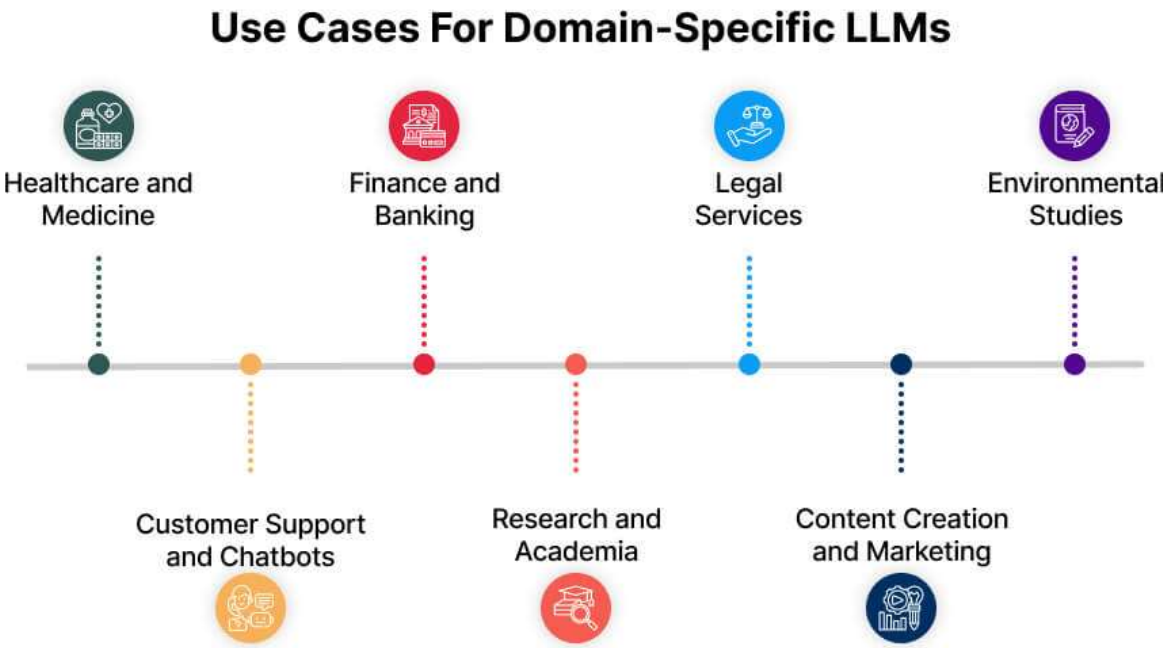
Современные LLM – часть 2

Что такое домен?

Домен в контексте адаптации больших языковых моделей (LLM) — это специализированная область знаний с собственной терминологией, типами текстов и особенностями данных.

Например, медицина, финансы, юриспруденция, e-commerce — всё это отдельные домены



