Ответ 3

В чем заключается синергия между квантованием (scalar и product quantization) и механизмом re-scoring в Qdrant, и как это позволяет одновременно оптимизировать использование памяти и поддерживать высокую точность поиска в больших коллекциях эмбеддингов?

Синергия между квантованием и механизмом re-scoring в Qdrant представляет собой двухэтапную стратегию, которая позволяет эффективно решать фундаментальную проблему хранения и поиска в огромных коллекциях векторов: необходимость одновременной оптимизации использования памяти и поддержания высокой точности.

1. Фундаментальная проблема: Стоимость хранения векторов

Как указано в учебном материале, современные эмбеддинги имеют высокую размерность. Например, вектор размерностью 1536 в стандартном формате `float32` занимает около 6 КБ. Для коллекции из миллиарда таких векторов потребуется 6 ТБ оперативной памяти, что является непомерно дорогим. Квантование напрямую решает эту проблему.

2. Шаг 1: Оптимизация памяти с помощью квантования

Квантование — это процесс снижения точности числовых данных для уменьшения занимаемого ими объема памяти и ускорения вычислений. Qdrant использует два основных метода:

- **Скалярное квантование (Scalar Quantization, SQ):**
    - **Механизм:** Каждое число `float32` (32 бита) в векторе преобразуется в `int8` (8 бит). Это достигается путем нахождения минимального и максимального значений для каждой размерности по всей коллекции и линейного отображения этого диапазона в диапазон `[-127, 127]`.
    - **Результат:** 4-кратное сокращение объема памяти. Это позволяет хранить в 4 раза больше векторов в том же объеме RAM.
- **Продуктовое квантование (Product Quantization, PQ):**
    - **Механизм:** Этот более сложный метод разбивает длинный вектор на несколько коротких под-векторов. Для каждого набора под-векторов с помощью кластеризации создается "кодовая книга" из типичных представителей (центроидов). Затем каждый под-вектор заменяется ID ближайшего к нему центроида.
    - **Результат:** Вместо хранения, например, 1536 чисел `float32`, система хранит 96 чисел `uint8`. Это обеспечивает сжатие до 64 раз и более.

Основная цель этого шага — радикальное уменьшение объема памяти. Поиск (например, с помощью HNSW) выполняется по этим сжатым, квантованным векторам, которые могут быть полностью загружены в RAM. Однако этот процесс сопряжен с компромиссом — потерей точности, поскольку квантованные векторы являются лишь приближением оригинальных.

3. Шаг 2: Восстановление точности с помощью Re-scoring

Механизм `re-scoring` в Qdrant предназначен для того, чтобы нивелировать потерю точности, внесенную квантованием. Он вступает в игру после завершения первоначального поиска.

- **Процесс:**
    1. **Быстрый поиск кандидатов:** Qdrant выполняет быстрый ANN-поиск по квантованным векторам. Цель этого этапа — не найти абсолютно точные результаты, а быстро отобрать небольшое количество "топ-N кандидатов", которые с высокой вероятностью являются релевантными. В материале этот этап также называется "грубым поиском".
    2. **Точный пересчет:** Затем система берет этот ограниченный список кандидатов и для них извлекает **оригинальные, полные `float32` векторы, которые хранятся на диске**.
    3. **Финальное ранжирование:** Расстояния (или сходство) между запросом и векторами кандидатов пересчитываются с использованием полных, точных представлений. Результаты сортируются на основе этих точных значений, и пользователю возвращается итоговый, высокоточный список.

4. Синергия: Сочетание скорости, экономии памяти и точности

Синергия заключается в том, что эти два механизма работают вместе, чтобы достичь того, что было бы невозможно по отдельности:

- **Квантование** решает проблему **памяти и скорости**. Оно позволяет хранить индекс в RAM и быстро выполнять поиск по миллиардам векторов.
- **Re-scoring** решает проблему **точности**. Он гарантирует, что финальные результаты, возвращаемые пользователю, имеют высокую точность, поскольку они ранжируются на основе исходных, неискаженных данных.

Таким образом, Qdrant не заставляет выбирать между экономией памяти и точностью. Система использует квантованные векторы для быстрой и экономичной навигации по огромному векторному пространству, чтобы сузить круг поиска до нескольких десятков или сотен наиболее вероятных кандидатов. Затем она применяет вычислительно более дорогие, но точные расчеты только к этому небольшому подмножеству, эффективно сочетая "скорость и экономию памяти квантованного поиска с точностью полного представления".