НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
январь–февраль 2024 Том 24 № 1 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
January–February 2024 Vol. 24 No 1 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2024-24-1-112-117
УДК 004.021

Метод хранения векторных представлений в сжатом виде с применением кластеризации

Никита Андреевич Томилов[1], Владимир Павлович Туров[2], Александр Амаякович Бабаянц[3]✉, Алексей Владимирович Платонов[4]

[1,2,3,4] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[1] programmer174@icloud.com, https://orcid.org/0000-0001-9325-0356
[2] firemoon@icloud.com, https://orcid.org/0009-0009-1470-7633
[3] babayants.alexander@gmail.com✉, https://orcid.org/0009-0004-6367-6398
[4] avplatonov@itmo.ru, https://orcid.org/0000-0002-8485-1296

Аннотация

Введение. Алгоритмы машинного обучения для информационного поиска позволяют представить текстовые и мультимодальные документы в виде векторов. Такие векторные представления (embeddings) сохраняют семантическое содержание документов и сводят задачу поиска к задаче определения расстояния между векторами. Сжатие векторных представлений позволяет уменьшить объем памяти, занимаемый ими, и повысить эффективность вычислений. В работе рассмотрены существующие способы сжатия векторных представлений без потери и с потерей точности. Предложен метод уменьшения ошибки путем кластеризации векторных представлений при использовании сжатия с потерей точности. **Метод.** Сущность метода состоит в предварительной кластеризации векторных представлений, сохранении центров каждого кластера и значений координат каждого векторного представления относительно центра его кластера. Центры каждого кластера сжимаются без потери точности, а получившиеся смещенные векторные представления с потерей точности. **Основные результаты.** Предложенный метод протестирован на наборах данных fashion-mnist-784-euclidean и NYT-256-angular. Проведено сравнение векторных представлений, сжатых с потерей точности при помощи уменьшения разрядности, с векторными представлениями, сжатыми по предложенному методу. При незначительном, около 10 %, увеличении размера сжатых данных средняя абсолютная величина ошибки от потери точности для наборов fashion-mnist-784-euclidean и NYT-256-angular снизилась в четыре и примерно в два раза соответственно. **Обсуждение.** Разработанный метод может быть применен для решения задач хранения и обработки векторных представлений мультимодальных документов, например, при разработке поисковых систем.

Ключевые слова

векторные представления, эмбеддинг, кластеризация, k-means, сжатие векторных представлений

Ссылка для цитирования: Томилов Н.А., Туров В.П., Бабаянц А.А., Платонов А.В. Метод хранения векторных представлений в сжатом виде с применением кластеризации // Научно-технический вестник информационных технологий, механики и оптики. 2024. Т. 24, № 1. С. 112–117. doi: 10.17586/2226-1494-2024-24-1-112-117

A method of storing vector data in compressed form using clustering

Nikita A. Tomilov[1], Vladimir P. Turov[2], Alexander A. Babayants[3]✉, Alexey V. Platonov[4]

[1,2,3,4] ITMO University, Saint Petersburg, 197101, Russian Federation

[1] programmer174@icloud.com, https://orcid.org/0000-0001-9325-0356
[2] firemoon@icloud.com, https://orcid.org/0009-0009-1470-7633
[3] babayants.alexander@gmail.com✉, https://orcid.org/0009-0004-6367-6398
[4] avplatonov@itmo.ru, https://orcid.org/0000-0002-8485-1296

Abstract

The development of the machine learning algorithms for information search in recent years made it possible to represent text and multimodal documents in the form of vectors. These vector representations (embeddings) preserve the semantic

content of documents and allow the search to be performed as the calculation of distance between vectors. Compressing embeddings can reduce the amount of memory they occupy and improve computational efficiency. The article discusses existing methods for compressing vector representations without loss of accuracy and with loss of accuracy. A method is proposed to reduce error by clustering vector representations using lossy compression. The essence of the method is in performing the preliminary clustering of vector representations, saving the centers of each cluster, and saving the coordinate value of each vector representation relative to the center of its cluster. Then, the centers of each cluster are compressed without loss of accuracy, and the resulting shifted vector representations are compressed with loss of accuracy. To restore the original vector representations, the coordinates of the center of the corresponding cluster are added to the coordinates of the displaced representation. The proposed method was tested on the fashion-mnist-784-euclidean and NYT-256-angular datasets. A comparison has been made of compressed vector representations with loss of accuracy by reducing the bit depth with vector representations compressed using the proposed method. With a slight (around 10 %) increase in the size of the compressed data, the absolute value of the error from loss of accuracy decreased by four and two times, respectively, for the tested sets. The developed method can be applied in tasks where it is necessary to store and process vector representations of multimodal documents, for example, in the development of search engines.

