Центральный университет
Направление ИИ

ANN: SotA review
HNSW vs Brute Force with GPU

10 октября 2024 г.

Автор:
Степан Высоцкий

Содержание

1 Abstract

Построение индекса - это задача построение такой упорядоченный структуры данных, которая учитывает согласно гипотезе компактности некоторую близость объектов, будь то метрическая, семантическая и т.д.. Задача поиска ближайших соседей или similarity search, то есть поиска по близости это построить этот индекс.

Так, один из важнейших типов моделей в рекомендациях на данный момент — это двух-башенные (two tower) нейросети (1). Выходы этих башен — это эмббединги, которые потом складываются в метрические расстояния, dot product или косинус.

Данные сети по сути частный случай матричной факторизации, принимая на вход только user id и item id. Позднее связывание идеально подходит для задачи кандидатогенерации. Чтобы построить рекомендации одному пользователю, нам нужно один раз посчитать запросную башню, после чего умножить этот эмбеддинг на эмбеддинги документов, которые обычно заранее предподсчитаны. Это очень быстро. Более того, эти предподсчитанные эмбеддинги документов можно сложить в индекс (например, HNSW), чтобы по ним быстро, не проходясь по всей базе, находить хороших кандидатов.

Оказывается, что использовать аппроксимацию ближайших соседей с текущими вычислительными ресурсами необязательно. То приближение ближайших соседей, которые мы получаем от индекса хорошо справляется с задачей поиска, однако это все же приближение и потеря качества ранжирования присутсвует. Современные вычислительные ресурсы, GPU позволяют проводить операции с матрицами эффективно. Данный обзор содержит достаточно весомый вывод об использовании Brute Force GPU NN приближенных соседей в задаче кандидатогенерации с некотороыми оговорками.

2 О проблеме построения индекса

С первого взгляда может показаться, что никакой проблемы нет: действительно, можно ведь просто перебрать все объекты из обучающей выборки $X = \{(x_i, y_i)\}_i$, посчитать для каждого из них расстояние до тестового объекта и затем найти минимум.

Однако несмотря на то, что сложность такого поиска линейная по ℓ, она также зависит и от размерности пространства признаков d. Сложность полного перебора имеет ассимптотику $O(ND)$. Добавив, что размерность эмбеддингов может быть порядка сотни, становится ясно, что такая сложность никуда не годится.

Проблема осложняется ещё и тем, что данный поиск необходимо выполнять на этапе инференса модели, который должен быть быстрым. Всё это означает, что возникает необходимость в более быстрых методах поиска ближайших соседей, чем простой перебор. Приближенные методы дают выигрыш в скорости поиска ближайшего соседа. Так, HNSW (2) это SotA-индекс, который основан на графовых сущностях наших объектов.

Для оптимальности NSW(Navigable Small World) должны выполняться предположения:

- Правило 6 рукопожатий: между любыми двумя точками существует короткий путь, или, более формально, матожидание числа кратчайшего пути между двумя случайно выбранными вершинами растёт как $O(logN)$.
  На самом деле, можно говорить об этом предположении как о всем известном правиле 6 рукопожатий. Так, для графов сущностей социальных сетей данное предположение уже выполняется на $\sim$3 связях (VK). Многие графы имеют такую природу.

- Средняя степень вершины мала. Можно быстро пройтись по соседям.

Иерархический NSW - HNSW - это улучшение базовой модели, описанной выше, которая решает проблему плотных кластеров: модели будет сложно выбраться, так часто плотные кластера связывают объекты-ботлнеки (popularity item, users). Иерархия позволяет доставать в слоях остав структуры: попадая в нижний слой, мы чаще всего оказываемся уже

в нужном кластере и просто уточняем результат работы алгоритма. Гарантии на время сходимости на инференсе хорошие. Подробнее в оригинальной статье (2).

