Approximate Nearest Neighbor Search on Standard Search Engines

Fabio Carrara[✉1][0000−0001−5014−5089], Lucia Vadicamo[1][0000−0001−7182−7038], Claudio Gennaro[1][0000−0002−3715−149X], and Giuseppe Amato[1][0000−0003−0171−4315]

ISTI CNR, Pisa, Italy
{fabio.carrara,lucia.vadicamo,claudio.gennaro, giuseppe.amato}@isti.cnr.it

Abstract. Approximate search for high-dimensional vectors is commonly addressed using dedicated techniques often combined with hardware acceleration provided by GPUs, FPGAs, and other custom in-memory silicon. Despite their effectiveness, harmonizing those optimized solutions with other types of searches often poses technological difficulties. For example, to implement a combined text+image multimodal search, we are forced first to query the index of high-dimensional image descriptors and then filter the results based on the textual query or vice versa. This paper proposes a text surrogate technique to translate real-valued vectors into text and index them with a standard textual search engine such as Elasticsearch or Apache Lucene. This technique allows us to perform approximate kNN searches of high-dimensional vectors alongside classical full-text searches natively on a single textual search engine, enabling multimedia queries without sacrificing scalability. Our proposal exploits a combination of vector quantization and scalar quantization. We compared our approach to the existing literature in this field of research, demonstrating a significant improvement in performance through preliminary experimentation.

Keywords: Surrogate Text Representation · Inverted Index · Approximate Search · High-Dimensional Indexing · Very Large Databases

1 Introduction

A key aspect that determined the success of the web was undoubtedly the arrival on the scene of search engines. Although in the beginning, the technology of the vector space model on which they are based was not immune to problems such as spam web pages, they were very efficient, scalable, and flexible. Not surprisingly, it was relatively easy to enhance and integrate them with other technologies such as hyperlink analysis (PageRank) and term proximity.

Underlying the power of search engines are inverted indexes, which in turn exploit the sparseness of the representation of documents to be retrieved. Unfortunately, artificial intelligence models produce learned vectors that are difficult

to deal with using inverted indexes. Neural networks for image or text representations, such as GeM [14] or BERT [7] to mention a few, produce high-dimensional dense vectors that are usually compared with the cosine similarity. This sprouted the development of solutions to solve maximum inner product search problems efficiently. Commonly used data structures exploit inverted indexes in combination with data partitioning techniques, such as Voronoi partition or proximity graphs, to restrict the search to a fraction of the database. Although existing solutions for high-dimensional vector search have proven great performance in terms of speed and accuracy [11,10,12], they still have drawbacks. Their implementation is often hardwired to run on main memory as a dense vector search system and nothing more. Most of them are not a proper database system, so multimodal queries such as images and text cannot be resolved. For example, search for all images similar to a given example image and match certain tags. Other limitations include extensive use of RAM or a lack of mature and transparent mechanisms to ensure scalability, such as fault-tolerance or load balancing. In contrast, NoSQL databases, such as Elasticsearch, can scale horizontally as the data size grows.

In this work, we tackle the problem of maximum inner product search of high-dimensional real-valued vectors using full-text search engines and Surrogate Text Representations (STRs) — a family of transformations to encode metric data into synthetic texts. We contextualize our work in the area of data structures for similarity search of dense vectors in secondary memory. All data structures based on metric spaces (such as M-Tree [6]) would be suitable in theory for this task. However, in this work, we focus mainly on those optimized explicitly for working with dense real-valued vectors. Many efficient vector similarity search approaches based on data partitioning techniques (such as [10,11,12]) use dedicated implementations of access structures such as inverted indexes. STR-based methods, on the other hand, rely on transformations that sparsify data and encode it as small sets of codewords indexed on standard text engines [9,2,4]. These approaches are successfully used to solve multimodal queries for combined text search with image similarity [1,3].

