［研究助成（A）］

Advanced Vector DB
―― データ構造ドリブンな LLM 知識注入 ――

2241027

研究代表者　東京大学
大学院情報理工学系研究科　講　師　松　井　勇　佑

［研 究 の 目 的］

ChatGPT といった Large Language Model（LLM）は社会を変革する力を持つが，「自分の持つ情報を LLM に読み込ませる」という極めて単純かつ重要な操作について，どうするのが一番よいかという明確な答えがまだ無い。例えば小説を一冊分 LLM に読み込ませ，LLM に「太郎の友達は誰？」と質問し返答させたいとする。このとき，小説の情報を LLM に読み取らせる最も簡単かつ有力な方式はベクトル探索（Vector Database; Vector DB）を元にしたものと言われている。ところが，この Vector DB による探索は現在「色々な人が色々なことをやってみている」という状況にある。私はこの Vector DB に対しデータ構造の深化という方向で取り組んだ。

Vector DB による LLM 知識注入は次のステップからなる：① 自分が持つ文章情報を細切れにし，それらを文章特徴量抽出器に入力し，特徴量ベクトルたちを得る。② LLM への質問文も同様に特徴量ベクトルに変換する。③ 質問に似ている特徴量を探す。④ その結果を質問文に追記し，質問する。ここにおける技術的課題は，上記の既存探索アプローチは極めてナイーブな「似てるもの探し」であり，少し複雑な状況になると途端に対応できないという点にある。例えば小説の例で「太郎にはたくさん友達がいるが，特に花子と親しく，花子に会ったシーンがたくさんある」とする。この時，「太郎の友達は誰？」という質問に対し，上記既存探索アプローチは花子シーンばかりを探してきてしまい，「太郎の友達は花子です」としか答えられないかもしれない。

このようなナイーブな似ているもの探しを脱却するために，申請者は探索部分のデータ構造・アルゴリズムレベルで，複雑な探索をサポートする方式を提案する。上記の例でいうと，「検索結果がうまく散らばる（花子ばかりではなく，花子・次郎・ケンジとする）」ように探索データ構造レベルで対応した。

［研究の内容，成果］

以下に，本計画の主たる成果である「多様な探索」の方式を記す。本方式はコンピュータビジョン分野に関する国内最大の会議「画像の理解・認識シンポジウム（MIRU）」にて，査読を経てオーラル発表に選定された。加えて，オーラル発表の各セッション中，投票で一位のものに与えられる「オーディエンス賞」を受賞した。

■はじめに

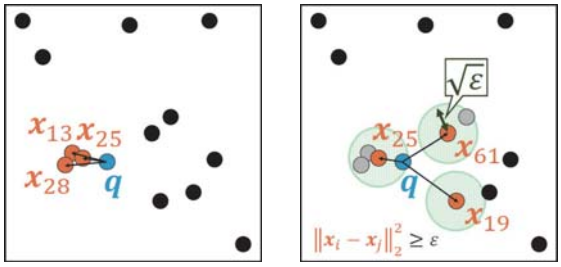
近似最近傍探索とは，クエリに近いベクトルをベクトル集合から高速に探す処理である[23]。近似最近傍探索は様々なシステムの構成要素であり，画像検索や情報推薦など広い分野で用いられている。近年は，大規模言語モデルに情報を入力する Retrieval Augmented Generation

(RAG)方式のために近似最近傍探索は注目されている[3]。

近似最近傍探索は有用だが，探索結果が必ずしも多様ではない。例えば猫画像をクエリとすると，その猫に似た画像が多数存在する場合，探索結果上位に同じ猫が並んでしまい，均一な結果となる。このような状況では，ただクエリに似ているだけでなく，探索結果がお互いに似て「いない」，すなわち多様な結果が好まれる場面がある。そのような探索問題は多様近傍探索[10,32,38]と呼ばれるが，多くの既存の多様近傍探索方式は以下の理由により遅い。(1) 最新の近似最近傍探索方式の知見が取り込まれていない，(2) 組み合わせを計算する必要がある，(3) 元のベクトルそのものにアクセスする必要がある。

我々は，超高速な多様近傍探索を実現する。提案方式は，事前に準備した足切り表を用い，近似最近傍探索結果に対し貪欲に候補を絞り込み，多様な探索を実現する。提案するデータ構造および探索方式はシンプルであり，高速である。その概念図を図1に示す。我々の貢献は以下である。