SoluLab

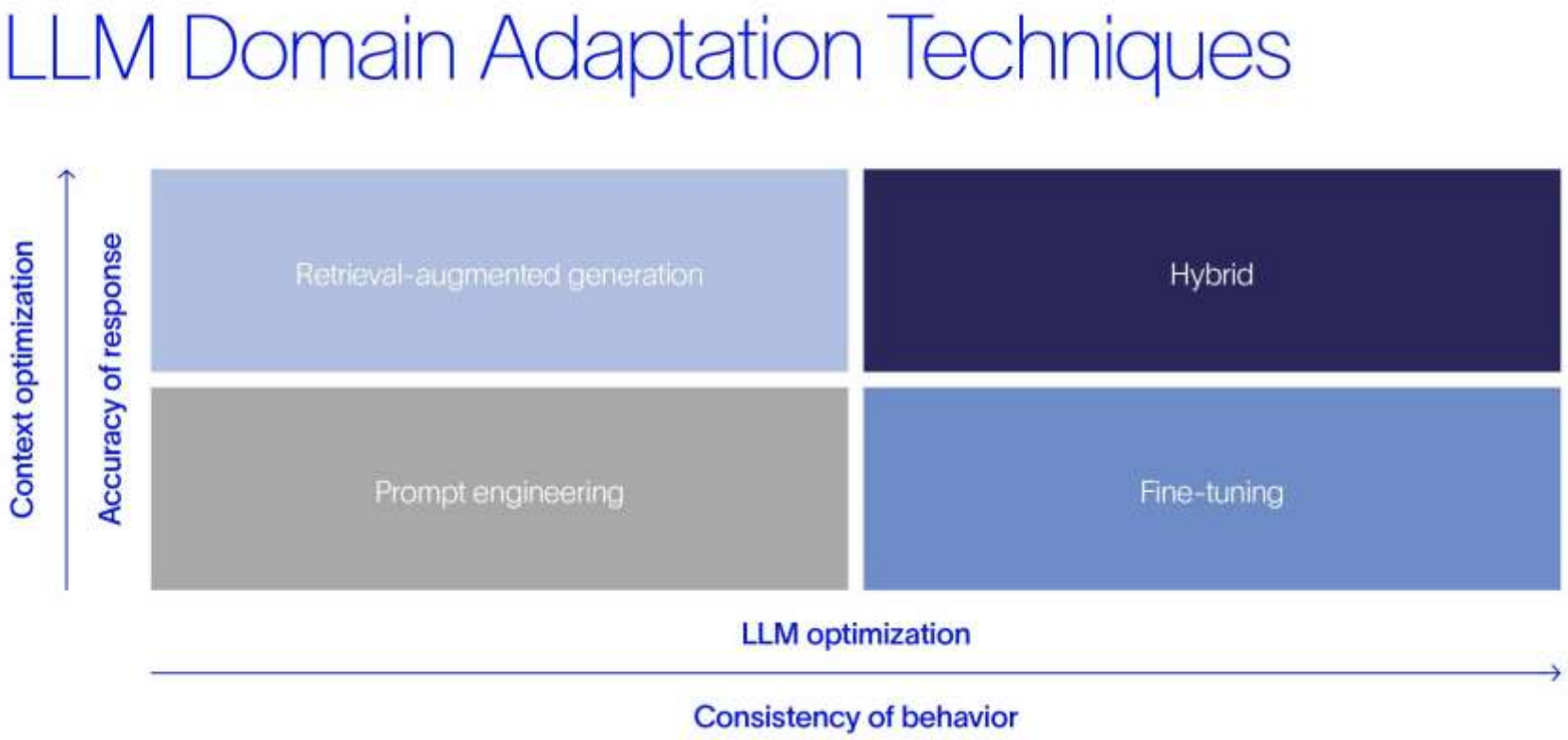
Проблемы без адаптации

- Низкое качество системы
- Галлюцинации
- Несоответствие терминологии
- Регуляторные риски

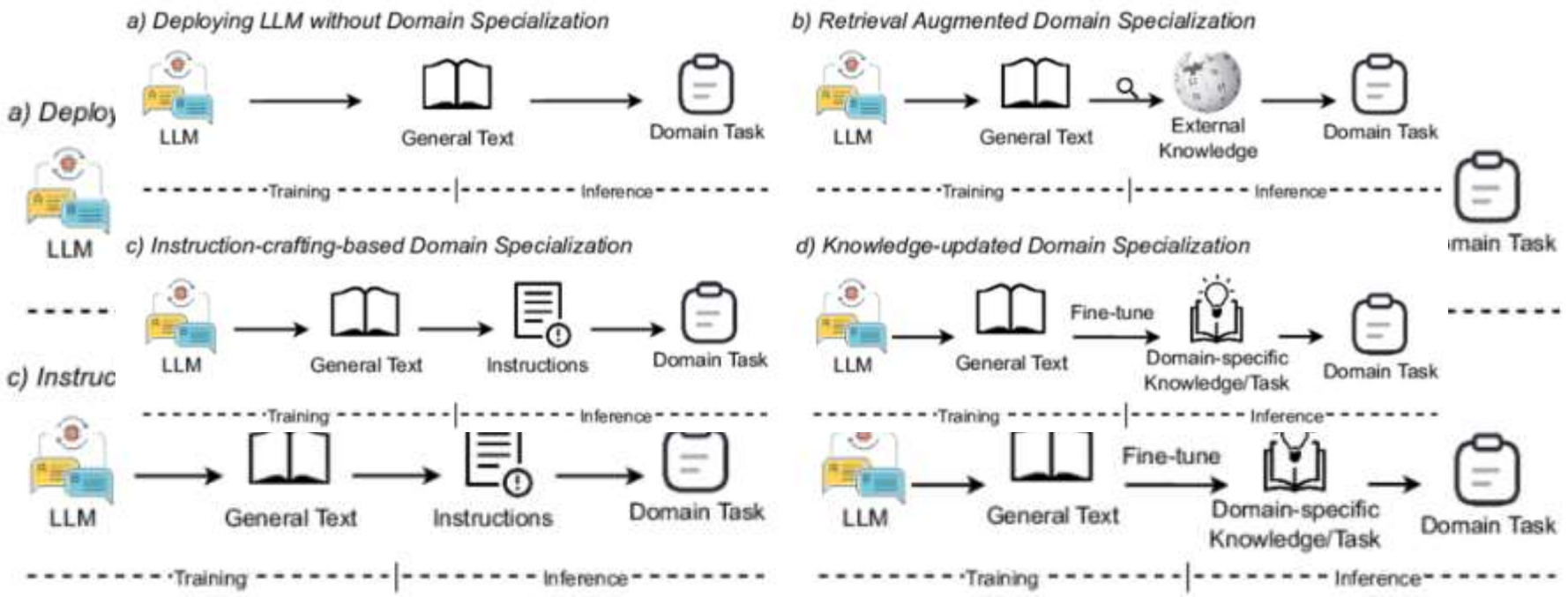
Domain adaptation VS task adaptation VS transfer learning

- **Transfer learning** – концепция использования знаний, полученных при решении одной задачи машинного обучения, для другой связанной задачи
- **Domain adaptation** - область transfer learning, которая фокусируется на обучении модели на одном распределении данных (исходный домен) и применении её к связанному, но отличному распределению данных (целевой домен)
- **Task adaptation** - адаптация модели для выполнения конкретных задач в рамках того же домена или схожих доменов. Это может включать fine-tuning модели для улучшения производительности на специфических задачах, таких как классификация тональности или генерация кода

