АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет инженерных и информационных технологий

Кафедра компьютерной инженерии

ПРОЕКТИРОВАНИЕ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ЛЕКЦИЯ № 10

Генеративный ИИ, LLM, embeddings и RAG

Теоретико-практический конспект

Составитель: Алтыбай А.

Кафедра компьютерной инженерии

Алматы, 2026

Аннотация и назначение лекции

Лекция 10 раскрывает тему «Генеративный ИИ, LLM, embeddings и RAG» в контексте разработки промышленной AI-системы. Материал опирается на тематический план дисциплины и соединяет определения, принципы проектирования, математические оценки, проектные артефакты, практический код и аргументированный анализ рисков. Вопросы построены вокруг решений, которые предстоит принимать инженеру, а не только вокруг названий алгоритмов. [1, 2]

План занятия

1. Foundation models
2. Токенизация и контекст
3. Prompt design
4. Эмбеддинги и релевантность
5. Chunking документов
6. RAG pipeline
7. Векторная база и индексация
8. Evaluation RAG
9. Практические сценарии, контроль качества и проектные задания.

Ожидаемые результаты обучения

По завершении студент способен:

1. объяснить и аргументированно применить принцип «Foundation models» к инженерному решению;
2. объяснить и аргументированно применить принцип «Токенизация и контекст» к инженерному решению;

3. объяснить и аргументированно применить принцип «Prompt design» к инженерному решению;
4. объяснить и аргументированно применить принцип «Эмбеддинги и релевантность» к инженерному решению;
5. объяснить и аргументированно применить принцип «Chunking документов» к инженерному решению;
6. объяснить и аргументированно применить принцип «RAG pipeline» к инженерному решению;
7. реализовать упрощённый проверяемый прототип и сформулировать критерии его приемки.
8. указать угрозы достоверности результатов и предложить проверки до эксплуатации.

Сквозной кейс. В течение лекции рассматриваем внутреннего помощника по регламентам. Существенные условия: галлюцинацию без источника, утечку закрытого документа. Основная проверяемая характеристика: retrieval recall и точность ссылок. Числа в расчётных примерах учебные и не являются результатами реальных измерений.

1. Базовые теоретические положения из конспекта курса

Ниже представлены исходные положения учебно-методического конспекта, связанные с данной темой. Следующие разделы раскрывают их с инженерными деталями и проверяемыми примерами. [1, 2] Цель лекции. дать инженерное понимание foundation models и LLM-приложений, embeddings, vector databases, RAG, prompt engineering, evaluation и выбора между RAG и fine-tuning.

Ключевые понятия. foundation model, LLM, token, context window, prompt, embedding, vector database, retrieval, RAG, chunking, reranking, fine-tuning, guardrails.

1.1 Foundation models и LLM как компонент системы

Foundation model — крупная предварительно обученная модель, которую можно адаптировать к множеству задач. При использовании API или локального LLM разработчик работает не только с моделью, но и с prompt, контекстом, инструментами, политиками безопасности, retrieval и postprocessing.

LLM-приложение является стохастической системой: одинаковый запрос при ненулевой temperature может давать разные ответы. Поэтому требования к evaluation и logging выше, чем для детерминированного сервиса.

1.2 Embeddings и векторный поиск

Embedding отображает объект в векторное пространство. Для semantic search пользовательский запрос и документы кодируются векторы, затем выбираются ближайшие по cosine similarity, dot product или другой метрике. Vector database ускоряет хранение, индексирование и approximate nearest neighbor search.

Качество retrieval зависит от embedding model, chunking, metadata filters и reranking. Нельзя оценивать только финальный ответ LLM, не измеряя retrieval отдельно.

1.3 RAG pipeline

RAG дополняет LLM внешними документами во время запроса. Этапы: подготовка корпуса → chunking → embeddings → индекс → retrieval → reranking → формирование context → генерация ответа → проверка/ссылки. Основное преимущество — возможность использовать актуальные и частные данные без полного переобучения модели.

RAG не гарантирует фактическую корректность. Ошибки могут возникнуть на retrieval, выборе chunk, prompt или самой генерации.