Однако построение такого индекса занимает много времени. Также требуется перестройка структуры для добавления новых объектов в индекс. Также стоит сказать о памяти, так как необходимо хранить графовые представления обо всех объектах в выборке.

3 Составляющая сравнения

В последних исследованиях Meta(организация признана экстремистской и запрещена на территории РФ) (4) и LinkedIn (3) показали, что в современном мире, благодаря вычислительным ресурсам, на этапе отбора кандидатов можно использовать GPU Brute Force NN и у этого есть свои преимущества.

При небольшой размерности эмбеддингов, да ещё и в квантизованном виде, на одной карточке A100 можно хранить порядка 100 миллионов документов и успевать с ними со всеми посчитать скалярное произведение за несколько десятков миллисекунд. Также можно использовать операции матричного перемножения, собирая итемные эмбеддинги в батчи. Но главный вопрос, зачем и что по вычислительной сложности?

Какие у этого преимущества?

1) Полнота поиска выше. Как бы мы ни любили ANN, их полнота на практике вычислений выше 95%, но всё-таки не 100%. А тут мы считаем brute force произведение по всей выборке: результат наилучший.

2.1) Если обычно мы отбираем одну или несколько тысяч кандидатов из ANN, то здесь можно выдавать сразу 100'000, мы не ограничены числом соседей. ANN с таким количеством работают уже не очень хорошо(Достаточно посмотреть на выдачу HNSW).

2.2) Только вот что делать дальше с этими 100000? Мета предлагает на следующей стадии ранжировать их моделью потяжелее, mixture-of-logits MoL (двух-башенная, но в конце не произведение, а более обученная метрика). Ее обучают тоже на GPU! И уже результат этого выдавать в среднее или тяжелое ранжирование, как и раньше.

3) Такой подход позволяет намного быстрее и чаще обновлять эмбеддинги документов. Их просто нужно обновить в памяти GPU, в ANN-индексе же это сложнее, поэтому обычно индексы в проде обновляют реже.

Подробнее с результатами можно ознакомиться в соответствующих статьях.

Делая вывод, хочется подчеркнуть, что это будет стоить дороже, чем знакомый нам ANN-индекс. Однако, на что мы готовы пойти ради качественных рекомендаций? Модель с быстрым перебором работает эффективнее, однако потратиться увы придется. Можно заметить, что ANN оказался оптимизацией, вполне качественной, однако эта оптимизация несет в себе тот самый трейд-офф скорости и качества. Раньше скалярные произведе-

ния на процессоре можно было делать до 100к-1М итемов. А теперь вот получается, что их 100М. Также важно сказать, что метод дорогой, если будет наблюдаться большая пользовательская нагрузка на рекомендательную систему.

4 Список Литературы

(1) https://dl.acm.org/doi/10.1145/2959100.2959190 Deep Neural Networks for YouTube Recommendations, RecSys '16: Proceedings of the 10th ACM Conference on Recommender Systems, 2016, Google, Paul Covington, Jay Adams, Emre Sargin
(2) https://arxiv.org/pdf/1603.09320 Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs, IEEE, Institute of Applied Physics of the Russian Academy of Sciences, Yu. A. Malkov, D. A. Yashunin
(3) https://arxiv.org/abs/2407.13218 LiNR: Model Based Neural Retrieval on GPUs at LinkedIn, LinkedIn Fedor Borisyuk, Qingquan Song, Mingzhou Zhou, Ganesh Parameswaran, Madhu Arun, Siva Popuri, Tugrul Bingol, Zhuotao Pei, Kuang-Hsuan Lee, Lu Zheng, Qizhan Shao, Ali Naqvi, Sen Zhou, Aman Gupta
(4) https://arxiv.org/abs/2306.04039 Revisiting Neural Retrieval on Accelerators, 2023, Meta, Jiaqi Zhai, Zhaojie Gong, Yueming Wang, Xiao Sun, Zheng Yan, Fu Li, Xing Liu