- 提案方式は近似最近傍探索結果に対する「後処理モジュール」である。そのため，最新の超高速な近似最近傍探索方式をブラックボックスとして利用出来る。
- パラメータである閾値 ε を事前に学習する方式を提案する。提案方式は煩雑なパラメータ調整が無い。
- 提案方式は大規模データ（106本の96次元ベクトル）にも有効であり，0.05[ms/query]しかかからない。



(a) 通常の近似最近傍探索　　(b) 提案する多様近傍探索

図1　提案方式の概要図

■事前準備

N 本の D 次元ベクトル $\{x_n\}_{n=1}^N$ があるとする。クエリ $\boldsymbol{q}\in\mathbb{R}^D$ が与えられた際，$\boldsymbol{q}$ に近くかつ多様な K 本のベクトルを得たい。得られた結果は，IDの集合として $\mathcal{K}\subseteq\{1,\ldots,N\}$ かつ $|\mathcal{K}|=K$ で表す。

探索は2ステップから成る：(1) クエリに対して近似最近傍探索を実行し，クエリに近い S 本のベクトルを得る。これを初期候補集合 $\mathcal{S}$ とする。2つ目のステップは，候補集合 $\mathcal{S}$ から，部分集合 $\mathcal{K}$ を選び取ることである。これは評価関数 $f:2^{\mathcal{S}}\to\mathbb{R}$ を最小化する，部分選択問題として定式化される：$\underset{\mathcal{K}\subseteq\mathcal{S},|\mathcal{K}|=K}{\operatorname{argmin}} f(\mathcal{K})$ ここで f はID集合が「クエリへの近さ」と「多様性」の両方を考慮してどれほど良いかを示す指標である。これを，我々は次のように立式する。

$$f(K)=\frac{1-\lambda}{K}\sum_{k\in\mathcal{K}}\|\boldsymbol{q}-\boldsymbol{x}_k\|_2^2-\lambda\min_{i,j\in\mathcal{K},i\neq j}\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2^2$$ ここで第一項は選択されたベクトルたちとクエリがどれほど近いかを示す，最近傍探索そのものである。第二項はベクトルたちの多様度を示す指標である。ここでは[1]，[16]に習い，ベクトル同士の中で最も近い距離とした。ここで λ は2つの項を調整するパラメータであり，0であれば最近傍探索に，1であればクエリを考慮せず集合の多様度を評価するMAX-MIN多様化問題[31]となる。

■提案方式

提案方式は各点から距離が近いものを事前に調べIDを表に保存し，探索時にIDを表引きして候補から除外する。これをOrderedSetというデータ構造で高速に実現する。

前処理

前処理をアルゴリズム1に示す。データと，二乗距離の閾値 ε を入力とする。L2において，まず近似最近傍探索のデータ構造（インデク

ス）I を構成する。I には任意のもの（例えば HNSW[22]）を使用して良い。次に L2-3 で足切り表を構成する。各点について，自身との二乗距離が ε 未満のベクトルの ID の集合 $\mathcal{L}_n$ を記録する。これを足切り表（整数配列の配列）と呼ぶ。足切り表を計算するには各点をクエリに範囲探索を実行する。この計算量を合計 $O(NT)$ とする。この処理は最近傍探索ではなく範囲探索だが，簡単のため近似最近傍探索と同じ $O(T)$ とした。後に表 2 で示す通り $N=10^6$ の場合このステップの実行時間は最大 10 秒程度である。

足切り表の要素数の平均値を L とすると，足切り表に必要な素朴なメモリ量は，要素に 64 bit 整数を用いて 64 LN[bit] となる（表 2 より，L は 100 以下程度）。

Algorithm 1: BUILD

Input: $\{\mathbf{x}_n\}_{n=1}^N \subseteq \mathbb{R}^D$, $\varepsilon \in \mathbb{R}$

1 $\mathcal{I} \leftarrow$ BUILDINDEX($\{\mathbf{x}_n\}_{n=1}^N$) # Index of ANN

2 for $n \in \{1,\dots,N\}$ do

3 $\quad \mathcal{L}_n \leftarrow \{i \in \{1,\dots,N\} \mid \|\mathbf{x}_n - \mathbf{x}_i\|_2^2 < \varepsilon, n \neq i\}$