Keywords

vector representations, embeddings, clustering, k-means, compression of vector representations

For citation: Tomilov N.A., Turov V.P., Babayants A.A., Platonov A.V. A method of storing vector data in compressed form using clustering. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2024, vol. 24, no. 1, pp. 112–117 (in Russian). doi: 10.17586/2226-1494-2024-24-1-112-117

Введение

В современном мире стремительно растет количество генерируемых данных, как в текстовом формате (статьи, публикации), так и в прочих форматах (изображения, звук). Соответственно, возникает необходимость быстрого поиска информации по этим данным. В последнее время наибольшую популярность получают алгоритмы поиска с применением машинного обучения, способные формировать векторные представления, позволяющие отображать семантическую составляющую текстовых [1] или мультимодальных документов в виде векторов (эмбеддингов). Далее процесс информационного поиска сводится к проведению операций над полученными векторами [2]. Для хранения векторных представлений в неизмененном виде часто требуется большой объем дискового пространства и памяти. С целью уменьшения размера векторных представлений используются различные алгоритмы сжатия и кодирования чисел. Наиболее популярным из таких алгоритмов является квантизация — пересчет каждого из значений в представление с меньшей точностью. Например, из FP32 (число с плавающей точкой 32-битной разрядности) в представление FP16 или FP8 (число с плавающей точкой 16-битной или 8-битной разрядности), и дальнейшее хранение и использование FP16 или FP8 [3]. Такой алгоритм является достаточно эффективным и позволяет кратно уменьшить размер векторного представления. Недостаток алгоритма — очевидная потеря точности, которая может влиять при дальнейшем использовании данных эмбеддингов, например, для их использования в моделях машинного обучения [4].

В настоящей работе предложен алгоритм сжатия векторных представлений, при котором вместо хранения исходных векторных представлений производится их кластеризация и вычисление центроидов кластеров. В результате каждое векторное представление может быть сохранено как пара из индекса кластера и компоненты разницы до центроида кластера (дельты), и затем записано на диск в сжатом виде. Данный метод позволяет сохранять центроиды и дельты с разной потерей точности, например: сжатие центроидов без потерь и сжатие дельт с пересчетом из FP32 в FP16 или FP8. Выдвигаемая гипотеза заключается в том, что это позволит снизить потерю точности относительно преобразования исходных векторных представлений из FP32 в FP16 или FP8.

Описание алгоритма

В работе предложен метод хранения и сжатия векторных представлений с дополнительной информацией. Каждое векторное представление, являющееся массивом чисел с плавающей точкой 32-битной разрядности, характеризуется, помимо непосредственно значений массива чисел, первичным и вторичным ключами. Такой подход расширяет и уточняет индексацию данных и позволяет, например, при хранении векторных представлений текстовых документов, указывать в качестве первичного ключа — идентификатор текстового документа, а в качестве вторичного ключа — идентификатор отрывка, например, номер предложения, при условии, что необходимо хранить и использовать именно эмбеддинги предложений [5].

Для хранения формируются два файла, которые далее помещаются в хранилище. Первый файл хранит непосредственно векторное представление в двоичном формате (N — количество кортежей, с одинаковым первичным ключом; кортежи, список длиной N). Каждый кортеж хранит информацию отдельного векторного представления, т. е. (вторичный ключ; длину массива; массив байт). Второй файл реализует индекс и хранит соответствие «первичный ключ — смещение в первом файле, начиная с которого находятся все векторные представления, относящиеся к этому первичному ключу».

Перед сохранением в файл данные каждого векторного представления кодируются и сжимаются. Рассмотрим три вида кодирования.

— FP32 (float32) — кодирование без потерь точности. Каждый компонент векторного представления занимает 4 Б.