Виды адаптации под домен



Виды адаптации под домен



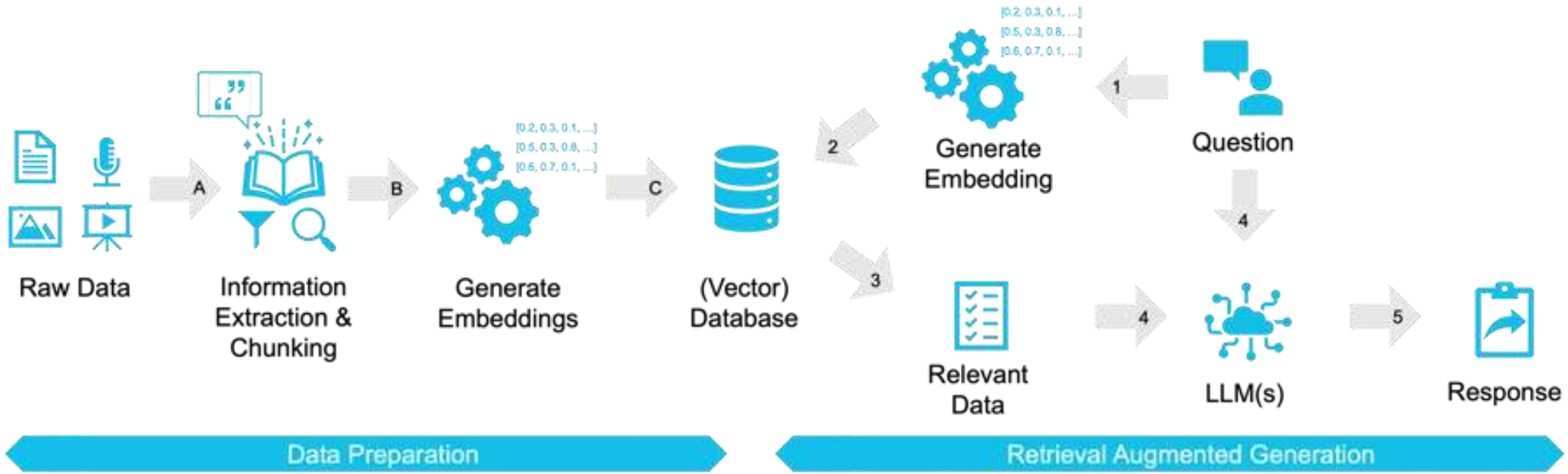
Prompt Engineering

- **Instruction-Following**. Предоставление инструкции перед задачей.
- **Chain-of-Thought**. Разбиения мыслительного процесса на ряд промежуточных шагов.
- **Impersonation**. Попросить LLM выдавать себя за эксперта в предметной области при ответе на вопрос, специфичный для предметной области.
- **Chaining**. Разбить задачу на цепочку маленьких более понятных
- …

Prompt Engineering - гайды

- Anthropic: Prompt engineering overview
- Google: Prompt Engineering
- Prompting Guide 101
- A Systematic Survey of Prompt Engineering Techniques

RAG


Raw Data
A
Information Extraction & Chunking
B
Generate Embeddings
C
(Vector) Database
2
Generate Embedding
1
Question
4
3
Relevant Data
4
LLM(s)
5
Response
Data Preparation
Retrieval Augmented Generation

Выбор стратегии адаптации

Prompt engineering – для/при:

- PoC / MVP
- Простых доменных задач без необходимости специализированных знаний
- Ограниченных ресурсах

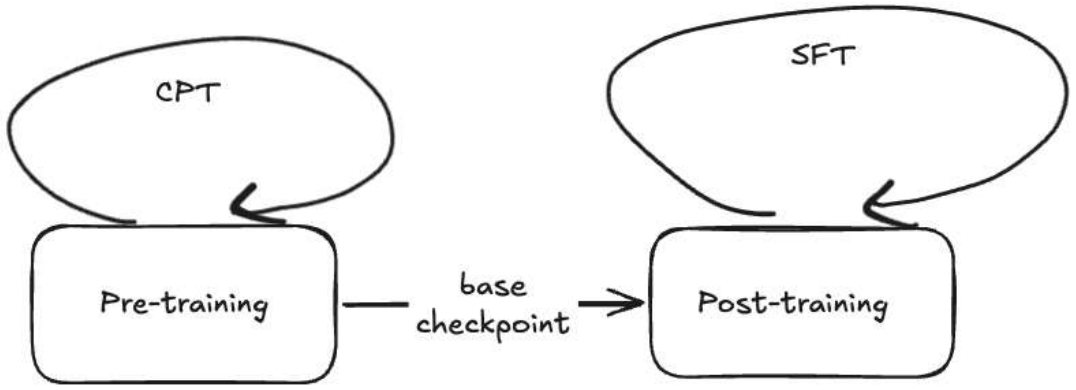
RAG:

- Динамические знания
- Фактическая точность критична
- Доступ к большим базам знаний возможен
- Ограниченный бюджет

Fine-tuning:

- Специализированные задачи требующих высокой точности
- Стабильная производительность в рамках конкретного домена
- Катастрофическое забывание общих знаний приемлемо
- Доступны качественные доменные данные для обучения

CPT vs SFT



- **Continual Pre-Training (CPT)** — это процесс дополнительного обучения уже предварительно обученной языковой модели на новых доменно-специфических данных с использованием той же задачи языкового моделирования.
- **Supervised Fine-Tuning (SFT)** — это процесс настройки предварительно обученной модели на конкретной задаче с использованием размеченных пар "вход-выход".

