное окно определяет объём последовательности, который модель может учитывать в рамках одного взаимодействия. В контекст могут входить системные инструкции, запрос пользователя, предыдущие сообщения и предоставленные документы.

Чем больше контекст, тем больше информации потенциально можно обработать одновременно, но наличие текста в контексте ещё не гарантирует, что модель одинаково точно использует каждую его часть.

Генерация ответа

После обработки контекста модель формирует распределение вероятностей следующего токена. Затем выбирается токен и процедура повторяется. На выбор могут влиять параметры генерации.

Параметр/подход	**Влияние**
Более детерминированный выбор	Повышает повторяемость
Temperature	Изменяет распределение вероятностей и разнообразие
Top-k / top-p	Ограничивает множество рассматриваемых продолжений
Ограничение длины	Контролирует максимальный объём генерации

Параметры генерации не добавляют модели новых знаний. Они изменяют способ выбора продолжения из уже сформированного моделью распределения.

GPT

GPT рассматривается как одно из семейств больших языковых моделей. GPT-модели основаны на Transformer-подходе и ориентированы на генеративную работу с последовательностями.

Такие модели могут выполнять широкий круг текстовых задач через единый интерфейс инструкций: генерацию, преобразование, суммаризацию, объяснение, извлечение структурированной информации и работу с программным кодом.

Gemini

Gemini включён в загруженное учебное пособие как пример современной LLM. При изучении конкретных коммерческих семейств важно разделять общие принципы LLM и особенности отдельного продукта, поскольку версии, возможности и условия доступа со временем изменяются.

Для учебного курса важнее понимать общую архитектурную логику: работа с контекстом, токенами, представлениями, вниманием, генерацией и мультимодальными входами.

Claude

Claude также указан как пример семейства современных больших языковых моделей. Его целесообразно рассматривать в сравнительном контексте вместе с другими LLM, не привязывая фундаментальные понятия курса к конкретной версии сервиса.

С практической точки зрения при выборе любой внешней LLM необходимо учитывать возможности работы с документами, контекстом, кодом, требования конфиденциальности, доступность API и правила обработки данных.

Локальные LLM

Локальная LLM разворачивается на инфраструктуре пользователя или организации, а не используется только через внешний облачный сервис. В учебном пособии локальные LLM выделены как отдельный элемент темы.



Рисунок 3 – Облачный и локальный варианты использования LLM

Критерий	**Облачная модель**	**Локальная модель**
Развёртывание	Инфраструктура поставщика	Инфраструктура пользователя/организации
Контроль данных	Зависит от условий сервиса	Можно обеспечить больший локальный контроль
Ресурсы пользователя	Обычно ниже	Могут требоваться значительные CPU/GPU и память
Обновления	Управляет поставщик	Управляет владелец развёртывания

Настройка	Зависит от сервиса	Возможна глубокая техническая настройка

Почему локальная модель не всегда лучше

Локальное размещение может быть полезно при повышенных требованиях к контролю данных, но оно создаёт собственные задачи: обеспечение вычислительных ресурсов, обновление модели, управление доступом, мониторинг, защита инфраструктуры и оценка качества.

Поэтому выбор между облачной и локальной LLM должен определяться требованиями задачи, политикой организации, чувствительностью данных, стоимостью и доступной инфраструктурой.

LLM и мультимодальность

Современная языковая модель может быть частью мультимодальной системы, способной работать не только с текстом. Вход может включать изображения, документы, аудио или другие представления, а результат — текст или иной тип контента.

Мультимодальность расширяет круг задач: анализ схем, изображений и документов, ответы по визуальным данным, объединение текста и изображений. Однако качество необходимо оценивать отдельно для каждой модальности.

Основные сценарии применения LLM

Задача	**Пример**
Суммаризация	Краткое содержание документа
Извлечение информации	Выделение сущностей и реквизитов
Классификация	Категоризация обращения по инструкции
Перевод	Преобразование текста между языками
Вопрос–ответ	Объяснение материала или ответ по документу
Программирование	Объяснение, создание и преобразование кода
Структурирование	Преобразование текста в таблицу или JSON-подобную структуру

LLM в информационной безопасности

В загруженных материалах по кибербезопасности отмечаются возможности LLM для анализа больших объёмов текстовой информации, контекстного анализа описаний инцидентов и адаптации к специализированным задачам.

