Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Илья Игоревич Келим

Добавление дополнительной функциональности в библиотеки поиска ближайших соседей

Магистерская диссертация

Научный руководитель:
доцент кафедры СП, к. т. н. Брыксин Т. А.

Рецензент:

Консультант:
JetBrains Research Литвинов Д. В.

Санкт-Петербург
2020

Оглавление

Введение

Проблема поиска ближайших соседей представляет из себя задачу нахождения по заданному данному вектору наиболее близких к нему по заданной метрике других векторов из некоторого заранее фиксированного списка. В индустрии эта проблема часто возникает в рекомендательных системах, системах поиска, машинном обучении, компьютерном зрении и многих других областях.

Для поиска ближайших соседей в основном используются два подхода: полный перебор и приближенный поиск. Полный перебор предполагает подсчет выбранной метрики для пар из данного вектора и всех векторов и списка. Затем пары сортируются по подсчитанной метрике и в качестве результата выдаются вектора из списка, состоящие в парах с минимальным значением метрики.

Для работы с большими объемами данных за меньшее время часто используются алгоритмы приблизительного поиска ближайших соседей. Этот класс алгоритмов предполагает построение структуры данных, позволяющей находить ближайших соседей с некоторой заданной точностью. Этот подход позволяет значительно сократить время работы и требует значительно меньше ресурсов, чем точные аналоги. Например, полный перебор работает за $O(d * n)$, где d — это размерность векторов, а n — количество векторов в индексе. Алгоритм приближенного поиска "KD дерево" за $O(n_0 + \log_2(n/n_0))$, где высота дерева $h = \log_2(n/n_0)$. Благодаря этому алгоритмы приблизительного поиска ближайших соседей могут использоваться в системах, требующих выдачи быстрого ответа, но не настолько высокой точности результата. Например, этот подход хорошо подходит для рекомендательных систем.

Для реализации подхода приближенного поиска ближайших соседей существует множество библиотек, написанных на языке Python. И у большинства популярных реализаций можно выделить две концептуальные проблемы: индекс занимает большой объем оперативной памяти и отсутствует возможность динамически добавлять и удалять вектора из него. Первая проблема возникает с ростом объема данных. Вторая

проблема возникает, например, в рекомендательных системах реального времени, контент в которых может обновляться не централизованно. После анализа информации, доступной о популярных библиотеках, стало понятно, что большинство библиотек не планирует развивать свои реализации в сторону решения этих проблем в ближайшем будущем.

Постановка задачи

Цель этой работы — разработать надстройку над классом библиотек поиска ближайших соседей, позволяющую быстро динамически обновлять индекс и уменьшать его размер.

- Исследовать алгоритмы и структуры данных, применяемые для приблизительного поиска ближайших соседей.
- Исследовать библиотеки на языке Python, предоставляющие функциональность поиска ближайших соседей.
- Разработать надстройку над самыми популярными библиотеками поиска ближайших соседей написанных на языке Python, позволяющую динамически добавлять и удалять вектора из списка.
- Разработать надстройку над самыми популярными библиотеками поиска ближайших соседей написанных на языке Python, позволяющую сжимать размер хранимой структуры.

1. Обзор

1.1. Библиотеки поиска ближайших соседей

В этом разделе рассмотрены самые популярные библиотеки приблизительного поиска ближайших соседей для языка Python, поддерживающие сохранение структуры поиска. Библиотеки, не поддерживающие сохранение, не представляют интереса для этой работы, так как они не применяются в индустрии для больших объемов данных и предлагаемые оптимизации не актуальны для них.

Annoy

Annoy[1] это библиотека, предоставляемая компанией Spotify[2]. Annoy позволяет строить и сохранять индекс, но не позволяет динамически обновлять его. Используемый для реализации алгоритм случайной проекции автоматически сжимает размер индекса, однако оставляет возможность для улучшения.

Flann

Библиотека Flann[3] также позволяет строить и сохранять индекс без возможности динамического изменения. Эта библиотека предоставляет список алгоритмов, из которого для каждого конкретного случая выбирается подходящий. Только некоторые представленные алгоритмы позволяют снизить размер индекса, однако, так как это не является основной целью алгоритмов, имеющиеся реализации оставляют простор для улучшения.

[1] `https://github.com/spotify/annoy`
[2] `https://www.spotify.com/`
[3] `https://github.com/flann-lib/flann`

NMSLIB

Библиотека NMSLIB[4] предоставляет те же возможности, что и две описанные выше библиотеки. Предоставляемые алгоритмы не снижают размер индекса в принципе.

Faiss

Библиотека Faiss[5] разработанная компанией Facebook[6] предоставляет возможность построение, сохранение и динамического обновления индекса. Однако динамическое обновление реализовано через перестроение индекса, поэтому использования его для добавления или удаления векторов по одному не оптимально по скорости. Некоторые предоставляемые алгоритмы сжимают размер индекса, однако большая их часть не дает такой возможности.

[4] https://github.com/nmslib/nmslib
[5] https://github.com/nmslib/nmslib
[6] https://www.facebook.com/

2. План работы

- Исследовать алгоритмы и структуры данных, позволяющие решить поставленные задачи.
- Реализовать прототип, работающий со всеми популярными библиотеками.
- Оценить скорость работы прототипа по сравнению с исходными библиотеками.
- Подобрать алгоритм и структуру данных для прототипа, дающие оптимальный результат.

Список литературы