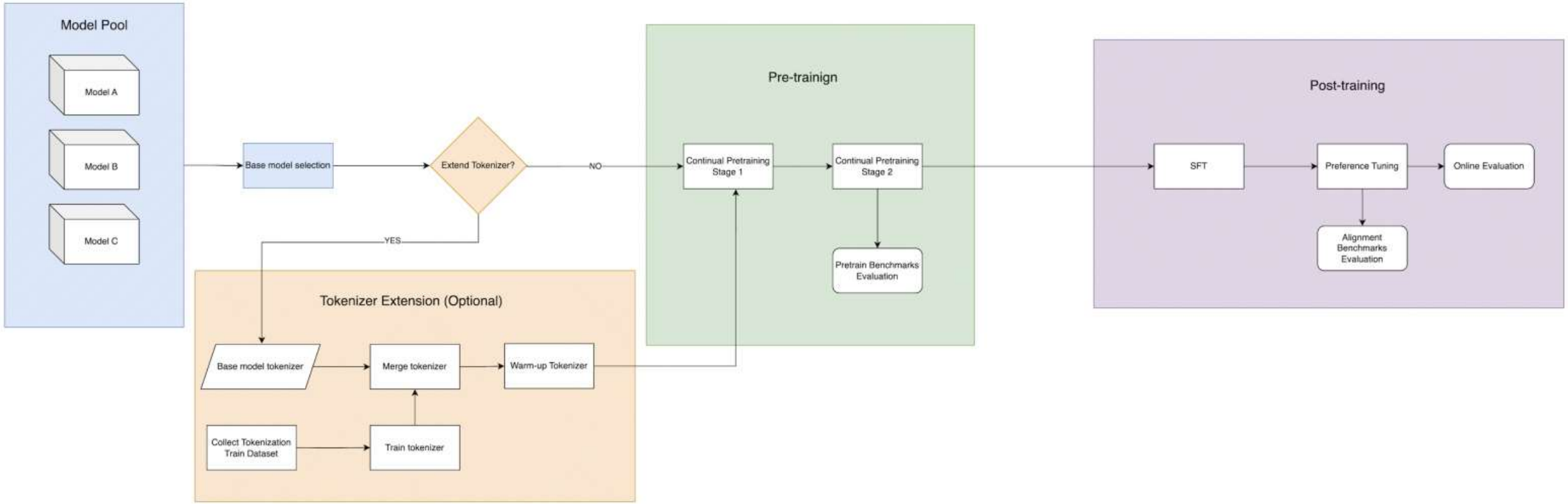
CPT vs SFT

CPT – когда уместен:

- Адаптация под специфическую лексику, отсутствующую в исходных данных обучения
- Интеграция новых фактических знаний
- Адаптация к новым языкам
- Хотим удлинить контекст

SFT – не добавляет новых знаний, менее требователен к ресурсам

CPT + SFT



[Модели T-lite и T-pro: training report]

GPT - токенизатор

Имеет смысл трогать токенизатор, если:

- Сильно разбивается терминология
- Неэффективное сжатие текстов в целом
- Нет каких-то языков/токенов в токенизаторе в принципе

Лучше не трогать, если:

- Уже хорошее покрытие доменных текстов
- Нет ресурсов на дообучение

CPT – что можно сделать с токенизатором?

CPT - токенизатор

- Собираем словарь
- Инициализируем матрицу эмбеддингов:

$$v_{\text{new}}(t_i^n) = \frac{1}{K}\sum_{j=1}^{K} v_{\text{old}}(t_j^o);$$

- Замораживаем все, кроме эмбеддингов и обучаем
- Постепенно или сразу размораживаем слои, обучаем

CPT - данные

- Ключевой рецепт успеха – микс данных претрейна* и домена
- Оставляем немного* данных на языках претрейна
- Следим за распределением данных по отношению к исходной модели
- Вычищаем совсем плохие данные

- FineWeb
- C4
- The Stack V2

Двухстадийный CPT

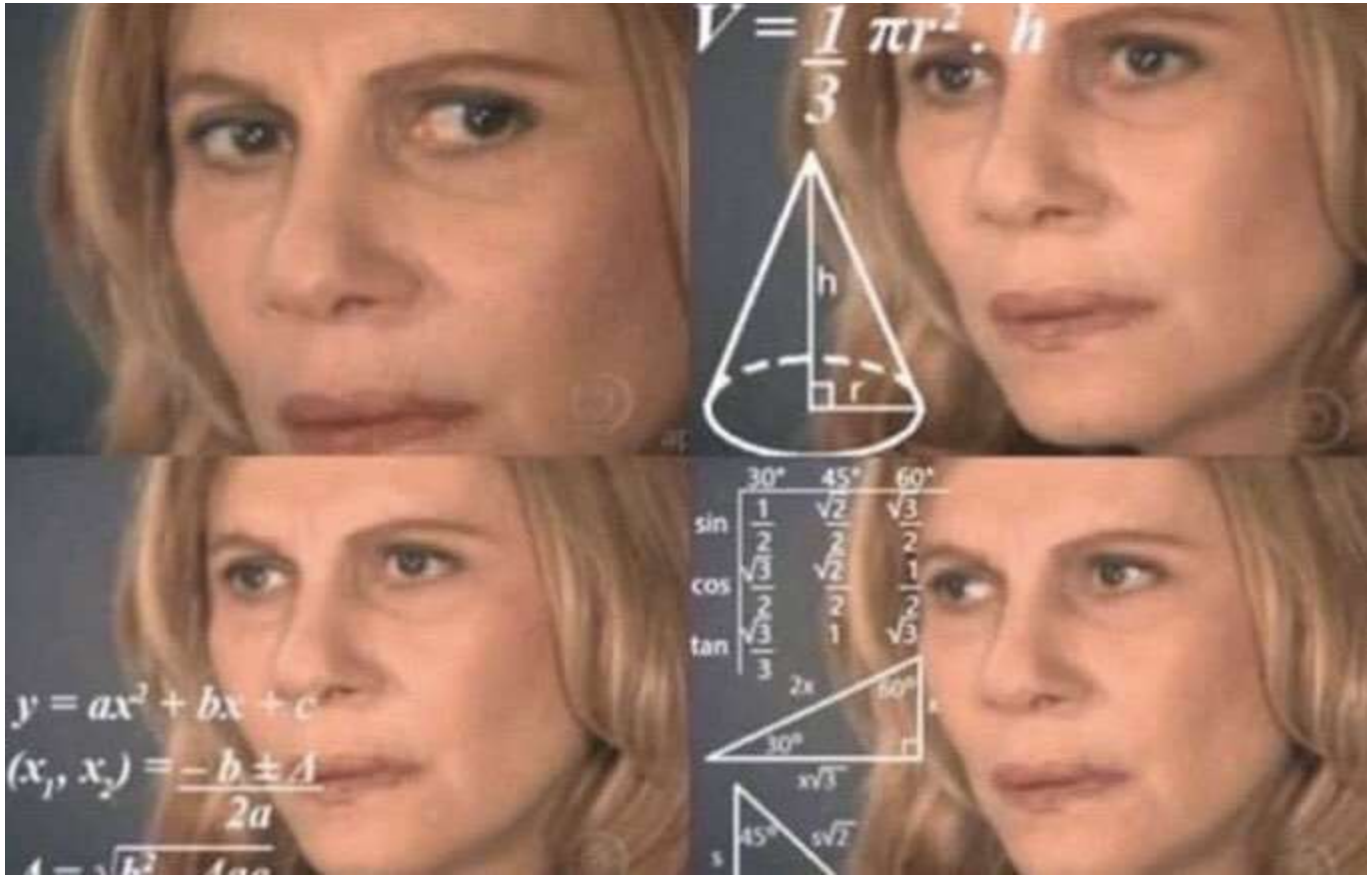
Первая стадия содержит больше данных, в основном это веб-страницы, за счет объема и разнообразности она закладывает основные знания.

