Задача:
Сделать умный поиск для поиска не только по ключевым словам
нам нужно получить топ 10

Есть возможность использовать ассесоров.

Функциональные требования:
- Поиск по сложному предложению
- предлагаем 10 вариантов
- Возможность уточнить в чате
- чат (взаимодействия)
- выдавать ссылку в ответе

Ограничения:
Время ответа: 500-800 ms
Памяти: 384 Гб
Железо: 4 * A800
A800 поддерживает до 80 Гб

Модели, которые поместятся в такие условия, но могут иметь потерю в качестве:
Llama 3 (70B) (~35–40 ГБ) в 4-битном квантовании поместится на 1 машину и при распараллеливании на 4 машины позволит получить нужную скорость. Но надо также учитывать пропускную способность шины между GPU.
Mixtral (MoE 8×7B) в 4-битном квантовании (~12–14 ГБ на эксперта)
Qwen-72B в 4-битном квантовании или (в bf16 будет 144 Гб)
Qwen-7B (FP16 или 8-bit)
И другие модели.

При квантовании до 4-битного или 8-битного формата, происходит уменьшение занимаемой памяти в 2-4 раза, но при этом может наблюдаться падение качества ответа.

Методы квантования моделей:
Аффинная квантизация (grid uniform with scale and zero point)
Статья SmoothQuant: https://arxiv.org/abs/2211.10438 (более универсальный квантизуются веса и активации)
Статья GPT-Q (OPT-Q): https://arxiv.org/abs/2210.17323 (квантизуются только веса – пример фреймворка llamacpp)Разница между bf16 и fp16:
bf16 имеет 8 бит под экспоенту (как у fp32) и 7 бит под мантиссу, имеет меньшую точность из-за меньшего количества бит под мантиссу, но сохраняет более широкий диапазон значений. Она лучше подходит для обучения моделей, так как реже возникают переполнения, но поддерживается не всеми GPU.
fp16 имеет 5 бит под экспоненту и 10 бит под мантиссу, более высокую точность, но есть риск переполнения/исчезновения градиентов.

Данные:
50 млн пользователей
Товарные эмбединги (текстовые описания товаров)
Пользовательские эмбединги
Категории товаров
Дополнительные данные: sql, spark

Метрики:
Бизнес метрики (онлайн метрики):
- конверсия в покупку через чат,
- конверсия в добавление в избранное из чата,
- ARPU доход за период / количество пользователей,
- CTR = количество кликов / количество показов
Оффлайн метрики:
- точность/полнота на топ 10, NDCG.
- бинарная оценка релевантности/нерелевантности ответа от пользователя/фокус группы ассесоров.
- loss.
- более сложая ассесорская разметка на отобраном датасете (например, G-eval) или прокрашивание.
- бенчмарки (для выбора llm)

Архитектура:
1. Оценим имеющиеся в контуре llm модели на внутренних бенчмарках (если предполагать что они есть). Насколько я знаю, у WB есть свой лидерборд.
2. Либо можно выбрать самые известные и выбивающиеся в топ модели.
Выбирать можно на основнии лидерборда hugging face:
https://huggingface.co/collections/open-llm-leaderboard/open-llm-leaderboard-best-models-652d6c7965a4619fb5c27a03
Так же есть лидерборд от llmarena:
https://lmarena.ai/?leaderboard=

Да и в целом есть модели, про которые часто пишут в сообществах, такие как qwen, llama, mistral, saiga-mistral, gemini, deepseek, yandexgpt.

Общая архитектура (есть несколько вариантов):
1. Пользователь задаёт вопрос.
2. Вопрос от пользователя трансформируется в поисковый запрос.
3. Поисковая система выдает релевантные документы, из них отбираем топ 10 + еще можно подмешать не из топа.
4. Генеративная модель получает на вход текста запроса пользователя и топ товаров с полезными фрагментами текста и из них составляет ответ.

Есть несколько вариантов как можно реализовать пункт 3.

Бейзлайн:
Считаем текстовые эмбеддинги запроса, считаем скор схожести между текстовыми эмбеддингами запроса и текстовыми эмбеддингами товаров (RAG).
Передаем полученные в топ 10 результаты на llm модель.

Прежде чем делать более сложное дообучение можно попробовать на llm один из методов peft (изменение только части параметров) – LoRA, IA3, Adapter, Prefix-Tuning.
Первый вариант.
Можно попробовать решить задачу получения информативных представлений (metric learning) – получение эмбедингов из конечного состояния нейросети. По полученным эмбеддингам мы можем ранжировать и получать поисковую выдачу.

