А.А. ПОНОМАРЕНКО,
Н. С. АВРЕЛИН, Б. НАЙДАН,
Л. М. БОЙЦОВ

Сравнительный анализ структур данных для приближенного поиска ближайшего соседа

ФГАОУ ВПО
«Национальный
исследовательский
университет
«Высшая
школа экономики»,
г. Нижний Новгород

Поиск по похожести широко применяется в различных областях компьютерных наук. Множество методов было предложено для решения задачи в точной постановке, однако все они подвержены "проклятью" размерности и не эффективны для данных высокой размерности. Приближенные алгоритмы отчасти позволяют справиться с "проклятьем". Однако из-за сложной стохастической природы, теоретические оценки для большинства приближенных алгоритмов отсутствуют. Более того, на данный момент времени, в литературе не существует работ, включающих всесторонний эмпирический анализ современных методов для поиска по подобию. Как правило, авторы алгоритмов ограничиваются небольшими численными экспериментами, сравнивая свой алгоритм с одним ранее известным методом. С целью устранения этого пробела в научном знании, в настоящей работе мы приводим результаты такого эмпирического анализа для методов: Vantage Point Tree, Locality Sensitive Hashing, List of Clusters, Метризованный Тесный Мир и несколько вариаций Permutation Index. Проведенные эксперименты показывают, что Метризованный Тесный Мир имеет наилучшее соотношение между вычислительной сложностью и

точностью, как для метрических, так и для неметрических пространств.

Введение

Задача поиска по похожести, формализуемая как поиск ближайшего соседа, является фундаментальной проблемой компьютерных наук имеющее широкое применение в компьютерном зрении [2], коллективной фильтрации [3], в рекомендательных системах и системах поиска паттернов [1]. Поиск по подобию состоит в нахождении наиболее похожего объекта в конечном множестве объектов X (в базе) к заданному объекту q (запросу). Степень похожести двух объектов $x, y \in X$ вычисляется, используя функцию расстояния или близости $d(x, y)$. Полагают, что чем меньше расстояние между объектами, тем больше они схожи (ближе) между собой. Когда функция d удовлетворяет трём аксиомам метрики: (1) позитивность (2) симметричность; (3) неравенство треугольника, то принято говорить о поиске в метрическом пространстве (X, d). Если хотя бы одна из аксиом может нарушаться, то употребляют термин "поиск в неметрическом пространстве" (non-metric space search).

Задача поиска ближайшего соседа заключается в поиске самого ближайшего объекта к запросу. Обобщением этой задачи является задача поиска k-ближайших соседей. В своей точной постановке она требует от алгоритма возвратить все k самых близких объектов к запросу.

Очевидно, что задача поиска ближайшего соседа имеет тривиальное решение полным перебором. Вычислив значение функции близости d до каждого объекта из X, станет ясно какой объект ближайший к q. Однако функция вычисления степени близости может быть вычислительно затратной операцией, а множество X велико. Поэтому, в литературе под термином "поиск ближайшего соседа" понимают построение такой структуры данных S на множестве X, чтобы операция поиска k-ближайших соседей имела как можно меньшую вычислительную сложность, что обычно соответствует, уменьшению количества вычислений функции близости.

Множество методов (структур данных) было предложено для решения задачи в точной постановке. Однако все они эффективны только для пространств невысокой размерности. (Под размерностью пространства мы имеем в виду количество компонент у векторного пространства [4]). Практика показывает, что точные методы по производительности не могут превзойти полный перебор на данных с размерностью больше десяти [5]. В литературе данный феномен называют "проклятьем размерности". Переход к приближенной версии задачи, допустив неточность поиска, позволяет в некоторой степени снять "проклятье" [18][25].

Одним из общепринятых способов оценки точности поиска является *recall*. Значение *recall* вычисляется как $\frac{|\{x : x \in knn(q, X)\}|}{k}$, где $knn(q, X)$ - множество истинно ближайших объектов к запросу q во множестве X.

На сегодняшний день было предложено множество структур данных, как для точного, так и для приближенного поиска k-ближайших. Хороший обзор методов для точного поиска приведён в [4] и методов для поиска в неметрическом пространстве [11]. Однако для многих методов отсутствуют теоретические оценки. Поэтому, основным способом выявления наиболее эффективной структуры данных, остаётся запуск методов на общих наборах данных с использованием одинакового оборудования. Во время экспериментов измеряется время работы и количество обращений к функции близости d на этапах предобработки (индексирования) и исполнения запросов.

Ранее сообществом поддерживалась библиотека методов для точного поиска ближайшего соседа "metric space library" [10]. Однако с пониманием неэффективности точных методов на практике, мотивация поддерживать данную библиотеку исчезла. Тем не менее, недавно была создана открытая библиотека “Non-Metric Space Library” [12], включающая методы для приближенного поиска, различные функции близости, включая неметрики [14][15], и наборы данных. В данной работе мы приводим результаты сравнения методов, используя эту библиотеку.