Вторая стадия может быть на порядки меньше, при этом содержит более высококачественные данные, такие как инструктивные данные, QA-датасеты, диалоги

GrandMaster-PRO-MAX

Данные для SFT

Большой, размеченный, качественный датасет… где взять?



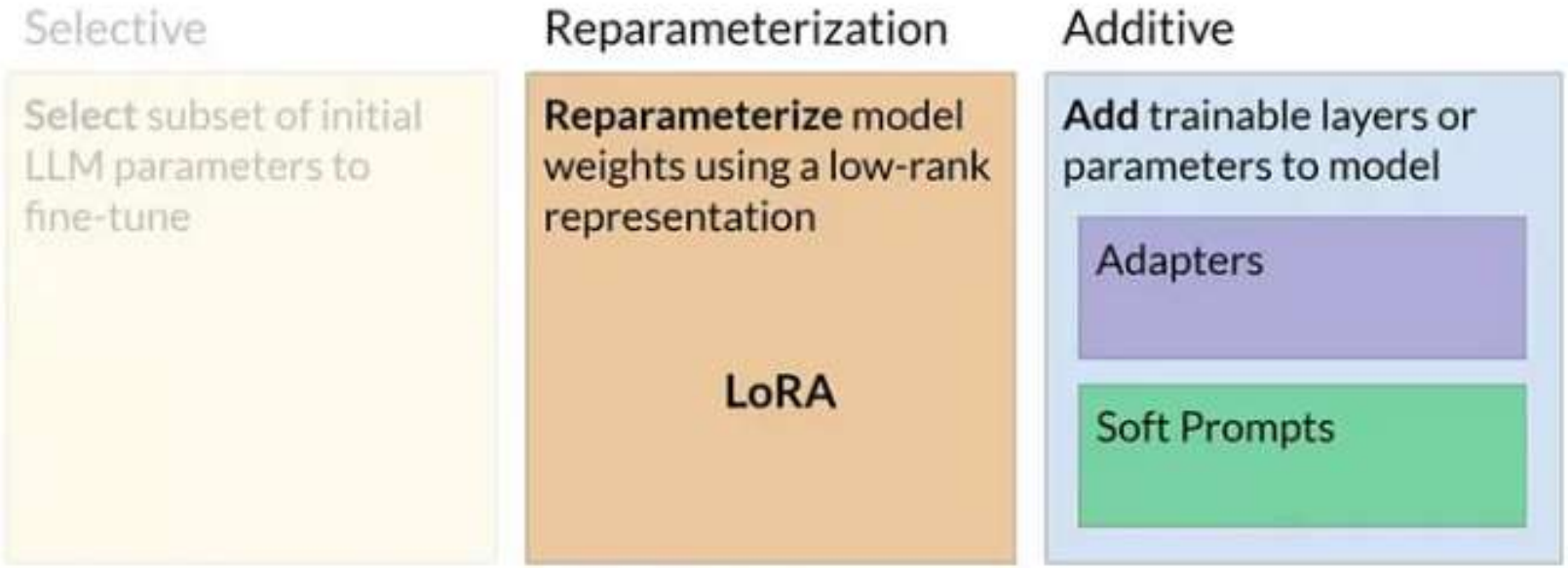
Данные для SFT

Большой, размеченный, качественный датасет… где взять?