1.4 RAG или fine-tuning?

RAG подходит, когда нужно дать модели доступ к изменяющимся знаниям или документам и желательно показывать источники. Fine-tuning полезен, когда требуется изменить стиль, формат поведения, устойчиво обучить специализированный паттерн или адаптировать модель к задаче. Часто эти методы дополняют друг друга.

Нельзя fine-tuning рассматривать как надежный способ «загрузить базу знаний»: модель может плохо воспроизводить конкретные факты и их обновление становится дорогим.

1.5 Evaluation и guardrails

LLM-систему следует оценивать по набору репрезентативных запросов. Отдельно измеряются retrieval recall, answer correctness, groundedness, citation accuracy, latency и cost. Для высокорисковых сценариев используется human review.

Guardrails включают проверку входа, ограничение инструментов, фильтрацию чувствительных данных, schema validation выходов, policy checks и защиту от prompt injection.

упрощенная логика RAG query_vec = embed(user_query) docs = vector_db.search(query_vec, top_k=5) context = ”\n”.join(d.text for d in docs) prompt = f”Ответь только по контексту:\n{context}\nВопрос: {user_query}” answer = llm.generate(prompt)

1.6 Выводы

- LLM-приложение включает больше компонентов, чем сама языковая модель.
- RAG разделяет задачу поиска информации и генерации ответа.
- Retrieval следует оценивать независимо от генерации.
- RAG и fine-tuning решают разные классы проблем.

- Guardrails и evaluation обязательны для production LLM-приложения.

1.7 Контрольные вопросы

- Какие основные этапы RAG pipeline?
- От чего зависит качество retrieval?
- Когда fine-tuning предпочтительнее RAG?
- Что такое groundedness?
- Какие меры снижают риск prompt injection?

2. Систематическое изложение темы и инженерные решения

2.1 Foundation models

Большие предобученные модели представляют универсальные функции обработки текста или других модальностей. Способ интеграции определяется качеством, стоимостью, правами на данные и обновляемостью. [2]

Проверяемый пример. Для помощника по внутренним инструкциям сравните локальную модель и облачный API.

Вопрос для обсуждения. Как проверить выполнение такого требования: Для помощника по внутренним инструкциям сравните локальную модель и облачный API.

2.2 Токенизация и контекст

Текст преобразуется в токены; ограниченный контекст и стоимость обработки влияют на chunking и выбор промпта. Длинный контекст сам по себе не гарантирует точную извлекаемость. [3]

Проверяемый пример. Рассчитайте бюджет токенов для вопроса, найденных документов, инструкции и ответа.

Вопрос для обсуждения. Как проверить выполнение такого требования: Рассчитайте бюджет токенов для вопроса, найденных документов, инструкции и ответа.

2.3 Prompt design

Системная инструкция, контекст и формат ответа разделяются. Отсутствие проверки сгенерированного JSON приводит к ошибкам интеграции. [4]

Проверяемый пример. Определите требуемую схему JSON и правило повтора при нарушении схемы.

Вопрос для обсуждения. Как проверить выполнение такого требования: Определите требуемую схему JSON и правило повтора при нарушении схемы.

2.4 Эмбеддинги и релевантность

Векторный поиск поддерживает приближенное семантическое соответствие, но может ошибаться на числах и редких названиях. Гибридный поиск часто дополняет семантический. [5]

Проверяемый пример. Для инструкции «заявка №1007» сравните lexical и semantic retrieval.

Вопрос для обсуждения. Как проверить выполнение такого требования: Для инструкции «заявка №1007» сравните lexical и semantic retrieval.

2.5 Chunking документов

Размер и overlap отрывков влияют на полноту контекста, стоимость и потерю связности. Чанки должны сохранять источник, заголовок и дату действия документа. [6]

Проверяемый пример. Разбейте регламент по разделам, сохранив номера страниц для цитирования.

Вопрос для обсуждения. Как проверить выполнение такого требования: Разбейте регламент по разделам, сохранив номера страниц для цитирования.

2.6 RAG pipeline

Retrieval, reranking, packing и генерация образуют независимые этапы. Ошибка поиска не исправляется более красивым промптом. [7]

Проверяемый пример. Соберите trace из top-k документов и обоснования итогового ответа.