Сценарий	**Возможная роль LLM**	**Необходимый контроль**
Анализ описаний инцидентов	Суммаризация и категоризация	Проверка аналитиком
Обработка журналов в текстовой форме	Структурирование и объяснение	Не передавать секретные данные без разрешения
Анализ угроз	Извлечение сущностей и связей	Сверка с первичными источниками
Поддержка SOC	Черновик отчёта или объяснения	Не заменяет технические средства обнаружения
Работа с кодом	Объяснение и ревью фрагмента	Тестирование и анализ безопасности

Ограничения LLM

Большая языковая модель не является гарантированно достоверной базой знаний. Она может формировать фактически неверные сведения, ошибочные ссылки или неправильные выводы. В загруженной лекции 1 отдельно подчёркивается, что убедительно звучащий ответ может быть фактически неправильным.

Другие ограничения связаны с зависимостью от качества данных, ограниченностью контекста, различиями качества между языками и предметными областями, вычислительной стоимостью и трудностью объяснения отдельных результатов.

Галлюцинации

Галлюцинацией называют правдоподобный, но неподтверждённый или неверный результат модели. Особенно критичны галлюцинации при работе с научными ссылками, нормативными документами, числами, медицинской, юридической и другой значимой информацией.

Основная мера — не считать генерацию доказательством. Значимые факты необходимо проверять по первичным или надёжным источникам.

LLM и конфиденциальность

Перед использованием внешней LLM необходимо определить, разрешено ли передавать конкретные данные поставщику сервиса. Пароли, ключи, закрытые документы, персональные данные, внутренний код и журналы безопасности нельзя автоматически помещать в запрос без соответствующего разрешения и защиты.

Для корпоративной системы должны быть определены правила доступа, журналирования, хранения запросов, обработки результатов и допустимые категории информации.

Prompt и LLM

Результат LLM зависит не только от модели, но и от инструкции. Хорошо сформулированный запрос уточняет задачу, контекст, ограничения и ожидаемый формат.

Однако подробный Prompt не устраняет фундаментальные ограничения модели и не гарантирует фактическую правильность. Методы Zero-Shot, Few-Shot и другие способы построения запросов рассматриваются в следующей отдельной теме Prompt Engineering.

Fine-tuning и адаптация

Если базовая модель недостаточно соответствует предметной области, её можно адаптировать. Один из подходов — дополнительное обучение на специализированных данных. Другой — настройка поведения через инструкции и примеры.

Подход	**Что изменяется**	**Когда полезен**
Prompting	Только входная инструкция	Быстрая настройка конкретного запроса
Few-shot examples	В контекст добавляются примеры	Нужно показать формат или способ решения
Fine-tuning	Изменяются параметры или часть параметров модели	Нужна более постоянная специализация
External knowledge	Модель получает релевантные внешние данные в контексте	Нужна работа с актуальной или внутренней информацией

LLM и внешние знания

Параметрические знания модели ограничены обучением и не должны рассматриваться как гарантированно актуальная база данных. Для задач, где важны конкретные документы или свежая информация, системе необходимо предоставить релевантный источник и требовать опоры на него.

Архитектуры, объединяющие поиск по внешнему хранилищу с генерацией ответа, позволяют связывать LLM с документами организации. Детальное проектирование таких систем требует отдельного рассмотрения качества поиска, доступа к данным и проверки цитирования.

Оценка качества LLM

Критерий	**Что проверяется**
Корректность	Соответствует ли ответ фактам и источникам
Релевантность	Отвечает ли результат поставленной задаче
Полнота	Не пропущены ли важные аспекты
Следование инструкции	Соблюдены ли формат и ограничения
Устойчивость	Насколько меняется качество на разных примерах
Безопасность	Не раскрывает ли модель защищённые данные и не создаёт ли нежелательный результат

Для прикладной LLM-системы следует создавать тестовый набор, отражающий реальные сценарии использования. Оценивать модель только по нескольким удачным демонстрационным запросам недостаточно.