We propose an improved approach combining Voronoi partitioning and STRs. Specifically, we associate a posting list to each Voronoi cell and use STRs to generate the entries of each posting list. Our proposal enables the exploitation of off-the-shelf text search engines, thus supporting combined text+image multimodal search that relies only on text retrieval technologies and platforms without implementing dedicated access methods. Code to reproduce experiments is available at https://github.com/fabiocarrara/str-encoders.

2 Surrogate Text Representation

As we explained in the introduction, our goal is to index and retrieve feature vectors by leveraging commercially available search engines.

Our primary objective is to define a family of transformations that map a feature vector into a textual representation. Of course, we also require that

such transformations preserve the proximity relations between the data as much as possible, i.e., maps similar feature vectors to similar textual documents. To achieve this, we need a transformation $f : \mathbb{R}^d \to \mathbb{N}^m$ that maps each original vector $\boldsymbol{y}$ into a vector $\overline{\boldsymbol{y}}$ whose components are integer-valued. Indeed, the core idea is then interpreting $\overline{\boldsymbol{y}}$ as a term frequency vector with respect to a codebook $\mathcal{C} = \{\tau_1, \ldots, \tau_m\}$ of m terms. The text document associated with the vector $\boldsymbol{y}$ will be a space-separated concatenation of the codebook terms so that τ_i is repeated a number of times equal to $\overline{y}_i$. We indicate with $T_{f,\mathcal{C}}(\cdot)$ the overall transformation from the original vectors to the text documents, which depends on both the function f and the used codebook $\mathcal{C}$. For example if $f(\boldsymbol{y}) = \overline{\boldsymbol{y}} = [2, 0, 1, 3]$ and $\mathcal{C} = \{$"A", "B", "C", "D"$\}$ then the text document associated to $\boldsymbol{y}$ will be $T_{f,\mathcal{C}}(\boldsymbol{y}) =$"A A C D D D". The rationale of this approach is that a full-text search engine based on the *vector space model* [15] will generate a vector representation of the text by counting the number of occurrences of the words in it, i.e., the term frequencies (TF). Therefore, the abstract transformation f represents a function that exactly generates the vectors that are internally represented by the search engine in the case of the simple TF-weighting scheme.

Since this approach is based on transforming the components of a vector $\boldsymbol{y}$ into the term frequencies of a synthetic text document, the employed transformation f should output a vector $\overline{\boldsymbol{y}}$ with positive components (no search engine admits negative TFs even though this in principle would be possible). Moreover, it should provide sparse vectors to ensure having a large number of zero components in the TF vectors and thus a good inverted index efficiency.

These assumptions form the basis of a family of approaches based on what is known as *Surrogate Text Representation* (STR) [9,4]. STR approaches differ primarily in the steps used to deal with negative values, sparsification, and the final real-to-integer discretization. Moreover, it is worth noting that these approaches are designed to solve Maximum Inner Product Searches, where the cosine similarity or the inner product is used to assess the similarity of the original feature vectors. Indeed, this similarity is approximated by the inner product between the associated TF vectors in the vector space model employed by the text search engine.

3 Voronoi Partitioning STR

In this work, we propose a STR technique that employs a Voronoi partitioning of the original features space and a specific codebook for each Voronoi cell. In a nutshell, we use a k-means data partitioning to assign feature vectors to Voronoi cells corresponding to a set of centroids $\{\boldsymbol{c_1}, \ldots, \boldsymbol{c_k}\}$, and then we use a different STR transformation for each Voronoi cell. Specifically, we build a codebook $\mathcal{C}_i = \{\tau_{i,1}, \ldots, \tau_{i,m}\}$ for the i-th cell, and we transforms the vectors in that cell using $T_{f,\mathcal{C}_i}$.

As space transformation f, we employed modified versions of two state-of-the-art STR approaches: the Deep Permutation STR [2] and the Scalar Quantization STR [4] that we briefly review below.