— FP16 (float16) — кодирование с потерей точности. Каждый компонент векторного представления преобразуется в формат с плавающей запятой 16-битной разрядности. Занимает в два раза меньше памяти.
— FP8 (float8/minifloat) — кодирование с потерей точности. Каждый компонент векторного представления преобразуется в формат с плавающей запятой 8-битной разрядности. Занимает в четыре раза меньше памяти.

После кодирования каждый из массивов байт сжимается одним из трех алгоритмов и форматов сжатия — ZLib[1], LZMA, XZ [6]. Все алгоритмы сжатия сконфигурированы такими параметрами, при которых предполагается достижение наибольшего коэффициента сжатия. Все целочисленные значения, а именно, первичные и вторичные ключи и смещения кодируются с применением кодирования переменной длины, что позволяет, несмотря на использование 4- и 8-байтных типов данных, использовать только необходимое число байт (например, числа до 127 кодируются одним байтом).

Такой способ хранения предоставляет возможность быстрого доступа к любому векторному представлению без необходимости распаковывать избыточные данные. Данная организация хранения позволяет создать потокобезопасную и быструю реализацию алгоритма перебора (аналогичную операции «full table scan» [7] для реляционных систем управления базами данных), необходимую для выполнения векторного поиска по всем векторным представлениям или для их экспорта в другие хранилища. Помимо этого, доступен выбор любой комбинации алгоритмов сжатия и кодирования векторных представлений.

Для проверки гипотезы о применимости кластеризации и хранении векторных представлений как дельт относительно центроидов кластеров, рассматриваемый алгоритм был дополнен следующими условиями. Все векторные представления с одинаковым первичным ключом кластеризуются с помощью алгоритма кластеризации k-means [8] до указанного (параметром N) числа кластеров, и каждому векторному представлению ставится в соответствие индекс кластера, к которому он принадлежит.

На основе всех доступных данных создано два хранилища. В первое хранилище помещены центроиды кластеров, у которых первичным ключом является собственный индекс кластера, а вторичным ключом — ноль. Во второе хранилище записаны дельты — расстояние между векторным представлением и центроидом кластера, к которому он относится, с сохранением первичного и вторичного ключей. Связь между хранилищами обеспечивает дополнительный служебный файл, который хранит кортежи (первичный и вторичный ключи, индекс кластера). Для извлечения оригинального векторного представления из описанной структуры необходимо произвести следующие операции.

— Из хранилища дельт извлечь дельту, соответствующую первичному и вторичному ключам.
— Из файла соответствий получить индекс кластера, к которому относится необходимое векторное представление.
— Из хранилища центроидов извлечь необходимый центроид кластера, к которому относится требуемое векторное представление.
— Рассчитать сумму центроида кластера и дельты и получить искомое векторное представление.

Такой подход позволяет выполнить сохранение центроидов и дельт кластеров с разными настройками сжатия. В результате возможно увеличить точность сохранения векторных представлений при сжатии с потерями при одинаковых настройках сжатия исходных векторных представлений и их дельт ценой небольшого дискового пространства, необходимого для хранения самих центроидов кластеров.

Описание эксперимента

Для проверки предложенного алгоритма было выбрано два тестовых набора данных, fashion-mnist-784-euclidean [9] и NYT-256-angular[2]. Данный выбор наборов данных обусловлен их общедоступностью, частотой использования в тестировании алгоритмов машинного обучения, а также тем, что наборы сильно отличаются друг от друга по семантике закодированных данных, диапазону значений и метрике расстояния. Это условия позволяют оценить работу алгоритма на разных входных данных.

Оба набора данных были преобразованы в форматы, соответствующие описанным хранилищам в разделе «Описание алгоритма». Для каждого полученного хранилища выполним замеры занимаемого места на диске и метрики ошибки. Метрика ошибки — расстояние между векторным представлением, сохраненным и затем извлеченным из хранилища, и исходным. При этом расстояние может быть евклидовым или угловым и выбирается в зависимости от используемого набора данных. Обозначив векторное представление из хранилища, заданное набором компонентов $(vc_0, \ldots, vc_n) - vc$; а исходное векторное представление $(vs_0, \ldots, vs_n) - vs$, получим следующие формулы для расчета метрик ошибок при использовании:

— евклидова расстояния

$$e = \sqrt{\sum_{k=0}^{n} (vc_k - vs_k)^2};$$

— углового расстояния

$$e = \frac{\sum_{k=0}^{n} vc_k vs_k}{\sqrt{\sum_{k=0}^{n} vc_n^2} \sqrt{\sum_{k=0}^{n} vs_n^2}}.$$

[1] Gailly J., Adler M. Zlib compression library [Электронный ресурс]. Режим доступа: https://www.repository.cam.ac.uk/items/248f35af-3d9c-40a4-8a70-52520d274a47, свободный. Яз. англ. (дата обращения: 28.05.2023).