4 return $\mathcal{I}, \{\mathcal{L}_n\}_{n=1}^N$

探索のアルゴリズム

アルゴリズム 2 と図 2 で探索について述べる。入力は，クエリ，初期探索の結果個数，最終的な結果個数，インデクス，足切り表になる。

まず L1（図 2a）にて近似最近傍探索を実行し候補集合を得る。次に L2 にて最終結果のための整数集合を準備する。L3-6 では，要素数が K になるまで集合 $\mathcal{K}$ に要素を追加する。L4-5 では，候補集合から，最もクエリに近い（配列の先頭にいる）ID を k として取り出し，くわえる。L6 が収容である。現在注目している k に関して，$\boldsymbol{x}_k$ に近いベクトルたちの ID は $\mathcal{L}_k$ に保存されている。よって，これを候補集合から除去することで，候補集合中から似たものを消すことができる（図 2b）。このステップを繰り返すことにより，最終的に得られる $\mathcal{K}$ の要素はお互いに少なくとも $\sqrt{\varepsilon}$ は離れているものとなる。

Algorithm 2: SEARCH

Input: $\mathbf{q} \in \mathbb{R}^D$, $S(\leq N)$, $K(\leq S)$, $\mathcal{I}$, $\{\mathcal{L}_n\}_{n=1}^N$

1 $\mathcal{S} \leftarrow \mathcal{I}$.SEARCH($\mathbf{q}, S$) # $\mathcal{S} \subseteq \{1,\dots,N\}$

2 $\mathcal{K} \leftarrow \emptyset$

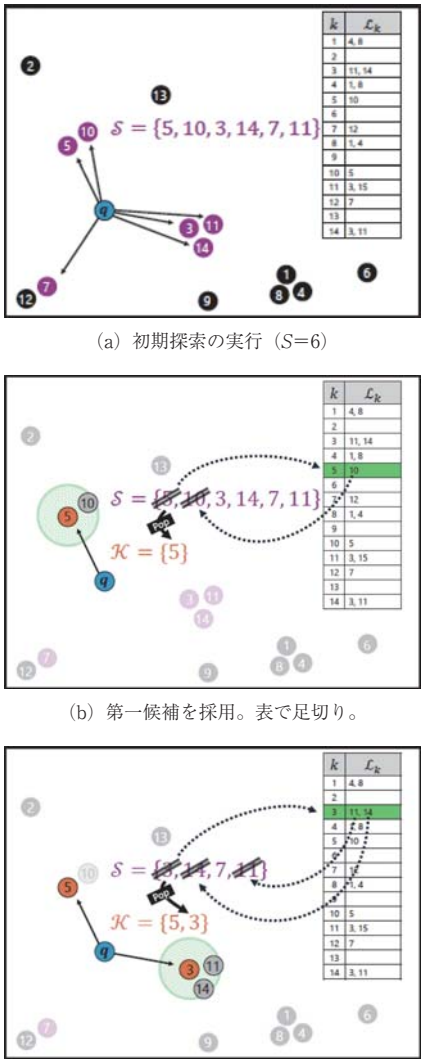
3 while $|\mathcal{K}| < K$ do # At most K times

4 $\quad k \leftarrow$ POP($\mathcal{S}$) # $\mathcal{O}(L)$

5 $\quad \mathcal{K} \leftarrow \mathcal{K} \cup \{k\}$ # $\mathcal{O}(1)$

6 $\quad \mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{L}_k$ # Remove as applicable. $\mathcal{O}(L)$

7 return $\mathcal{K}$ # $\mathcal{K} \subseteq \mathcal{S}$ where $|\mathcal{K}| = K$



（a）初期探索の実行（S=6）

（b）第一候補を採用。表で足切り。

（c）次の候補を採用。表で足切り。

図 2　提案方式の流れ

■実験

性能を評価する。全実験は AWS EC2 インスタンス（3.2 GHz Intel Xeon CPU，仮想 32 コア，64 GiB メモリ）で実行した。前処理はマルチスレッドで，探索部分はシングルスレッドで実行した。近似最近傍探索部分は faiss ライブラ