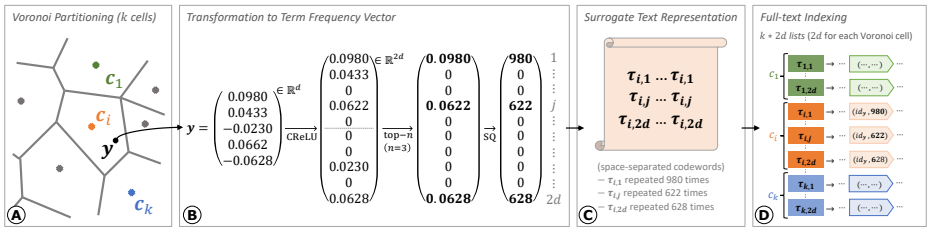

Fig. 1: Overview of the proposed VP-SQ surrogate text representation method. (**A**) The database is Voronoi-partitioned using k-means, and (**B**) elements of each partition are encoded into sparse term-frequency vectors using a surrogate text representation technique (SQ in this case, that produces $2d$-dimensional vectors with n non-zero components). (**C**) Surrogate documents are created by repeating tokens of partition-specific codebooks. (**D**) Documents are indexed using a full-text search engine; all the codebooks form a vocabulary of $2kd$ terms, and each database element is present in exactly n posting lists among the $2d$ ones related to the Voronoi cell containing the element.

Deep Permutation (DP) STR The term frequency vector $\overline{\boldsymbol{y}} = f_{\text{DP}}(\boldsymbol{y})$ is obtained from the original vector $\boldsymbol{y}$ by assigning an integer importance value from 1 to n to the top-n components of $\boldsymbol{y}$ and dropping (setting to zero) all other components. Formally, $\overline{y}_i = \max(r_i - d + n, 0)$, where r_i is the 1-based rank of $\overline{y}_i$ when sorting the components of $\boldsymbol{y}$ in ascending order (e.g., $r = 1$ for the minimum-valued component, and $r = d$ for the maximum-valued one), and d is the dimensionality of the vector. For example, given a real-valued vector $\boldsymbol{y} = [0.5, -0.7, 2.45, -1.2]$, the vector with the ranks in ascending order is $\boldsymbol{r} = [3, 1, 4, 2]$, thus for $n = 2$, $\overline{y}_1 = \max(3 - 4 + 2, 0) = 1$, $\overline{y}_2 = \max(1-4+2, 0) = 0$, and so on, finally getting $f_{\text{DP}}(\boldsymbol{y}) = [1, 0, 2, 0]$. This formulation was initially thought for non-negative (post-ReLU) neural network activations and assigns less importance to negative values that, however, contribute to informativeness in the general case. Thus, Amato et al. [2] proposed to apply the Concatenated Rectified Linear Unit (CReLU) transformation [16], which simply makes an identical copy of vector elements, negates it, concatenates both the original vector and its negation, and then applies ReLU altogether. Formally, $\boldsymbol{y}^+ = \text{CReLU}(\boldsymbol{y}) = \text{ReLU}([\boldsymbol{y}, -\boldsymbol{y}])$, where the $\text{ReLU}(\cdot) = \max(\cdot, 0)$ is applied element-wise. For example, given $\boldsymbol{y} = [0.5, -0.7, 2.49, -1.2]$, its transformed version is $\boldsymbol{y}^+ = [0.5, 0, 2.49, 0, 0, 0.7, 0, 1.2]$. To avoid the imbalance towards positive activations at the expense of negative ones, we use the CReLU transformation before applying f_{DP}. Following the previous example, $f_{\text{DP}}(\boldsymbol{y}^+) = [0, 0, 2, 0, 0, 0, 0, 1]$ for $n = 2$.