[2] Dua D. and Graff C. UCI Machine Learning Repository. 2019. [Электронный ресурс]. Режим доступа: https://archive.ics.uci.edu/ml/datasets/bag+of+words, свободный. Яз. англ. (дата обращения: 30.05.2023).

Вычислим метрики для каждого из векторных представлений и запишем их минимальное, максимальное и медианное значение. Чем меньше полученное значение ошибки, тем ближе векторные представления друг к другу и тем меньше потеря точности при сохранении в хранилище.

Для сравнения с эталонным примером хранения данных, векторные представления из обоих наборов преобразуем в базы данных SQLite3 [10] с сохранением в таблицу в виде массивов байт после применения алгоритмов сжатия ZLib, LZMA, XZ. Отметим, что для замера места на диске был включен оригинальный файл набора данных в формате HDF5, который может дать «порядок» размера оригинальных данных несмотря на то что, кроме использованных векторных представлений (взятых из «обучающей» выборки наборов данных), содержит еще и «проверочную» выборку наборов данных.

Тестовый стенд имеет следующие характеристики: AMD Ryzen 7 7700X 8C16T; 64GB RAM; NVMe WD SN850X 2TB; операционная система Ubuntu 22.04; OpenJDK 17; разработанный в рамках предыдущих исследований инструмент сравнения алгоритмов векторного поиска [11].

Заметим, что измерений времени извлечения единичного векторного представления и времени выполнения полного перебора не производилось. Это связано с тем, что использованные реализации алгоритмов сжатия LZMA и XZ из пакета Apache Commons Compress задействовали промежуточные преобразования массивов байт, ведущие к большим накладным расходам.

Результаты эксперимента

Первый набор данных, fashion-mnist-784-euclidean, содержит 60 000 векторных представлений изображений различных предметов одежды. Каждое векторное представление состоит из 784 компонентов [9]. В силу общедоступности, простоты анализа и не очень большого размера, этот набор данных получил широкое распространение как один из эталонов для тестирования алгоритмов машинного обучения [12]. Для работы с первым набором данных использована метрика евклидова расстояния. Каждый компонент векторного представления кодирует яркость соответствующего пиксела числом в диапазоне от 0 до 255. Данный набор векторных представлений содержит неплотные (разреженные) вектора, содержащие большое количество нулевых компонентов.

Размеры файлов хранилищ представлены в табл. 1. При использовании формата FP32 ошибки нет, так как исходные данные были представлены в этом же формате — потери точности не происходит. При использовании формата FP8 наблюдается очень большая потеря точности. При этом, если сравнивать кодирование в FP8 без кластеризации и с кластеризацией, то можно отметить, что хранение центроидов кластеров значительно снижает ошибку, хотя и не устраняет ее полностью. Однако, в связи с тем, что набор данных содержит значения в ограниченном диапазоне от 0 до 255, представляется целесообразным хранить их как 8-битные целые числа. При условии, что все значения векторных представлений оригинального набора данных фактически целочисленные (хотя и хранятся как FP32), при преобразовании в INT8 и сжатия с помощью кодирования переменной длины [13] потерь точности не происходит, и метрика потери точности для всех полученных хранилищ равна нулю. Тем не менее, эти результаты показывают, насколько метод кодирования с кластеризацией может улучшать точность при использовании форматов с малым диапазоном значений (FP8), а также что дельты сжимаются хуже, чем оригинальные векторные представления той же размерности. Также из этих результатов можно сделать вывод о том, что применение предложенного алгоритма с хранением

Таблица 1. Размер на диске и ошибка (по метрике евклидова расстояния) для набора данных fashion-mnist-784-euclidean, сохраненного с использованием разных алгоритмов сжатия