- Размечаем самостоятельно / LLM
- Генерируем LLM
- Используем open-source датасеты
- Переводим подходящие датасеты с других языков
- …

CPT + PEFT

- CPT, SFT + PEFT
- CPT + PEFT, SFT + PEFT, одинаковые адаптеры
- CPT + доменные PEFT, SFT + task-PEFT



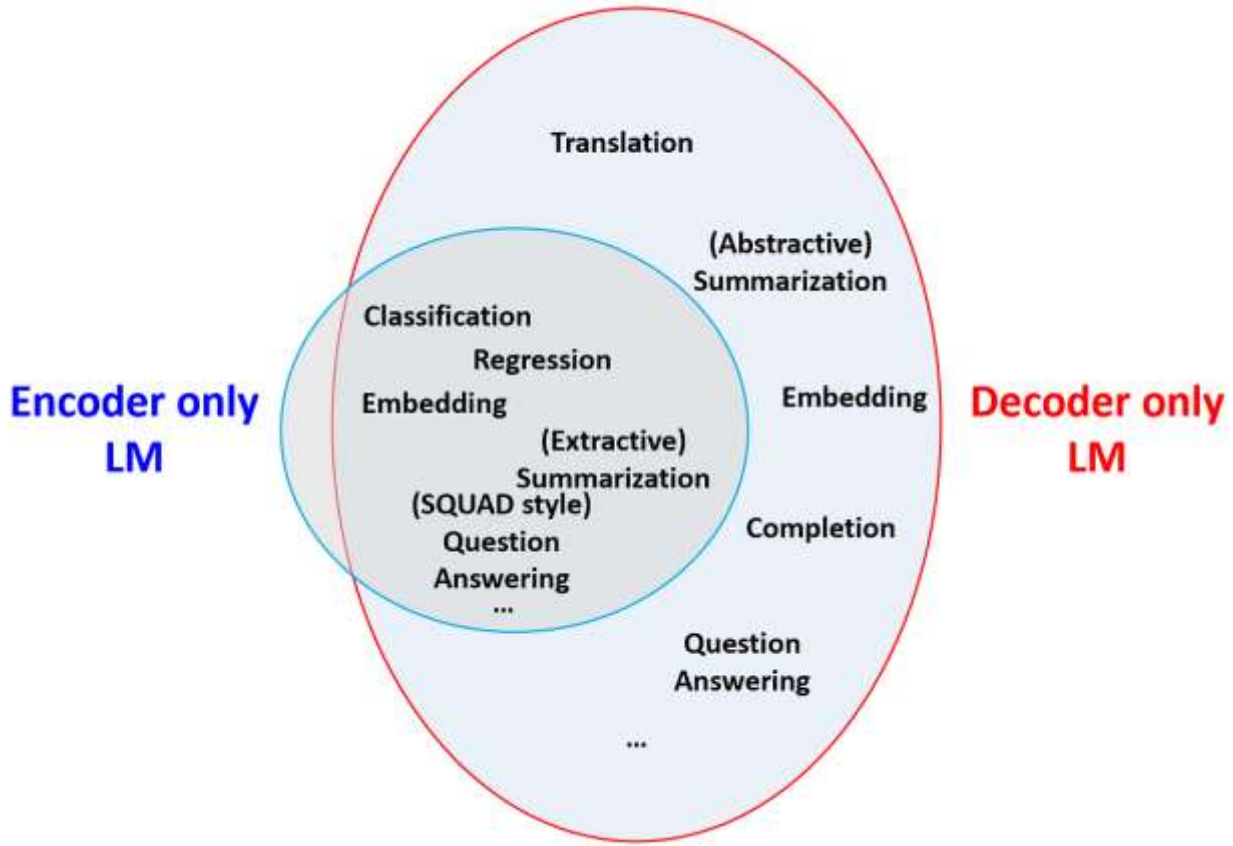
Что может пойти не так?

- Катастрофическое забывание
- Доменный сдвиг
- Переобучение
- Контаминация с бенчмарками