Наш вклад заключается в том, что мы реализовали и сравниваем в рамках данной библиотеки недавно предложенный

метод [9], который демонстрирует в большинстве случаев наилучшее соотношение точности и производительности.

Статья организованна следующим образом. В первой части приведено краткое описание сравниваемых методов, функций близости и наборов данных. Во второй, описание экспериментов и результаты.

Сравниваемые методы

Vantage Point Tree.

Vantage Point Tree (VPT) – метод основанный на иерархическом разбиении пространства, который на этапе поиска использует неравенство треугольника для исключения областей заведомо не содержащих запрос [16][17]. Оригинальная версия этого метода поддерживала только точный поиск. Тем не менее, ослабление неравенства треугольника, допуская «растяжение» [18], позволяет использовать данную структуру данных для приближенного поиска, как в метрическом, так и в неметрических пространствах [7].

Оптимальное значение коэффициента растяжения определялось экспериментально. Сначала строилась структура из небольшого числа случайно выбранных объектов; производился поиск 10 ближайших элементов, используя различные значения коэффициента растяжения; замерялась производительность и точность поиска. Далее, выбиралось значение коэффициента, при котором достигалась наибольшая скорость для заданного значения точности.

Методы основанные на перестановках (Permutation Methods).

Идея методов основанных на перестановках, заключается в том, что близкие к друг другу объекты метрического пространства будут "видеть" фиксированное множество $P = \{p_i \in D\}_{i=1}^{m}$, называемых опорными объектами (pivots) в одном и том же порядке. Эта последовательность рассматривается в качестве перестановки, которую можно представить в виде целочисленного вектора. Так каждый объект $x \in X$ представляется перестановкой Π_x элементов множества P. Перестановка Π_x соответствующая объекту x, определяется порядком следования опорных точек в порядке

возрастания расстояния $d(x, p_i)$. Первая позиция в перестановке, соответствует номеру самого близкой опорного объекта к объекту x, последняя - номеру самого удалённого.

Данный подход позволяет сократить размерность и помогает перейти к векторному представлению данных, в случае когда объекты метрического пространства изначально не имеют координатного представления.

На этапе исполнения запроса сначала вычисляется расстояние между запросом q и опорными точками P, строится перестановка Π_q. Далее, проиндексированные объекты, сортируются в порядке похожести перестановок относительно перестановкиΠ_q. В свою очередь, похожесть перестановок можно определять различными способами. Это приводит к различным вариациям пермутационных методов. Один из первых пермутационных методов был предложен независимо Чавесом и соавторами [19], а так же Амато и Савино [20].

В базовой версии опорными точками является множество случайно выбранных объектов из множества X. Перестановки хранятся в виде массива и вычисляются для каждого объекта из X. На этапе поиска вычисляется похожесть (обычно используется евклидово расстояние) между перестановкой Π_q и каждой перестановкой Π_i. Формируется список объектов кандидатов, сортируемый относительно близости их перестановок к перестановке Π_q. Выбирается подмножество объектов с наименьшим значением расстояния. До каждого объекта из подмножества вычисляется расстояние, непосредственно, до объекта-запроса q, используя изначальную меру близости d. Среди них выбираются k ближайших к q и возвращаются в качестве результата.

Существует несколько путей улучшить базовую версию. Первый – уменьшить число перестановок близких к Π_q [19]. Второй, - проиндексировать перестановки, вместо того, чтобы искать наиболее билизкую перестановку последовательно. Для этого можно использовать дерево префиксов [21], обратный индекс

[20], либо метод для поиска в метрических пространствах, например VP-Tree [22].

Недавно, было предложено индексировать окрестность опорных объектов, т.е. для каждого объекта $x \in X$ формировать множество P_x наиболее близких опорных точек, такое что $|P_x| << m$ и $|P_x \cap P| = 0$ [23] Идея в том, что близкие объекты x и y имеют не пустое пересечения окрестностей $|P_x \cap P_y| \neq 0$.

Предварительные эксперименты показали, что в зависимости от набора данных, наиболее эффективнами методами основанными на перестановках, являются следующие методы: базовая версия использующая частичную сортировку (incremental sorting), приближенная версия использующая VP-tree в качестве индекса перестановок "permutation (VP-tree)", и вариация индексирующая окресность опорных векторов (pivot neighborhood index).

Метризованный Тесный Мир.

Метризованный Тесный Мир (Metrized Small World; сокр. MSW) – подход использующих в качестве структуры данных граф с навигационными свойствами тесного мира [8][9]. MSW подход предполагает что, каждому объекту из множества X однозначно ставится в соответствие вершина графа. Таким образом, задача поиска ближайшего соседа сводится к задаче поиска вершины на графе, а построение структуры данных заключается в построение графа. В качестве основы для алгоритма поиска используется жадный алгоритм (Greedy Walk). В графе MSW выделяют логически два типа ребёр: короткие – обеспечивают корректность работы жадного поиска и в совокупности рассматриваются как аппроксимация графа Делоне; длинные – обеспечивают существования коротких путей в графе.