リ[8]中の HNSW 方式[22]を用いた。提案方式は，c＋＋ コードを nanobind[17]で python から呼んだ。

Deep1B[4]の一部である Deep1M を探索対象とした。これは，10^6 本の 96 次元ベクトルから成る。104 本のクエリ，および訓練のために訓練セットの先頭 1000 本を用いた。

表 1 にて方式を比較する。比較対象は近似最近傍探索単体の HNSW，および，S に対しお互いが遠くなるように貪欲に結果を更新する GMM[1,31]を選んだ。

・提案手法は HNSW 単体に比べてより多様な結果（小さい多様化項）を実現している。GMM は最も多様な結果を得ているが，その分探索項が犠牲となっている（クエリに似ていない結果となっている）。
・提案方式は高速（0.047［ms/query］）であり，初期探索（0.273［ms/query］）の 1/6 程度しかかからない。そのため，多様化無しの HNSW 単体と同程度の時間しかかからない。一方，GMM は全対を計算するため，提案の 30 倍である 10.1［ms/query］の時間がかかる。

表 1　比較実験の結果

	精度（↓）			実行時間［ms/query］（↓）		
手法	探索項	多様化項	最終性能（f）	初期探索	多様化	合計
HNSW［22］	0.559	−0.162	0.199	0.207	−	0.207
HNSW［22］＋GMM［31］	0.652	−0.479	0.086	0.280	9.87	10.1
HNSW［22］＋足切り表（提案）	0.585	−0.280	0.153	0.273	0.047	0.320

■まとめ

多様な探索を実現するための「後処理モジュール」を提案した。提案方式は足切りを実現するテーブルを用いる。実験により，提案方式は最新の近似最近傍探索と同程度の時間で多様な探索を実現した。

参考文献

［1］ Amagata, D.: Diversity Maximization in the Presence of Outliers, Proc. AAAI（2023）.
［2］ André, F., Kermarrec, A.-M. and Scouarnec, N. L.: Quicker ADC: Unlocking the Hidden Potential of Product Quantization With SIMD, IEEE TPAMI, Vol. 43, No. 5, pp. 1666–1677（2021）.
［3］ Asai, A., Min, S., Zhong, Z. and Chen, D.: ACL2023 Tutorial on Retrieval-based Language Models and Applications.
［4］ Babenko, A. and Lempitsky, V.: Efficient Indexing of Billion-Scale Datasets of Deep Descriptors, Proc. IEEE CVPR（2016）.
［5］ Baranchuk, D., Babenko, A. and Malkov, Y.: Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors, Proc. ECCV（2018）.
［6］ Chen, D., Li, W., Li, Y., Ding, B., Zeng, K., Lian, D. and Zhou, J.: Learned Index with Dynamic ϵ, Proc. ICLR（2023）.
［7］ Ding, J., Nathan, V., Alizadeh, M. and Kraska, T.: Tsunami: A Learned Multi-dimensional Index for Correlated Data and Skewed Workloads, Proc. VLDB（2020）.
［8］ Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L. and Jégou, H.: The Faiss Library, arXiv, Vol. 2401.08281（2024）.
［9］ Douze, M., Sablayrolles, A. and Jégou, H.: Link and code: Fast indexing with graphs and compact regression codes, Proc. IEEE CVPR（2018）.
［10］ Drosou, M. and Pitoura, E.: Search Result Diversification, Proc. SIGMOD（2010）.
［11］ Drosou, M. and Pitoura, E.: DisC Diversity: Result Diversification based on Dissimilarity and Coverage, Proc. VLDB（2012）.
［12］ Ferragina, P., Lillo, F. and Vinciguerra, G.: Why Are Learned Indexes So Effective?, Proc. ICML（2020）.
［13］ Ferragina, P. and Vinciguerra, G.: Learned Data Structures, Springer International Publishing（2020）.
［14］ Ferragina, P. and Vinciguerra, G.: The PGM-Index: a fully dynamic compressed learned index with provable worst-case bounds, Proc. VLDB（2020）.
［15］ Fu, C., Xiang, C., Wang, C. and Cai, D.: Fast Approximate Nearest Neighbor Search with the Navigating Spreading-out Graph, Proc. VLDB（2019）.
［16］ Hirata, K., Amagata, D., Fujita, S. and Hara, T.: Solving Diversity-Aware Maximum Inner Product Search Efficiently and Effectively, Proc. RecSys（2022）.
［17］ Jakob, W.: nanobind: tiny and efficient C＋＋/Python bindings（2022）. https://github.com/wjakob/nanobind.
［18］ Jégou, H., Douze, M. and Schmid, C.: Product