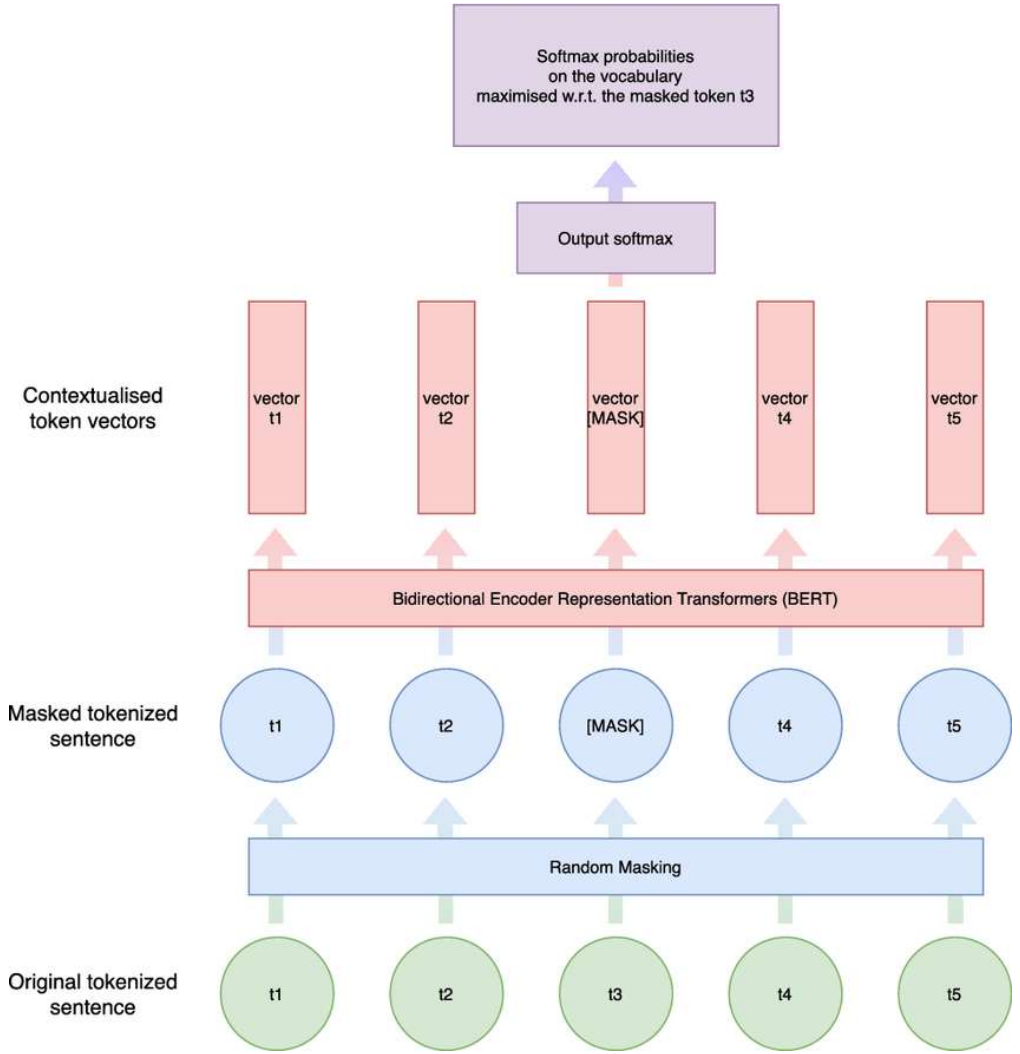
Оценка качества – best practices

- Следить за intrinsic и extrinsic метриками
- Валидировать модель на доменных и общих бенчмарках
- При этом дедуплицировать содержимое сплитов и бенчмарков!
- При изменениях в токенизаторе – стоит посмотреть на новое покрытие, loss по новым токенам
- Быстрая диагностика - проверить качество zero- и few-shot на доменных задачах
- Если удлиняли контекст – проверка извлечения фактов из разных частей длинных текстов