Table 1. Disk size (in megabytes) and error (as euclidean distance) for the fashion-mnist-784-euclidean dataset, saved with different compression algorithms

Хранилище	Тип данных	Размер на диске при алгоритме сжатия, МБ				Метрика ошибки (евклидово расстояние)		
		без сжатия	ZLib	XZ	LZMA	среднее	медиана	максимум
Оригинальный файл HDF5	FP32	227,5	—	—	—	нет		
База данных SQLite3	FP32	235,7	46,1	40,6	37,5	нет		
Бинарное без кластеризации	FP32	180,1	40,8	36,2	33,6	нет		
	FP16	89,7	33,6	40,6	38,0	0,001	0,001	0,013
	FP8	45,4	25,5	27,7	24,6	4750,7	4568,5	16 362,0
Бинарное, 1000 кластеров	FP32	183,0	111,2	110,0	107,3	нет		
	FP16	93,2	67,5	70,2	67,5	0,045	0,039	0,327
	FP8	51,0	24,6	24,0	25,1	656,9	569,8	4318,4
Бинарное, 2000 кластеров	FP32	186,2	108,6	106,4	103,7	нет		
	FP16	96,3	67,5	70,6	67,9	0,037	0,031	0,250
	FP8	50,9	33,1	34,1	31,0	598,5	509,0	4293,3

Таблица 2. Размер на диске и ошибка (по метрике углового расстояния) для набора данных NYT-256-angular, сохраненного с использованием разных алгоритмов сжатия

Table 2. Disk size (in megabytes) and error (as angular distance) for the NYT-256-angular dataset, saved with different compression algorithms

Хранилище	Тип данных	Размер на диске при алгоритме сжатия, МБ				Метрика ошибки (угловое расстояние)		
		без сжатия	ZLib	XZ	LZMA	среднее	медиана	максимум
Оригинальный файл HDF5	FP32	300,6	—	—	—	нет		
База данных SQLite3	FP32	382,6	288,3	382,3	382,3	нет		
Бинарное без кластеризации	FP32	286,1	280,2	300,1	287,5	нет		
	FP16	144,1	145,9	161,1	150,4	0,0001	0,0001	0,0001
	FP8	73,0	53,9	80,7	68,0	0,0011	0,0007	0,0221
Бинарное, 1000 кластеров	FP32	289,4	282,8	302,9	290,2	нет		
	FP16	146,7	148,7	163,9	153,2	0,0001	0,0001	0,0001
	FP8	77,4	71,0	77,0	72,1	0,0007	0,0004	0,0199
Бинарное, 2000 кластеров	FP32	290,2	282,9	303,0	290,3	нет		
	FP16	147,9	149,2	164,6	153,8	0,0001	0,0001	0,0001
	FP8	77,4	72,0	73,4	71,1	0,0007	0,0004	0,0199

векторов в двоичных файлах занимает на несколько мегабайт меньше памяти, чем хранение векторов в SQLite при использовании одинаковых алгоритмов сжатия.

Второй набор данных, NYT-256-angular [10], содержит 290 тысяч векторных представлений текстов статей газеты New York Times. Каждое векторное представление состоит из 256 компонентов, для работы с этим набором данных использована метрика углового (angular) расстояния. Второй набор данных был выбран для тестирования, так как он сильно отличается по содержанию и диапазону значений от набора fashion-mnist-784-euclidean, а также использует другую метрику расстояния. В значениях компонентов каждого из векторных представлений практически нет нулей, а все значения лежат в диапазоне от [–1; 1].

Объемы файлов хранилищ, а также вычисленные расстояния (показатель ошибки) представлены в табл. 2. Аналогично первому набору данных, при использовании формата FP32 ошибки нет, исходные данные были представлены в этом же формате и потери точности не происходит. Для второго набора данных практически отсутствует разница в эффективности сжатия дельт по сравнению с эффективностью сжатия исходных векторных представлений, что не наблюдалось для набора данных fashion-mnist-784-euclidean с повторяющимися нулевыми компонентами в векторных представлениях. При сжатии через кластеризацию для одинаковых настроек хранилище, использующее кластеризацию, занимает на фиксированное число мегабайт больше дискового пространства, чем хранилище без кластеризации, а разница обусловлена именно хранением центроидов кластеров; однако эта разница небольшая и в худшем случае центроиды занимают около 4 МБ. Из данных результатов можно сделать вывод что использование кластеризации для тысячи кластеров (и хранении центроидов кластеров в FP32 без потерь сжатия) позволяет уменьшить значения метрики ошибки, что подтверждает заявленную гипотезу.