Вопрос для обсуждения. Как проверить выполнение такого требования: Соберите trace из top-k документов и обоснования итогового ответа.

2.7 Векторная база и индексация

Индекс хранит embedding и метаданные фильтрации; ACL-фильтрация должна применяться до выдачи контекста. Устаревший индекс может нарушать права доступа. [1]

Проверяемый пример. Смоделируйте удаление закрытого документа и его фрагментов из поискового индекса.

Вопрос для обсуждения. Как проверить выполнение такого требования: Смоделируйте удаление закрытого документа и его фрагментов из поискового индекса.

2.8 Evaluation RAG

Отдельно проверяют retrieval recall, релевантность, groundedness и правильность финального ответа. Нельзя сводить оценку к «ответ кажется хорошим». [2]

Проверяемый пример. Составьте набор 30 вопросов с эталонными документами, ответами и недопустимыми галлюцинациями.

Вопрос для обсуждения. Как проверить выполнение такого требования: Составьте набор 30 вопросов с эталонными документами, ответами и недопустимыми галлюцинациями.

2.9 Fine-tuning против RAG

RAG подходит для часто меняющихся фактов, fine-tuning — для поведения и формата, но граница не абсолютна. Выбирают по эмпирическим испытаниям и обновляемости. [3]

Проверяемый пример. Для еженедельного регламента обоснуйте индексирование, а не переобучение весов каждый понедельник.

Вопрос для обсуждения. Как проверить выполнение такого требования: Для еженедельного регламента обоснуйте индексирование, а не переобучение весов каждый понедельник.

2.10 Hallucination и отказ

Модель может отвечать уверенно без доказательного источника. Система должна уметь сообщать о недостаточности данных и приводить корректные ссылки. [4]

Проверяемый пример. Потребуйте отказ при отсутствии релевантного документа после reranking.

Вопрос для обсуждения. Как проверить выполнение такого требования: Потребуйте отказ при отсутствии релевантного документа после reranking.

2.11 Prompt injection

Извлеченный текст является недоверенным вводом: не должен переопределять системные правила и расширять доступ к ресурсам. [5]

Проверяемый пример. Вставьте в учебный документ текст «игнорируй правила» и протестируйте изоляцию инструкций.

Вопрос для обсуждения. Как проверить выполнение такого требования: Вставьте в учебный документ текст «игнорируй правила» и протестируйте изоляцию инструкций.

2.12 Экономика и задержка

Стоимость зависит от числа токенов, модели, reranking, кэша и частоты обращений. Улучшение качества оценивают совместно с latency и денежными затратами. [6]

Проверяемый пример. Прикиньте суточную стоимость для 2000 запросов при заранее принятой цене токена.

Вопрос для обсуждения. Как проверить выполнение такого требования: Прикиньте суточную стоимость для 2000 запросов при заранее принятой цене токена.

3. Математические инструменты и количественные критерии

Для перехода от неформального решения к проверяемой постановке введём следующие величины.

$$\mathrm{sim}(q, d) = \frac{e_q^\top e_d}{\|e_q\| \, \|e_d\|} \tag{1}$$

$$\mathrm{RAG}(q) = \mathrm{LLM}\big(q, \mathrm{TopK}_{d \in D} \, \mathrm{sim}(q, d)\big) \tag{2}$$

Выбор top-k зависит от recall поиска, бюджета контекста и качества источников. LLM не получает доступа к документу, который не прошёл retrieval.

При выборе числового критерия нужно отличать обучающую функцию потерь, продуктовую ценность и эксплуатационные ограничения. Первую можно оптимизировать дифференцированием, вторая определяется процессом принятия решения, а третьи проверяются нагрузочными испытаниями, политиками безопасности и мониторингом.