Scalar Quantization (SQ) STR The DP method transforms real-valued components into integer-valued ones but completely disregards the value of the original component and how much it contributes to the inner product computation. On the other hand, the SQ method can retain this information. The SQ

STR simply applies Scalar Quantization to vector components to store them as integers. Formally, the Scalar Quantization is a transformation of the form $\boldsymbol{z} \rightarrow \text{floor}(s \cdot \boldsymbol{z})$, where s is a scalar scaling factor and the floor operation is applied element-wise. As mentioned earlier, STR-based approaches must output positive TF vectors. Nonetheless, both negative and positive elements of the original feature vectors contribute to informativeness. The CReLU transformation is applied in the SQ approach as a first step to coping with negative values. To avoid storing all the components, vector sparsification is achieved similarly to DP by zeroing out the least significant components, i.e., keeping the first-n largest components of $\text{CReLU}(\boldsymbol{y})$. For example, for $n = 2$ the sparsified version of the $\text{CReLU}(\boldsymbol{y})$ considered above will be $[0, 0, 2.49, 0, 0, 0, 0, 1.2]$. Then, the final term frequency vector is obtained after scaling and truncation (zero values are left untouched); for $s = 10$, the corresponding SQ of the vector $\boldsymbol{y}$ would be $f_{\text{SQ}}(\boldsymbol{y}) = [0, 0, 24, 0, 0, 0, 0, 12]$.

Note that DP and SQ only differ in the definition of the function f used to associate term frequency vectors to the original feature vectors. However, both these approaches are limited by construction to using a codebook that contains exactly $m = 2d$ terms if using the CReLU, d if using the ReLU, where d is the dimension of the original feature vectors. This means that the total number of posting lists in the inverted index is limited by the dimensionality d as well, which may compromise the efficiency of the search (e.g., if d is too small compared to the size of the dataset, then the inverted index may have few posting lists, but each contains a large fraction of the original dataset). For example, dimensionality reduction techniques (e.g., PCA) are often used to reduce high-dimensional vectors without a considerable loss of effectiveness. However, we may have no advantage in using the DP and SQ STR techniques to index and search these reduced vectors on a large scale.

We propose to use Voronoi Partitioning (VP) on top of the DP and SQ approaches, allowing the disentanglement of the cardinality of the codebook from d and hence the tuning of the number of posting lists. Indeed, our extension of DP and SQ approaches, which we named VP-DP and VP-SQ, allow producing an inverted index with $m = k * 2d$ posting lists, where the number of partitions k can be tuned to guarantee a higher level of efficiency. We obtain k centroids in the original vector space using k-means clustering. Each data vector $\boldsymbol{y}$ is transformed as $T_{f,\mathcal{C}_i}(\boldsymbol{y})$, where i is the index of its closest centroid, $\mathcal{C}_i = \{\tau_{i,1}, \ldots, \tau_{i,m}\}$ is a specific codebook associated to the centroid $\boldsymbol{c}_i$, and f is either f_{SQ} or f_{DP}. Note that each object will be stored in exactly n posting lists related to its closest centroid. Figure 1 shows an example for VP-SQ.

To process a query $\boldsymbol{x}$, we first compute its P closest centroids, $\boldsymbol{c}_{i_1}, \ldots, \boldsymbol{c}_{i_P}$, and then we transform the query vector into the text document obtained by concatenating the texts $T_{f,\mathcal{C}_{i_h}}(\boldsymbol{x})$ for all $h = 1, \ldots, P$. This corresponds to accessing nP posting lists, i.e., n posting lists for the P Voronoi cells closest to the query.