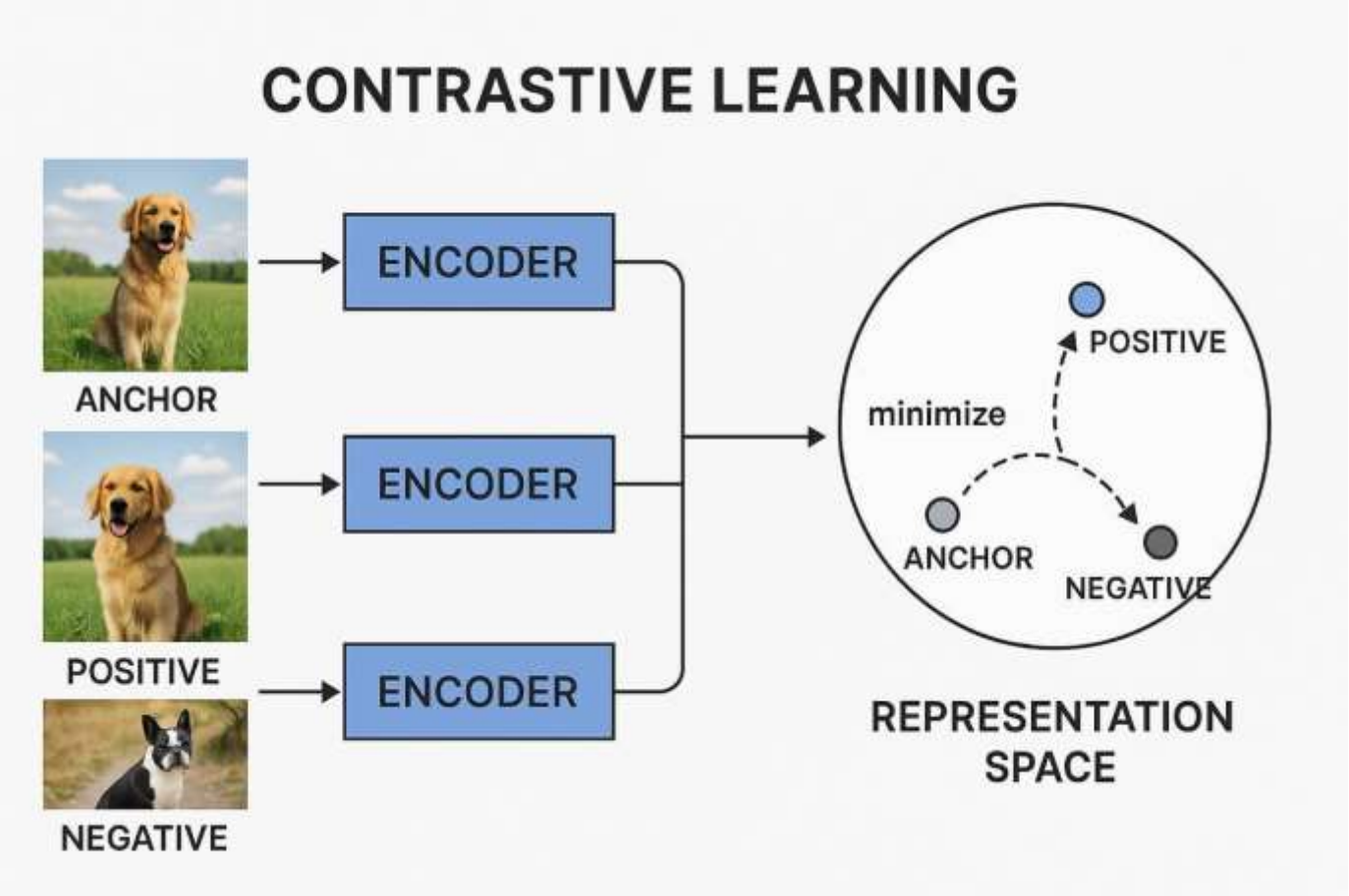
Энкодеры


Encoder only
LM
Translation
(Abstractive)
Summarization
Classification
Regression
Embedding
(Extractive)
Summarization
(SQUAD style)
Question
Answering
...
Embedding
Completion
Question
Answering
...
Decoder only
LM

Unsupervised-адаптация - MLM



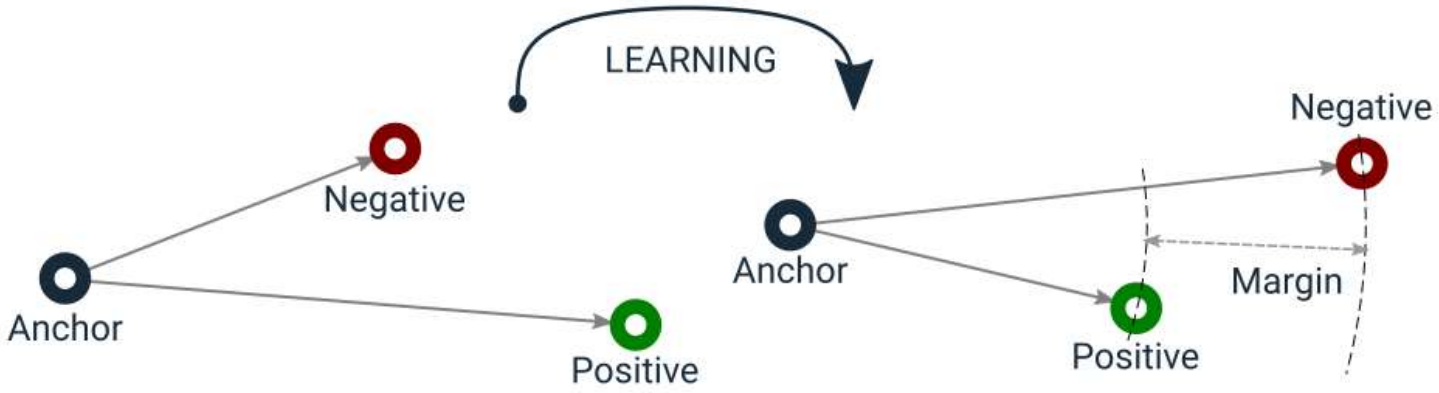
Contrastive Learning



Немного деталей

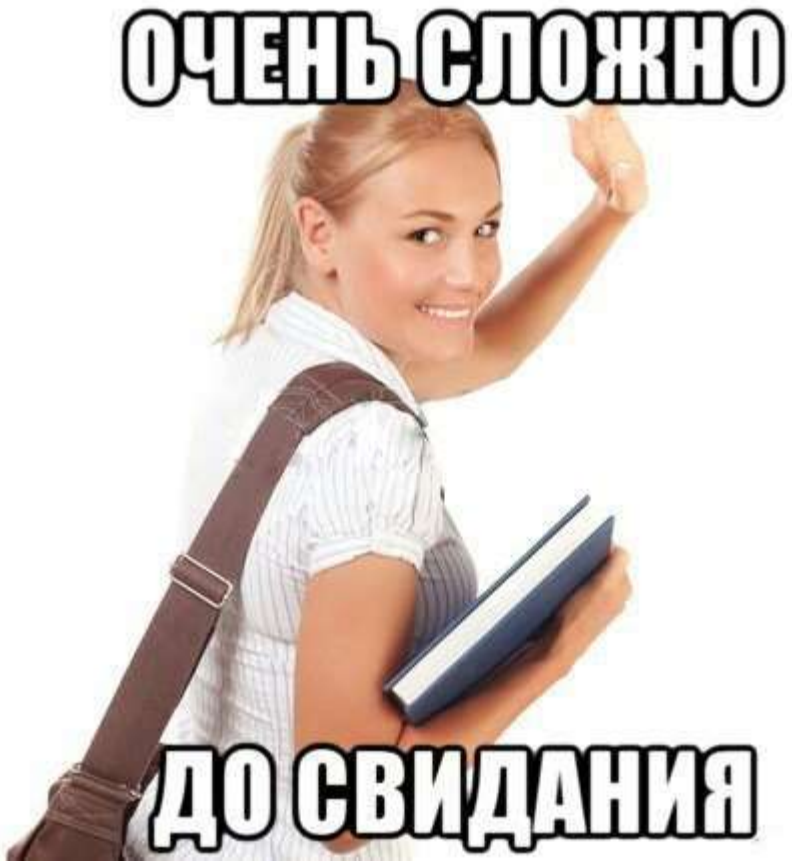
- Triplet loss: $\mathcal{L} = max(d(a,p) - d(a,n) + margin, 0)$

- InfoNCE - то же самое, но N негативных примеров



[The Beginner's Guide to Contrastive Learning]

Где взять данные для CL?



Где взять данные для CL?

- Downstream разметка
- Эвристики на доменных данных: rule-base, векторный поиск, …

Важен hard-negative mining!