Несмотря на логическое выделение двух типов рёбер, формировать их можно единообразно, если стоить граф путём последовательного добавления случайно выбранных объектов из множества X, обеспечивая достаточно хорошую аппроксимацию графе Делоне [24] на каждом этапе. При таком подходе, если не удалять рёбра, сформированные на ранних этапах, то они становятся длинными, если их рассматривать в контексте структуры

после добавления всех элементов из множества X. Более того, так появляются рёбра всех масштабов (длинные, средние и короткие).

В работах [8][9] предлагается стоить аппроксимацию графа Делоне, соединяя каждый добавляемый объект с k-ближайшими объектами, которые уже есть в структуре к моменту добавления.

По причине возможного присутствия в структуре локальных минимумов, для улучшения поиска могут использоваться идеи из дискретной оптимизации. Например, производить поиск использую серию жадных поисков от случайно выбранных вершин [9], или табу поиск [32] и другие.

Locality Sensitive Hashing.

Locality Sensitive Hashing - класс методов использующих хэш-функции принимающих с большой одно и тоже значение для близких объектов и разное для удаленных. На сегодняшний день было предложено множество классов таких функций для различных видов мер близости и распределений данных [13][25]. Метод широко известен за счёт наличия вероятностных оценок и простоте реализации. Однако LSH имеет трудности обработки данных с различной плотностью распределения. Кроме того LSH нельзя использовать для произвольных метрических пространств. Он применим только для определенных семейств функций расстояний, для которых известны эффективные LSH-функции. Тем не менее это очень сильный метод в случае L^p-пространств.

В наших эксперементах мы использовали версию эффективно использующую память multiprobe LSH [13], которая была реализованна в рамках библиотеки LSHKIT [13].

List of Clusters

Список Кластеров (List of Clusters) – метод для точного поиска ближайшего соседа в метрическом пространстве, который основывается на группировании объектов в кластеры [26]. Кластеры строятся последовательно, начиная со случайно выбранного центра кластера. Далее, точки находящиеся от центра кластера на расстоянии не далее чем R или N ближайших точек, ассоциируют с соответствующим кластером. После чего процедура кластеризации повторяется для оставшихся точек. В наших экспериментах мы

использовали стратегию ассоциирования N ближайших к центрам кластеров и центры выбирались случайным образом.

Алгоритм поиска сканирует множество сконструированных кластеров и, используя неравенство треугольника, проверяет, может ли потенциальный ответ лежать в рассматриваемом кластере. Если кластер может содержать ответ, то каждый объект из кластера сравнивается непосредственно с запросом. Далее проверяется, может ли ответ содержаться вне рассматриваемого кластера. Если не может, то поиск останавливается.

В экспериментах мы использовали приближенную версию алгоритма, добавив в алгоритм поиска стратегию ранней остановки. Процесс поиска останавливается после просмотра фиксированного числа кластеров (параметр алгоритма). Множество кластеров рассматривается в порядке возрастания расстояния от запроса до центров кластеров.

Эксперименты

Мы производили сравнения методов на различных наборах данных, используя три различные функции расстояния:

1. Евклидово расстояние (L2);

2. Неметрическая функция расстояния Кульбабка Лейблера (KL-дивергениция) [14]: $d(x,y)=\sum x_i \log\frac{x_i}{y_i}$;

3. Косинус угла между векторами (cosine similarity): $d(x,y)=1-\frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$.

Наборы Данных

- CoPhIR (L2): коллекция 208-мерных векторов извлечённых из изображений в формате MPEG7 [27]. Вектора составлены из пяти различных свойств MPEG7.

- SIFT (L2): часть коллекции данных TexMex [28]. Содержит 1 миллион 128-мерных векторов. Каждый вектор соответствует дескрипторам извлеченных из изображений методом «Scale Invariant Feature Transformation» (SIFT) [29]

- Википедия (cosine similarity): коллекция состоящая из 3.2 миллионов векторов в разряженном формате. Каждый вектор

хранит частоту встречаемости каждого слова в конкретной странице Wikipedia. Вектора были построены при помощи библиотеки gensim [30]. Вектора в этом наборе данных имеют сверхвысокую размерность – более 100 тысяч компонентов. Однако, вектора разряженные. В среднем, каждый вектор имеет только около 150 ненулевых компонент.

- Unif64 (L2): синтетический набор 64-мерных векторов сгенерированных случайно с равномерным распределением в единичном гиперкубе.