Сравнение подходов к работе с языком

Критерий	**Классический NLP**	**LLM**
Признаки	TF-IDF, вручную выбранные признаки	Контекстные представления
Тип задач	Обычно отдельная модель на задачу	Одна модель может выполнять разные задачи по инструкции
Ресурсы	Обычно ниже	Обычно выше
Контроль поведения	Часто проще	Требует тщательной оценки
Малые специализированные задачи	Классические методы могут быть достаточны	LLM не всегда экономически оправдана
Генерация связного текста	Ограничена	Одна из сильных сторон

Практический пример: анализ документа

Пусть необходимо получить структурированное резюме внутреннего документа. Сначала определяется, разрешено ли использовать выбранную LLM для данного типа информации. Затем документ передаётся модели целиком или релевантными фрагментами, формулируется инструкция с требуемыми разделами и ограничением не добавлять сведения, которых нет в источнике.

После генерации результат сравнивается с исходным документом. Факты, даты, суммы, имена и выводы проверяются. Только после такой проверки текст может использоваться как рабочий результат.

Практический пример: LLM для SOC

LLM может получить разрешённое текстовое описание инцидента и подготовить краткую структурированную сводку: что произошло, какие активы затронуты, какие признаки наблюдаются и какие сведения требуют проверки. Это снижает объём ручной работы с текстом.

Но модель не должна самостоятельно объявлять инцидент подтверждённым только на основании языковой генерации. Итоговый вывод должен опираться на журналы, телеметрию, средства мониторинга и проверку специалиста.

Основные выводы

LLM — большие нейронные языковые модели, способные выполнять широкий круг задач с естественным языком.

Transformer и механизм внимания лежат в основе многих современных LLM.

Текст преобразуется в токены и векторные представления, после чего модель учитывает их контекстные связи.

Авторегрессионная LLM формирует ответ последовательным прогнозированием токенов.

GPT, Gemini и Claude представлены в учебном пособии как примеры современных семейств LLM, а локальные LLM — как отдельный вариант развёртывания.

LLM могут применяться к документам, программированию, NLP-задачам и задачам информационной безопасности.

LLM способны формировать правдоподобные ошибки, поэтому значимые результаты требуют проверки.

Выбор облачной или локальной модели должен учитывать конфиденциальность, ресурсы, безопасность и требования организации.

Контрольные вопросы

1. Что такое большая языковая модель и как связаны LLM, NLP и генеративный искусственный интеллект?
2. Как текст преобразуется в токены и embeddings и почему модели необходима информация о позиции токенов?
3. В чём состоит архитектурная идея Transformer и какую роль выполняет механизм Self-Attention?
4. Чем отличаются encoder-only, decoder-only и encoder–decoder архитектуры?
5. Как проходит предварительное обучение LLM и чем оно отличается от последующей адаптации модели?
6. Что такое контекстное окно и как авторегрессионная модель формирует ответ?
7. Какие общие особенности семейств GPT, Gemini и Claude важно учитывать при изучении LLM?
8. Чем локальная LLM отличается от облачной и какие преимущества и ограничения имеет локальное развёртывание?
9. Какие задачи могут решать LLM в информационной безопасности и почему необходим контроль специалиста?
10. Какие риски связаны с галлюцинациями, конфиденциальностью и фактической достоверностью результатов LLM?

Список источников

1. Учебное пособие «Основы искусственного интеллекта». Глава 12 «Большие языковые модели»: Transformer, GPT, Gemini, Claude, локальные LLM.

2. Беликова Е.К., Гудилина И.С., Круглова М.А., Саратовская Л.Б., Ямпольская А.В. English Reader in Artificial Intelligence = Хрестоматия по искусственному интеллекту на английском языке. Учебное пособие. Москва, 2024. – 171 с.

3. Бессмертный И.А., Авдюшина А.Е., Кугаевских А.В., Королева Ю.А., Рущенко Н.Г. Системы искусственного интеллекта. Методическое пособие. Санкт-Петербург: Университет ИТМО, 2024. – 281 с.

4. Андреянов Н.В., Евдокимова Т.С., Павлов А.Д., и др. Методы искусственного интеллекта. Учебное пособие. Казань: КНИТУ-КАИ, 2024. – 392 с.