Заключение

При хранении векторных представлений применение универсальных алгоритмов сжатия без потерь, а именно ZLIB, LZMA и XZ, оправданно и позволяет существенно снизить объем занимаемого дискового пространства для хранения. При работе с разреженными векторными представлениями предложенный и описанный алгоритм хранения векторных представлений с использованием кодирования FP8 позволяет на порядок уменьшить занимаемое дисковое пространство для хранения. Также происходит уменьшение пространства при сравнении хранения векторных представлений в SQLite и с применением универсальных алгоритмов сжатия без потерь. В частности, для набора данных fashion-mnist-784-euclidean векторные представления, закодированные FP8 и сжатые с помощью ZLIB, используют 25,5 МБ дискового пространства, что почти в два раза меньше, чем сжатие оригинальных векторных представлений с помощью ZLIB (40,8 МБ), в 60 раз меньше, чем использование баз данных SQLite3 без сжатия (236 МБ) и на несколько мегабайт меньше, чем — SQLite3 со сжатием. При работе с плотными векторными представлениями подход с использованием кластеризации позволил увеличить точность сохранения векторных представлений при использовании сжатия с потерями, ценой незначительного увеличения дискового пространства за счет дополнительного хранения центроидов кластеров. На протестированных наборах NYT-256-angular и fashion-mnist-784-euclidean при использовании дополнительного хранения центроидов кластеров метрика ошибки уменьшилась в два и четыре раза соответственно.

Литература

1. Grbovic M., Cheng H. Real-time personalization using embeddings for search ranking at airbnb // Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018. P. 311–320. https://doi.org/10.1145/3219819.3219885
2. Berry M.W., Drmac Z., Jessup E.R. Matrices, vector spaces, and information retrieval // SIAM Review. 1999. V. 41. N 2. P. 335–362. https://doi.org/10.1137/S0036144598347035
3. Micikevicius P., Narang S., Alben J., Diamos G., Elsen E., Garcia D., Ginsburg B., Houston M., Kuchaiev O., Venkatesh G., Wu H. Mixed precision training // arXiv. 2017. arXiv:1710.03740. https://doi.org/10.48550/arXiv.1710.03740
4. Zhang J., Yang J., Yuen H. Training with low-precision embedding tables // Systems for Machine Learning Workshop at NeurIPS. 2018. V. 2018.
5. Moghadasi M.N., Zhuang Y. Sent2Vec: A new sentence embedding representation with sentimental semantic // Proc. of the 2020 IEEE International Conference on Big Data (Big Data). 2020. P. 4672–4680. https://doi.org/10.1109/BigData50022.2020.9378337
6. Parekar P.M., Thakare S.S. Lossless data compression algorithm–a review // International Journal of Computer Science & Information Technologies. 2014. V. 5. N 1. P. 276–278.
7. Manro A., Sinha S., Chaturvedi B., Mohan J. Index seek versus table scan performance and implementation of RDBMS // Lecture Notes in Electrical Engineering. 2019. V. 526. P. 411–420. https://doi.org/10.1007/978-981-13-2553-3_40
8. Kanungo T., Mount D., Netanyahu N., Piatko Ch., Silverman R., Wu A. The analysis of a simple k-means clustering algorithm // Proc. of the sixteenth annual symposium on Computational geometry. 2000. P. 100–109. https://doi.org/10.1145/336154.336189
9. Xiao H., Rasul K., Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms // arXiv. 2017. arXiv:1708.07747. https://doi.org/10.48550/arXiv.1708.07747
10. Bi C., China P.R. Research and application of SQLite embedded database technology // WSEAS Transactions on Computers. 2009. V. 8. N 1. P. 83–92.
11. Туров В.П., Томилов Н.А., Бабаянц А.А. Разработка инструмента сравнения алгоритмов векторного поиска // XI Конгресс молодых учёных: сборник научных трудов. Т. 1. 2022. С. 446–450.
12. Aumüller M., Bernhardsson E., Faithfull A. ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms // Lecture Notes in Computer Science. 2017. V. 10609. P. 34–49. https://doi.org/10.1007/978-3-319-68474-1_3
13. Golomb S. Run-length encodings (Corresp.) // IEEE Transactions on Information Theory. 1966. V. 12. N 3. P. 399–401. https://doi.org/10.1109/TIT.1966.1053907