- Final16, Final64 и Final256 (KL-дивергенция): набор из 500 тысяч гистограмм заголовков полученных с помощью латентного размещения Дирихле (LDA) [31]. Числовой суффикс названия коллекции соответствует размерности, которая равна количеству LDA-заголовков (топиков). Данные коллекции были созданы Кейтоном [6]

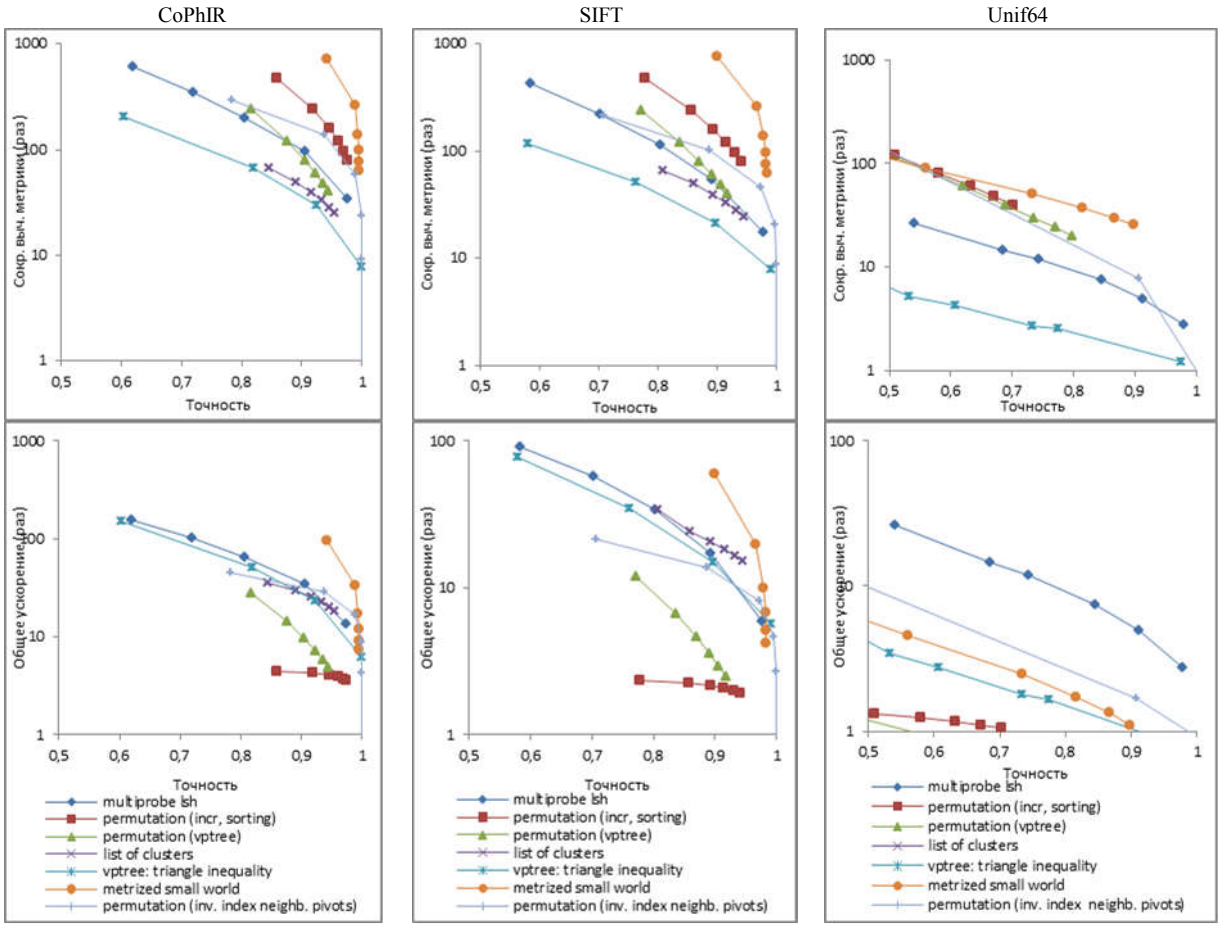


Рис. 1. Результаты экспериментов поиска 10-ближайших с функцией расстояния L2. Графики из одного столбца соответствуют одному набору данных. CoPhIR - первый столбец, SIFT - второй, Unif64 - третий.

Методология оценки

Эксперименты производились на Linux Intel Xeon сервере с частотой 3.60 GHz, 32GB оперативной памяти в однопоточном режиме, используя библиотеку *Non-Metric Space Library* в качестве инструмента для валидации [12]. Программная реализация была написана на C++ и скомпилирована при помощи GNU C++ 4.7 (с ключом оптимизации «-Ofast»).