Возьмем запрос и документ (ответ на запрос), как два текста, которые мы хотим использовать в этом домене. Это два текста которые мы хотим рассматривать как два примера query&doc.
Мы хотим обучить какие-то эмбеддинги, которые нам будут говорить, когда query похоже на документ и когда query не похоже на документ. У нас есть позитивные и негативные примеры по разметке. Негатив - это случайный пример, который является ответом на другой query. margin - расстояние которое может быть между позитивами и негативами.
Для обучения используем triplet loss.

1) Сделаем одну нейросеть с двумя головами (или две нейросети), одна голова нужна для эмбеддингов запроса query, а другая для эмбеддингов документа doc.
2) Обучаем две головы одновременно с задачей получения информативных эмбеддингов.

a - anchor, p-positive, n- negative, d(.,.) - distance function, m - margin
L(a, p, n) = max(d(a, p) - d(a, n) + m, 0)
Learning Embediings with Triplet Loss: https://arxiv.org/pdf/1810.04652

Второй вариант:
Делаем еще одну модель (SFT модель), которая может перевести диалог пользователя в текстовый запрос.
Дообучаем ее по аналогии с алгоритмом ниже** на возвращение короткого релевантного запроса для поиска.
Дальше этот запрос отправляем в существующую поисковую систему и отбираем в ней топ вариантов.
Дальше мы можем брать тексты из отобранных вариантов, ранжировать по скору близости к запросу пользователя и с кусочками и ссылками передавать на llm.
Для того чтобы сохранять ссылки на товары нужна база данных, которая в своих метаданных хранит ссылку, относящуюся к тексту (пример такой базы - ChromaDB).

****Мы можем дообучить нашу llm модель под необходимый нам формат ответа:**
1. Возьмем несколько тысяч запросов и составим для них идеальные ответы при помощи ассесоров.
2. Обучим SFT модель на том что получили в 1.

Мы получили какой-то бейзлайн для нашей дообученной модели.
***Дальше, мы можем использовать различные методы aligment для того чтобы эту модель улучшитью
Например такие методы:
кросс-энтропийный метод (более простой),
градиент по политике (PPO)
DPO (более простой, чем PPO, но лучше работает на практике).
Proximal Policy Optimization, https://arxiv.org/abs/1707.06347
Direct Preference Optimization https://arxiv.org/pdf/2305.18290.pdf

3. Возьмем десятки тысяч длинных запросов от пользователей и для каждого сгенерируем множество пар ответов с помощью модели из 2.
4. Для каждой пары разметим с помощью ассесоров или пользователей, какой ответ лучше, а какой хуже.
5. На этой разметке из 4 обучим reward модель (дистилляция ассесоров в нейросеть при помощи модели Бредли Терри (например, для ранней остановки при обучении SFT

модели)).
6. Возьмем десятки тысяч диалогов и на каждый из них сформируем несколько сотен вариантов ответа, используя различные версии генеративной модели, и семплируя с разными температурами.
7. Скорим ответы при помощи reward-модели.
*8. Поверх модели из предыдущего шага запускаем несколько стадий методов из ***, пусть будет DPO.*

Модель Бредли-Терри:
https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%91%D1%80%D1%8D%D0%B4%D0%BB%D0%B8_%E2%80%94_%D0%A2%D0%B5%D1%80%D1%80%D0%B8

Третий вариант:
Повторение архитектуры рекомендательной системы.
Сначала делаем генерацию кандидатов при помощи als (взаимодействия пользователей).
Затем ранжируем кандидатов – для этого нам пригодятся эмбеддинги пользователей и эмбеддинги товаров.
Релевантные пользователю товары сравниваем по текстовой близости с запросом пользователя и возвращаем топ 10 максимальных по близости и релевантности пользователю.

Инференс:
tensorRT или triton, имеют преимущество, так как они поддерживаются и разрабатываются компанией Nvidia (то есть меньше ошибок, стабильность релизов, качество).
Есть vllm и sqlang – это open source проекты.

Про сравнение фремворков для llm:
https://sersavvov.com/blog/7-frameworks-for-serving-llms

Сравнивать нужно по нескольким основным метрикам: Tokens per Second, Query per Second, latency. Также учитывать отсутствие ошибок и утечек памяти.