3.1 Пошаговый расчёт на учебных данных

Рассмотрим две конфигурации одной системы. Вариант A имеет условную оценку качества $q_A = 0.86$ и задержку $t_A = 90$ мс; вариант B — $q_B = 0.89$ и $t_B = 280$ мс. Предположим, что требование заказчика задаёт $q \geq 0.85$ и $p95 \leq 150$ мс. Тогда A удовлетворяет обоим условиям, тогда как B нарушает второе. Следовательно, выбор B только потому, что его offline-метрика выше, не является обоснованным. Числа иллюстративны; реальный вывод требует измерений при одинаковой нагрузке и одинаковом наборе данных.

$$\mathrm{accept}(A) = \mathbf{1}\{0.86 \geq 0.85 \wedge 90 \leq 150\} = 1,$$
$$\mathrm{accept}(B) = \mathbf{1}\{0.89 \geq 0.85 \wedge 280 \leq 150\} = 0.$$

3.2 Как интерпретировать число, а не только вычислить его

Сначала следует проверить единицы измерения, состав выборки и смысл агрегирования. Например, p95 на спокойном стенде нельзя переносить на рабочую нагрузку с иным числом одновременных запросов. Затем исследуют чувствительность к изменениям входных данных и неопределённость оценок. Наконец, фиксируют последствия порога: какую ошибку он предотвращает, а какую может увеличить. Этот подход делает количественный критерий частью инженерной аргументации, а не украшением отчёта. [1, 2]

4. Практическое программирование на Python

Пример предназначен для демонстрации ключевого инженерного принципа. Он не претендует на обработку реального production-трафика. Запускать код следует в изолированном окружении и фиксировать версии библиотек, если они используются. Вывод зависит от указанных входных данных; для стохастических алгоритмов он может меняться.

4.1 Основной воспроизводимый пример

```python
import numpy as np

def cosine_search(query, documents, k=2):
    query = np.asarray(query, dtype=float)
    docs = np.asarray(documents, dtype=float)
    query = query / max(np.linalg.norm(query), 1e-12)
    docs = docs / np.maximum(np.linalg.norm(docs, axis=1, keepdims=True), 1e-12)
    scores = docs @ query
    indexes = np.argsort(-scores)[:k]
    return [(int(i), float(scores[i])) for i in indexes]

doc_vectors = [[1., 0.], [0.8, 0.2], [0., 1.]]
print(cosine_search([1., 0.], doc_vectors, k=2))
# Production uses a learned embedding model, document ACL and citations.
```

Проверка результатов. Сначала убедитесь, что код завершился без исключений; проверьте формы массивов, схему результата и отсутствие недопустимых значений. Если используются внешние данные, включите в отчёт способ их получения и идентификатор версии. Измените один входной параметр и объясните, почему наблюдаемый результат изменился или остался прежним.

4.2 Компактная проверка контракта или метрики

```python
def rag_guard(answer, citations):
    if not citations:
```

```
        return "Недостаточно подтвержденных источников"
    return answer
print(rag_guard("Example answer", []))
```

При самостоятельном повторении предложите минимум один негативный тест: некорректное значение, пустой вход, несовместимую версию схемы либо нарушение SLO. Переход от успешного демонстрационного запуска к надежной системе требует именно таких проверок.

5. Разбор трёх инженерных ситуаций

5.1 Ситуация 1: Foundation models

Заказчик использует внутреннего помощника по регламентам. Новое требование заставляет рассмотреть принцип «Foundation models». Основная неопределённость состоит в том, как одновременно обеспечить retrieval recall и точность ссылок, не допустить утечку закрытого документа и сохранить возможность изменения модели.

Постановка. Большие предобученные модели представляют универсальные функции обработки текста или других модальностей. Способ интеграции определяется качеством, стоимостью, правами на данные и обновляемостью.

Действие инженера. Для помощника по внутренним инструкциям сравните локальную модель и облачный API.

Альтернативы. Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

Вариант	Данные и зависимости	Проверка	Основной риск
Правила	Фиксированные условия	Набор тест-кейсов	Ограниченная гибкость
ML-модуль	Версионированные признаки	Offline и API tests	Drift и skew
Сложный сервис	Несколько компонентов	End-to-end и нагрузка	Каскадный отказ

Проверка решения. Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

5.2 Ситуация 2: RAG pipeline

Заказчик использует внутреннего помощника по регламентам. Новое требование заставляет рассмотреть принцип «RAG pipeline». Основная неопределённость состоит в том, как одновременно обеспечить retrieval recall и точность ссылок, не допустить утечку закрытого документа и сохранить возможность изменения модели.