Мы использовали реализации функций расстояний оптимизированные при помощи SSE 4.2 SIMD инструкций. В случае KL-дивергенции дополнительное ускорение достигалось за счёт предварительного вычисления логарифмов элементов векторов на этапе построения индекса [7]. В реализации функции вычисления косинуса угла между векторами, использовалась инструкция сравнения _mm_cmpinstrm «все-против-всех» . Эта реализация около 2.5 раза быстрее, чем реализация на чистом C++.

Для проведения экспериментов мы случайным образом разделили каждый тестовый набор на две части. Меньшая часть содержала только 1000 объектов и использовалась в качестве множества запросов. Оставшиеся объекты индексировались. После индексации тестировалась производительность методов, производя поиск 10 ближайших элементов (10-NN search). Все методы полностью размещались в оперативной памяти. Процедура поиска повторялась пять раз для различных значений параметров для того, чтобы получить различные значения точности и иметь некоторое усреднение (значение параметров выбирались вручную).

Большинство методов тестировались на всех тестовых наборах данных. Multi-probe LSH и Список Кластеров использовались только для Евклидова расстояния. VP-tree не использовалась для Wikipedia, связи с высокой размерностью тестового набора, на котором VP-tree работала чуть лучше полного перебора.

Результаты сравнения методов для Евклидова расстояния и для KL-дивергенции приведены на рисунках 1 и 2 соответственно. График в первой строке показывает во сколько раз тот или иной метод делает меньше вычислений функции расстояний в сравнении с полным перебором для различных значений точности. Во второй строке по вертикали отложено значение во сколько раз метод быстрее полного перебора (время работы).