Методы ускорения инференса (как правило уже есть в функциональности фреймворков):
1) Распараллеливание на несколько GPU машин.
2) Квантизация (описано выше).
3) Дистилляция знаний большой модели в маленькую. Минимизация лосса между фичами (предсказаниями) учителя и студента. Методы: hard-label, Soft-label, KL – дивергенция и MiniLLM – по метрикам в статье работает лучше других и покрывает их минусы.
MiniLLM: https://arxiv.org/abs/2002.10957
4) peft метод + использование внешнего api.
5) Speculative Decoding: https://arxiv.org/abs/2302.01318

6)Тензорный параллелизм и дата параллелизм https://arxiv.org/pdf/1909.08053
1. **Data Parallelism on Batches**: Splits the input data across multiple devices, with each device processing a separate batch and synchronizing gradients to update the model.
2. **Tensor Parallelism**: Distributes individual layers of the model (e.g., weights or computations) across multiple devices to handle larger models.

3. **Pipeline Parallelism**: Splits the model into stages, with each stage running on a different device, allowing sequential processing of data across devices.

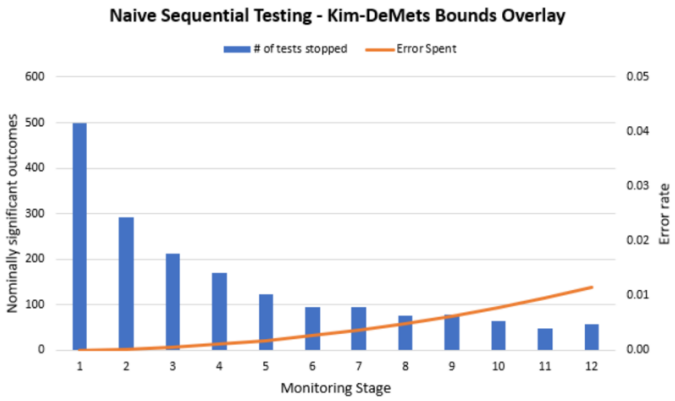
7) использование kv-cache
8) Mixed chunk

Мониторинг:
Weights&Biases, grafana, langfuse.

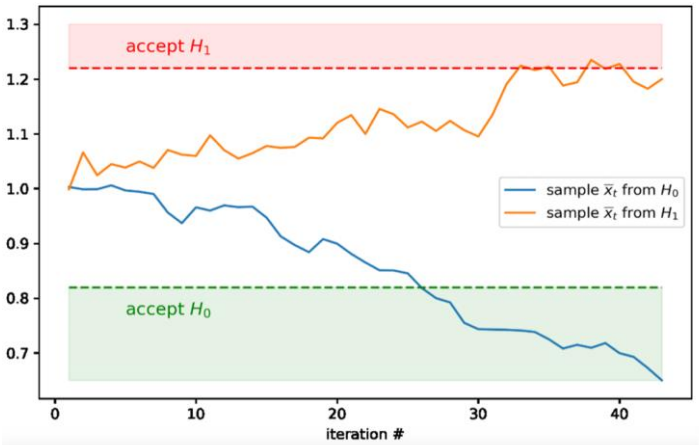
АБ тест качества модели:
Разделить пользователей на две репрезентативные группы из общего числа пользователей
Провести АА тест (нет эффекта без воздействия).
Далее нужно определить длительность теста, можно определить контрольные точки замера метрик.
Запустить тест и сравнивать метрики.

Если мы хотим не фиксированный промежуток времени, а раннюю остановку теста (чтобы не тратить много ресурсов), то нам нужно использовать методы,которые будут решать проблему подглядывания.
Первый вариант, постепенно увеличивать p-value и останавливаться только при явном наличии изменений.
Здесь мы сможем остановиться раньше только при явном наличии эффекта.

Stage	z	p-value
1	4.02	.00003
2	3.53	.00021
3	3.22	.00064
4	2.98	.00144
5	2.78	.00272
6	2.61	.00453
7	2.45	.00714
8	2.3	.01072
9	2.16	.01539
10	2.03	.02118
11	1.9	.02872
12	1.78	.03754



Другой вариант теста, при котором возможна ранняя остановка с вердиктом отсутствия эффекта. Это критерий Ваальда, в котором используется отношение правдоподобий K для n испытаний. Нулевая гипотеза эффекта нет.



Зададим положительные константы A и B: A < 1 < B Если K > B, то отклоняем нулевую гипотезу и останавливаемся. Если K < A, то принимаем нулевую гипотезу и останавливаемся. Иначе продолжаем собирать данные.