References

1. Grbovic M., Cheng H. Real-time personalization using embeddings for search ranking at airbnb. *Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 311–320. https://doi.org/10.1145/3219819.3219885
2. Berry M.W., Drmac Z., Jessup E.R. Matrices, vector spaces, and information retrieval. *SIAM Review*, 1999, vol. 41, no. 2, pp. 335–362. https://doi.org/10.1137/S0036144598347035
3. Micikevicius P., Narang S., Alben J., Diamos G., Elsen E., Garcia D., Ginsburg B., Houston M., Kuchaiev O., Venkatesh G., Wu H. Mixed precision training. *arXiv*, 2017, arXiv:1710.03740. https://doi.org/10.48550/arXiv.1710.03740
4. Zhang J., Yang J., Yuen H. Training with low-precision embedding tables. *Systems for Machine Learning Workshop at NeurIPS*, 2018, vol. 2018.
5. Moghadasi M.N., Zhuang Y. Sent2Vec: A new sentence embedding representation with sentimental semantic. *Proc. of the 2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 4672–4680. https://doi.org/10.1109/BigData50022.2020.9378337
6. Parekar P.M., Thakare S.S. Lossless data compression algorithm–a review. *International Journal of Computer Science & Information Technologies*, 2014, vol. 5, no. 1, pp. 276–278.
7. Manro A., Sinha S., Chaturvedi B., Mohan J. Index seek versus table scan performance and implementation of RDBMS. *Lecture Notes in Electrical Engineering*, 2019, vol. 526, pp. 411–420. https://doi.org/10.1007/978-981-13-2553-3_40
8. Kanungo T., Mount D., Netanyahu N., Piatko Ch., Silverman R., Wu A. The analysis of a simple k-means clustering algorithm. *Proc. of the sixteenth annual symposium on Computational geometry*, 2000, pp. 100–109. https://doi.org/10.1145/336154.336189
9. Xiao H., Rasul K., Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv*, 2017, arXiv:1708.07747. https://doi.org/10.48550/arXiv.1708.07747
10. Bi C., China P.R. Research and application of SQLite embedded database technology. *WSEAS Transactions on Computers*, 2009, vol. 8, no. 1, pp. 83–92.
11. Turov V.P., Tomilov N.A., Babaiantc A.A. Development of a tool for comparing vector search algorithms. *Proc. of the XI Congress of Young Scientists. V. 1*. 2022, pp. 446–450. (in Russian)
12. Aumüller M., Bernhardsson E., Faithfull A. ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Lecture Notes in Computer Science*, 2017, vol. 10609, pp. 34–49. https://doi.org/10.1007/978-3-319-68474-1_3
13. Golomb S. Run-length encodings (Corresp.). *IEEE Transactions on Information Theory*, 1966, vol. 12, no. 3, pp. 399–401. https://doi.org/10.1109/TIT.1966.1053907

Авторы

Томилов Никита Андреевич — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 57225127284, https://orcid.org/0000-0001-9325-0356, programmer174@icloud.com

Туров Владимир Павлович — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0009-1470-7633, firemoon@icloud.com

Бабаянц Александр Амаякович — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0004-6367-6398, babayants.alexander@gmail.com

Платонов Алексей Владимирович — кандидат технических наук, доцент, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0000-0002-8485-1296, avplatonov@itmo.ru

Authors

Nikita A. Tomilov — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 57225127284, https://orcid.org/0000-0001-9325-0356, programmer174@icloud.com

Vladimir P. Turov — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0009-1470-7633, firemoon@icloud.com

Alexander A. Babayants — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0004-6367-6398, babayants.alexander@gmail.com

Alexey V. Platonov — PhD, Associate Professor, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0000-0002-8485-1296, avplatonov@itmo.ru

Статья поступила в редакцию 08.06.2023
Одобрена после рецензирования 10.01.2024
Принята к печати 29.01.2024

Received 08.06.2023
Approved after reviewing 10.01.2024
Accepted 29.01.2024