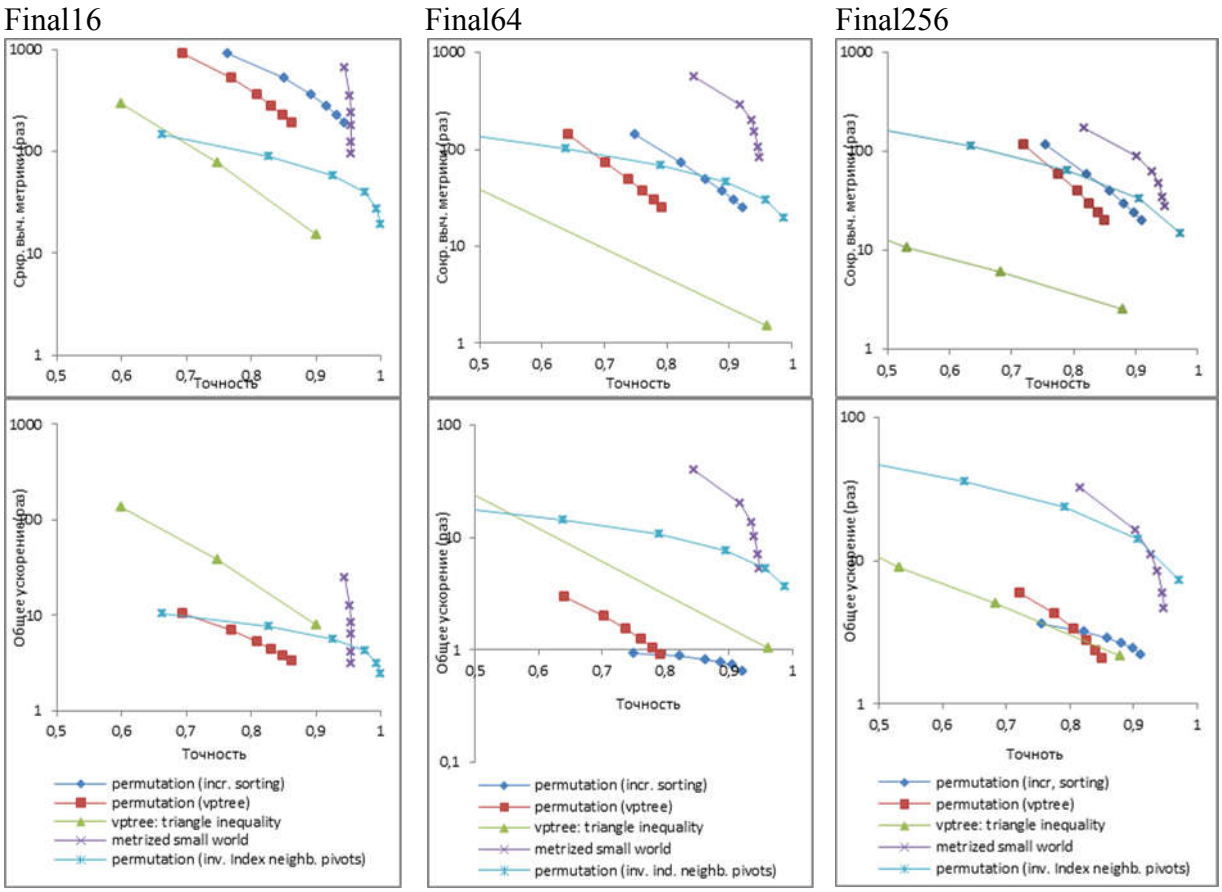
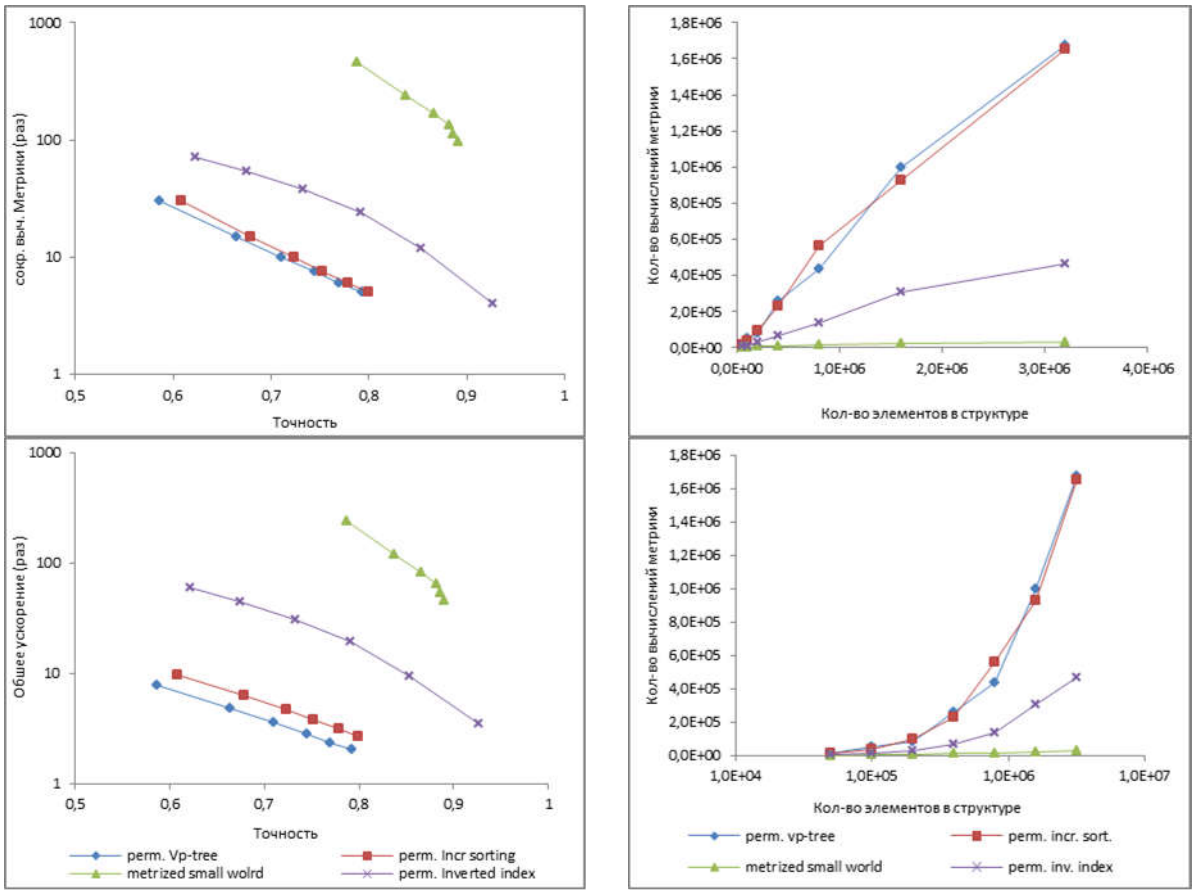


Рис. 2. Результаты экспериментов поиска 10-ближайших с функцией расстояния KL-дивергенции. Графики из одного столбца соответсвуют одному набору данных

Как видно из графиков на рисунках 1 и 2, метод MSW имеет наилучшее соотношение точности и производительности на всех трёх тестовых наборах данных. Рассмотрим, например, тестовый набор данных CoPhIR (Рис. 1). Для значения точности (recall) 0.9 метод MSW вычисляет в 1000 раз меньше расстояний, чем полный перебор. Остальные методы для этого значения точности превосходят полный перебор менее чем в сто раз. Подобную производительность метод LSH имеет только для значения точности 0.4.

Обычно, чем меньше раз метод вычисляет функцию расстояния, тем быстрее он работает. Однако для «недорогих» функций расстояний, например, Евклидова расстояния, скорость работы метода зависит не напрямую от количества вычисленных функций расстояния. Рассмотрим тестовый набор Unif64 на рис. 1 и Final256 на рис. 2: несмотря на то, что MSW использовал меньше

всего вычислений расстояний во всех случаях, однако дополнительные вычислительные расходы, связанные с обходом графа, могут быть высокими. Как результат pivot neighborhood index или multi-probe LSH иногда показывают лучшую итоговую производительность.