Постановка. Retrieval, reranking, packing и генерация образуют независимые этапы. Ошибка поиска не исправляется более красивым промптом.

Действие инженера. Соберите trace из top-k документов и обоснования итогового ответа.

Альтернативы. Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

Вариант	Данные и зависимости	Проверка	Основной риск
Правила	Фиксированные условия	Набор тест-кейсов	Ограниченная гибкость
ML-модуль	Версионированные признаки	Offline и API tests	Drift и skew
Сложный сервис	Несколько компонентов	End-to-end и нагрузка	Каскадный отказ

Проверка решения. Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

5.3 Ситуация 3: Prompt injection

Заказчик использует внутреннего помощника по регламентам. Новое требование заставляет рассмотреть принцип «Prompt injection». Основная неопределённость состоит в том, как одновременно обеспечить retrieval recall и точность ссылок, не допустить утечку закрытого документа и сохранить возможность изменения модели.

Постановка. Извлеченный текст является недоверенным вводом: не должен переопределять системные правила и расширять доступ к ресурсам.

Действие инженера. Вставьте в учебный документ текст «игнорируй правила» и протестируйте изоляцию инструкций.

Альтернативы. Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

Вариант	Данные и зависимости	Проверка	Основной риск
Правила	Фиксированные условия	Набор тест-кейсов	Ограниченная гибкость
ML-модуль	Версионированные признаки	Offline и API tests	Drift и skew
Сложный сервис	Несколько компонентов	End-to-end и нагрузка	Каскадный отказ

Проверка решения. Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

6. Самостоятельная работа: индивидуальные проектные задания

Для каждого задания студент создаёт отдельный небольшой артефакт: схему, набор тестов, код либо таблицу сравнения, прикладывает наблюдаемый результат и объясняет ограничения.

1. **Foundation models.** Для помощника по внутренним инструкциям сравните локальную модель и облачный API. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

2. **Токенизация и контекст.** Рассчитайте бюджет токенов для вопроса, найденных документов, инструкции и ответа. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

3. **Prompt design.** Определите требуемую схему JSON и правило повтора при нарушении схемы. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

4. **Эмбеддинги и релевантность.** Для инструкции «заявка №1007» сравните lexical и semantic retrieval. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

5. **Chunking документов.** Разбейте регламент по разделам, сохранив номера страниц для цитирования. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

6. **RAG pipeline.** Соберите trace из top-k документов и обоснования итогового ответа. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

7. **Векторная база и индексация.** Смоделируйте удаление закрытого документа и его фрагментов из поискового индекса. Сформулируйте

проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

8. **Evaluation RAG.** Составьте набор 30 вопросов с эталонными документами, ответами и недопустимыми галлюцинациями. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

9. **Fine-tuning против RAG.** Для еженедельного регламента обоснуйте индексирование, а не переобучение весов каждый понедельник. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

10. **Hallucination и отказ.** Потребуйте отказ при отсутствии релевантного документа после reranking. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

11. **Prompt injection.** Вставьте в учебный документ текст «игнорируй правила» и протестируйте изоляцию инструкций. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

12. **Экономика и задержка.** Прикиньте суточную стоимость для 2000 запросов при заранее принятой цене токена. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

7. Вопросы для текущего и итогового контроля

1. В чём назначение и ограничения принципа «Foundation models»?

2. Как проверить на практике утверждение: «Для помощника по внутренним инструкциям сравните локальную модель и облачный API.»?

3. В чём назначение и ограничения принципа «Токенизация и контекст»?

4. Как проверить на практике утверждение: «Рассчитайте бюджет токе-

нов для вопроса, найденных документов, инструкции и ответа.»?