Quantization for Nearest Neighbor Search, IEEE TPAMI, Vol. 33, No. 1, pp. 117–128（2011）.
[19] Kochenderfer, M. J. and Wheeler, T. A.: Algorithms for Optimization, The MIT Press（2019）.
[20] Kraska, T., Beutel, A., Chi, E. H., Dean, J. and Polyzotis, N.: The Case for Learned Index Structures, Proc. SIGMOD（2018）.
[21] Liu, Q., Zheng, L., Shen, Y. and Chen, L.: Stable Learned Bloom Filters for Data Streams, Proc. VLDB（2020）.
[22] Malkov, Y. A. and Yashunin, D. A.: Efficient and Robust Approximate Nearest Neighbor Search using Hierarchical Navigable Small World Graphs, IEEE TPAMI, Vol. 42, No. 4, pp. 824–836（2020）.
[23] Matsui, Y., Aumüller, M. and Xiao, H.: CVPR2023 Tutorial on Neural Search in Action.
[24] Matsui, Y., Imaizumi, Y., Miyamoto, N. and Yoshifuji, N.: ARM 4-bit PQ: SIMD-based Acceleration for Approximate Nearest Neighbor Search on ARM, Proc. IEEE ICASSP（2022）.
[25] Matsui, Y., Yamaguchi, T. and Wang, Z.: CVPR 2020 Tutorial on Image Retrieval in the Wild.
[26] Mitzenmacher, M.: A Model for Learned Bloom Filters, and Optimizing by Sandwiching, Proc. NeurIPS（2018）.
[27] Nathan, V., Ding, J., Alizadeh, M. and Kraska, T.: Learning Multi-dimensional Indexes, Proc. SIGMOD（2020）.
[28] Oguri, Y. and Matsui, Y.: General and Practical Tuning Method for Off-the-Shelf Graph-Based Index: SISAP Indexing Challenge Report by Team UTokyo, Proc. SISAP（2023）.
[29] Ono, N. and Matsui, Y.: Relative NN-Descent: A Fast Index Construction for Graph-Based Approximate Nearest Neighbor Search, Proc. MM（2023）.
[30] Rao, V., Jain, P. and Jawahar, C.: Diverse Yet Efficient Retrieval using Locality Sensitive Hashing, Proc. ICMR（2016）.
[31] Ravi, S. S., Rosenkrantz, D. J. and Tayi, G. K.: Heuristic and Special Case Algorithms for Dispersion Problems, Operations Research, Vol. 542, No. 2, pp. 299–310（1994）.
[32] Santos, R. L. T., Macdonald, C. and Ounis, I.: Search Result Diversification, Foundations and Trends in Information Retrieval, Vol. 9, No. 1, pp. 1–90（2015）.
[33] Sato, A. and Matsui, Y.: Fast Partitioned Learned Bloom Filter, Proc. NeurIPS（2023）.
[34] Subramanya, S. J., Devvrit, F., Simhadri, H. V., Krishnawamy, R. and Kadekodi, R.: DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node, Proc. NeurIPS（2019）.
[35] Vaidya, K., Knorr, E., Mitzenmacher, M. and Kraska, T.: Partitioned Learned Bloom Filters, Proc. ICLR（2021）.
[36] Wang, M., Xu, X., Yue, Q. and Wang, Y.: A Comprehensive Survey and Experimental Comparison of Graph- Based Approximate Nearest Neighbor Search, Proc. VLDB（2021）.
[37] Wu, J., Zhang, Y., Chen, S., Wang, J., Chen, Y. and Xing, C.: Updatable Learned Index with Precise Positions, Proc. VLDB（2021）.
[38] Zheng, K., Wang, H., Qi, Z., Li, J. and Gao, H.: A Survey of Query Result Diversification, Knowledge and Information Systems, Vol. 51, pp. 1–36（2017）.

［成果の発表，論文など］

（1） 松井勇佑，“学習型足切り表による高速な多様近傍探索”，画像の認識・理解シンポジウム，（MIRU），オーディエンス賞，Oral，2024.