(а) Фиксированный набор состоящий из 3 200 000 элементов.

(б) Зависимость скорости поиска от числа проиндексированных элементов.

Рис. 3. Результаты с набором данных Wikipedia. В качестве расстояния использовалась функция косинуса угла между векторами.

Отметим, что MSW и inv neighborhood index хорошо работают для расстояния KL-дивергенции. Для всех трёх наборов данных с функций расстояния KL-дивергенции они показывают ускорение более чем в 10 раз, по сравнению с полным перебором, для значения точности 0.9.

Результаты сравнения методов на данных Wikipedia представлены на рис. 3(а). Верхний график показывает во сколько раз метод совершает меньше вычислений расстояний по сравнению с полным перебором для различных значений точности. Соответственно, нижний график изображает во сколько раз методы

работают быстрее полного перебора для разных значений точности. Несмотря на то, что мы использовали улучшенную реализацию функции вычисления косинуса угла между двумя векторами с помощью инструкций SIMD, которая 2.5 раза быстрее, чем обычная реализацию на чистом C++, вычисление скалярного произведения продолжает оставаться вычислительно затратной операцией. По этой причине сокращение количества вычисленных расстояний, напрямую влияет на общую скорость работы методов.

Как видно из графика, на этом наборе данных MSW работает значительно лучше остальных методов. Например, для точности 0.87 MSW примерно в 40 раз быстрее полного перебора. Следующий по производительности метод (pivot neighborhood index) достигает такой производительности при значительной меньшей точности 0.6.

Для того чтобы понять, как производительность методов зависит от размера набора данных, мы произвели эксперименты с Википедией. Мы производили замеры для подмножеств Википедии, чей размер варьировался от 12.5 тыс. до 3.2 миллионов элементов. На каждом тестовом подмножестве для каждого метода мы запускали процедуру поиска несколько раз с различными параметрами для того, чтобы получить результаты для различных значений точности. Для MSW мы использовали параметры, которые давали 0.9, тогда как для других методов мы использовали параметры, приводившие к точности 0.8. Результаты экспериментов приведены на рис. 3(б). Нижний график включает в себя результаты для всех методов, верхний – только MSW и pivot neighborhood index.

Как видно из рис. 3(б), все методы, основанные на перестановках, имеют зависимость от количества объектов близкую к линейной. В то время как MSW демонстрирует зависимость близкую к логарифмической (добавить график в двойном логарифмическом масштабе) и при этом имеет большую точность (0.9 против 0.8). Таким образом, MSW имеет принципиально другую вычислительную сложность и превосходит в производительности все остальные методы тем больше, чем больше набор входных данных.

Заключение

Используя большое число тестовых наборов, мы провели большой сравнительный анализ методов для приближенного поиска ближайшего соседа. Наши эксперименты включали функции расстояния являющимися метриками, так и функции которые не удовлетворяют аксиомам метрики (KL-дивергенция не симметрична и не удовлетворяет неравенству треугольника). Для того чтобы любой мог легко воспроизвести полученные результаты, мы сделали наш код публичным, в качестве части открытой библиотеки non-metric space library. Все тестовые наборы данных, за исключением CoPhIR, находятся в свободном доступе.

Наши эксперименты показали, что метод Metrized Small World превосходит остальные методы для большинства значений точности. Эксперименты с частотными векторами слов Википедии продемонстрировали логарифмическую зависимость числа вычислений функций расстояний от размера входных данных, что подтвердило ранее полученный результат [8]. Несмотря то, что этот тестовый набор имеет экстремально высокую размерность, MSW позволяет быстро получить результаты поиска с точностью более 90%. Все это даёт нам основания для выдвижения гипотезы о том, что подход, используемый в MSW один из наиболее обещающих методов, как для использования в метрических, так и неметрических пространствах.

В наших экспериментах MSW почти всегда превосходит все остальные методы по числу сокращений вычислений функций расстояний. В случаях, когда функция расстояния не является вычислительно затратной операцией, метод MSW не всегда показывает лучшую общую производительность, данное обстоятельство мы связываем с накладными расходами обхода графа. Поэтому дальнейшие усилия стоит направить на более эффективную реализацию метода MSW.

Благодарности

Первый автор был частично поддержан грантом РНФ 14-41-00039.

Литература

1. Cover T. M., Hart P. E. Nearest neighbor pattern classification //Information Theory, IEEE Transactions on. – 1967. – Т. 13. – №. 1. – С. 21-27.

2. Flickner M. et al. Query by image and video content: The QBIC system //Computer. – 1995. – Т. 28. – №. 9. – С. 23-32.

3. Sarwar B. et al. Item-based collaborative filtering recommendation algorithms //Proceedings of the 10th international conference on World Wide Web. – ACM, 2001. – С. 285-295.