5. В чём назначение и ограничения принципа «Prompt design»?
6. Как проверить на практике утверждение: «Определите требуемую схему JSON и правило повтора при нарушении схемы.»?
7. В чём назначение и ограничения принципа «Эмбеддинги и релевантность»?
8. Как проверить на практике утверждение: «Для инструкции «заявка №1007» сравните lexical и semantic retrieval.»?
9. В чём назначение и ограничения принципа «Chunking документов»?
10. Как проверить на практике утверждение: «Разбейте регламент по разделам, сохранив номера страниц для цитирования.»?
11. В чём назначение и ограничения принципа «RAG pipeline»?
12. Как проверить на практике утверждение: «Соберите trace из top-k документов и обоснования итогового ответа.»?
13. В чём назначение и ограничения принципа «Векторная база и индексация»?
14. Как проверить на практике утверждение: «Смоделируйте удаление закрытого документа и его фрагментов из поискового индекса.»?
15. В чём назначение и ограничения принципа «Evaluation RAG»?
16. Как проверить на практике утверждение: «Составьте набор 30 вопросов с эталонными документами, ответами и недопустимыми галлюцинациями.»?
17. В чём назначение и ограничения принципа «Fine-tuning против RAG»?
18. Как проверить на практике утверждение: «Для еженедельного регламента обоснуйте индексирование, а не переобучение весов каждый понедельник.»?
19. В чём назначение и ограничения принципа «Hallucination и отказ»?
20. Как проверить на практике утверждение: «Потребуйте отказ при отсутствии релевантного документа после reranking.»?

21. В чём назначение и ограничения принципа «Prompt injection»?
22. Как проверить на практике утверждение: «Вставьте в учебный документ текст «игнорируй правила» и протестируйте изоляцию инструкций.»?
23. В чём назначение и ограничения принципа «Экономика и задержка»?
24. Как проверить на практике утверждение: «Прикиньте суточную стоимость для 2000 запросов при заранее принятой цене токена.»?
25. Почему высокое offline-качество не гарантирует надежность промышленной AI-системы?
26. Какие артефакты нужны для воспроизводимости и анализа инцидента?

8. Глоссарий ключевых понятий

Foundation models
Большие предобученные модели представляют универсальные функции обработки текста или других модальностей.

Токенизация и контекст
Текст преобразуется в токены; ограниченный контекст и стоимость обработки влияют на chunking и выбор промпта.

Prompt design
Системная инструкция, контекст и формат ответа разделяются.

Эмбеддинги и релевантность
Векторный поиск поддерживает приближенное семантическое соответствие, но может ошибаться на числах и редких названиях.

Chunking документов
Размер и overlap отрывков влияют на полноту контекста, стоимость и потерю связности.

RAG pipeline
Retrieval, reranking, packing и генерация образуют независимые этапы.

Векторная база и индексация
Индекс хранит embedding и метаданные фильтрации; ACL-фильтрация должна применяться до выдачи контекста.

Evaluation RAG
Отдельно проверяют retrieval recall, релевантность, groundedness и правильность финального ответа.

Fine-tuning против RAG
RAG подходит для часто меняющихся фактов, fine-tuning — для поведения и формата, но граница не абсолютна.

Hallucination и отказ
Модель может отвечать уверенно без доказательного источника.

Prompt injection
Извлеченный текст является недоверенным вводом: не должен переопре-

делять системные правила и расширять доступ к ресурсам.

Экономика и задержка
Стоимость зависит от числа токенов, модели, reranking, кэша и частоты обращений.

9. Итоги и связь со следующей темой

Представленные методы имеют смысл как части целостного управляемого жизненного цикла. Ключевой результат инженерного проектирования состоит не в максимальном числе компонентов или сложности модели, а в убедительной связи требований, проектных решений, измерений и механизмов сопровождения. Дальнейшее изучение курса развивает эти связи на следующем этапе жизненного цикла.

Список литературы

[1] Lewis P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS, 2020. https://arxiv.org/abs/2005.11401.

[2] Vaswani A. et al. Attention Is All You Need. NeurIPS, 2017. https://arxiv.org/abs/1706.03762.

[3] Huyen C. Designing Machine Learning Systems. O'Reilly Media, 2022. https://www.oreilly.com/library/view/designing-machine-learning/9781098107956/.

[4] OWASP. Top 10 for Large Language Model Applications. https://genai.owasp.org/llm-top-10/.

[5] NIST. AI Risk Management Framework (AI RMF 1.0). NIST AI 100-1, 2023. https://doi.org/10.6028/NIST.AI.100-1.

[6] Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. https://www.deeplearningbook.org/.

[7] Murphy K. P. Probabilistic Machine Learning: An Introduction. MIT Press, 2022. https://probml.github.io/pml-book/.