4. Chávez E. et al. Searching in metric spaces //ACM computing surveys (CSUR). – 2001. – Т. 33. – №. 3. – С. 273-321.

5. Weber R., Schek H. J., Blott S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces //VLDB. – 1998. – Т. 98. – С. 194-205.

6. Cayton L. Fast nearest neighbor retrieval for bregman divergences //Proceedings of the 25th international conference on Machine learning. – ACM, 2008. – С. 112-119.

7. Boytsov L., Naidan B. Learning to prune in metric and non-metric spaces //Advances in Neural Information Processing Systems. – 2013. – С. 1574-1582.

8. Malkov Y. et al. Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces //Similarity Search and Applications. – Springer Berlin Heidelberg, 2012. – С. 132-147.

9. Malkov Y. et al. Approximate nearest neighbor algorithm based on navigable small world graphs //Information Systems. – 2014. – Т. 45. – С. 61-68.

10. Figueroa K., Navarro G., Chávez E. Metric spaces library //Online http://www. sisap. org. – 2007.

11. Skopal T., Bustos B. On nonmetric similarity search problems in complex domains //ACM Computing Surveys (CSUR). – 2011. – Т. 43. – №. 4. – С. 34.

12. Boytsov L., Naidan B. Engineering Efficient and Effective Non-metric Space Library //Similarity Search and Applications. – Springer Berlin Heidelberg, 2013. – С. 280-293.

13. Dong W. et al. Modeling lsh for performance tuning //Proceedings of the 17th ACM conference on Information and knowledge management. – ACM, 2008. – С. 669-678.

14. Kullback S., Leibler R. A. On information and sufficiency //The annals of mathematical statistics. – 1951. – С. 79-86.

15. Itakura F., Saito S. Analysis synthesis telephony based on the maximum likelihood method //Proceedings of the 6th International Congress on Acoustics. – pp. C17–C20, 1968. – Т. 17. – С. C17-C20.

16. Uhlmann J. K. Satisfying general proximity/similarity queries with metric trees //Information processing letters. – 1991. – Т. 40. – №. 4. – С. 175-179.

17. Yianilos P. N. Data structures and algorithms for nearest neighbor search in general metric spaces //SODA. – 1993. – Т. 93. – №. 194. – С. 311-321.

18. Chávez E., Navarro G. Probabilistic proximity search: Fighting the curse of dimensionality in metric spaces //Information Processing Letters. – 2003. – Т. 85. – №. 1. – С. 39-46.

19. Gonzalez E. C., Figueroa K., Navarro G. Effective proximity retrieval by ordering permutations //Pattern Analysis and Machine Intelligence, IEEE Transactions on. – 2008. – Т. 30. – №. 9. – С. 1647-1658.

20. Amato G., Savino P. Approximate similarity search in metric spaces using inverted files //Proceedings of the 3rd international conference on Scalable information systems. – ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. – С. 28.

21. Esuli A. Use of permutation prefixes for efficient and scalable approximate similarity search //Information Processing & Management. – 2012. – Т. 48. – №. 5. – С. 889-902.

22. Figueroa K., Frediksson K. Speeding up permutation based indexing with indexing //Proceedings of the 2009 Second International Workshop on Similarity Search and Applications. – IEEE Computer Society, 2009. – С. 107-114.

23. Tellez E. S., Chávez E., Navarro G. Succinct nearest neighbor search //Information Systems. – 2013. – Т. 38. – №. 7. – С. 1019-1030.

24. Samet H. Foundations of Multidimensional and Metric Data Structures Morgan Kaufmann Publishers Inc. – 2005.

25. Indyk P., Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality //Proceedings of the thirtieth annual ACM symposium on Theory of computing. – ACM, 1998. – С. 604-613.

26. Chávez E., Navarro G. A compact space decomposition for effective metric indexing //Pattern Recognition Letters. – 2005. – Т. 26. – №. 9. – С. 1363-1376.

27. Bolettieri P. et al. CoPhIR: a test collection for content-based image retrieval //arXiv preprint arXiv:0905.4627. – 2009.

28. Jegou H., Douze M., Schmid C. Product quantization for nearest neighbor search //Pattern Analysis and Machine Intelligence, IEEE Transactions on. – 2011. – Т.3. – №. 1. – С. 117-128.

29. Lowe D. G. Distinctive image features from scale-invariant keypoints //International journal of computer vision. – 2004. – Т. 60. – №. 2. – С. 91-110.

30. Řehůřek R., Sojka P. Software framework for topic modelling with large corpora. – 2010.

31. Blei D. M., Ng A. Y., Jordan M. I. Latent dirichlet allocation //the Journal of machine Learning research. – 2003. – Т. 3. – С. 993-1022.

32. Ruiz G. et al. Finding Near Neighbors Through Local Search //Similarity Search and Applications. – Springer International Publishing, 2015. – С. 103-109.