4 Experiments

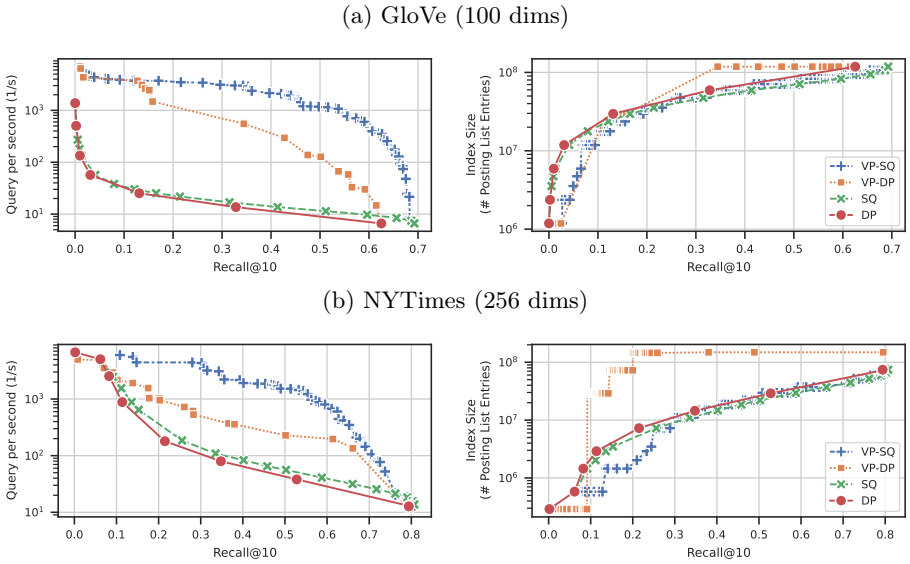

Fig. 2: Time (Query per seconds, left column, top-right is better) and Space Efficiency (n. of elements, right column, bottom-right is better) versus Effectiveness (Recall@10). We only plot configurations belonging to the Pareto frontier.

Datasets. We adopt the GloVe-100 and NYTimes-256 benchmarks for maximum inner product search prepared by Aumüller et al. [5] for a preliminary evaluation of the proposed methods. GloVe-100 [13] is a collection of more than one million 100-dimensional real-valued vectors representing word embeddings learned in an unsupervised fashion. NYTimes-256 [8] is a collection of 280k 256-dimensional real-valued vectors containing bag-of-word-derived document representations of NYTimes news articles. Both datasets provide a set of 10k test queries and the corresponding 100 nearest neighbors for each query. We normalize all vectors to the unitary L_2 norm to implement the intended scoring function (cosine similarity) as the inner product between vectors.

Tested Configurations. We encoded all vectors (data and queries) using DP, SQ, VP-DP, and VP-SQ, obtaining surrogate term-frequencies vectors as sparse integer matrices. For VP-SQ and VP-DP, we vary the number of k-means centroids $k \in \{128, 256, 512, 1024, 2048, 4096, 8192\}$ and the number of voronoi cells accessed at query-time $P \in \{1, 2, 5, 10, 25, 50\}$. For SQ and VP-SQ, we use a scalar quantization factor $s = 10^5$. For all methods, we vary the number of kept

elements n for each vector from 1% to 100% of the original vector dimensionality d. For simplicity, we skip the configurations providing a query throughput lower than ten queries per second (query time $>$ 100 ms).

Implementation Details. We perform experiments on a Ubuntu 20.04 server with Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz and 64GB of RAM. To isolate the evaluation of our proposal from the specifics of a particular textual search engine, we simulated the full-text search on surrogate texts by using SciPy sparse matrix multiplication on encoded vectors. This is only possible as long as all encoded vectors fit in RAM; despite being feasible given the scale of these preliminary benchmarks and our hardware, we suggest using fully-featured disk-based textual search engines, such as Elasticsearch or Apache Solr, to implement larger-scale and more efficient searches. The results of our simulated search can be interpreted as lower bounds to search and storage efficiency that can be boosted using dedicated software.

Results and Discussion. Figure 2 reports the query times (as the number of queries per second, left column) and index storage occupation (as the number of non-zero elements of encoded vectors, right column) as a function of the search effectiveness measured by the Recall@10. For each method, we report only the configurations that belong to the Pareto frontier. We note that VP-SQ dominates the other methods in the time-effectiveness trade-off. Both VP methods improve on their non-VP variants, with VP-SQ deriving a more significant benefit than VP-DP. Concerning the space-effectiveness trade-off, we observe a slight improvement of VP-SQ with respect to non-VP methods in the NYTimes benchmark for low recall regimes, whereas VP-DP usually needs more space to reach higher recalls.

5 Conclusions

In this paper, we proposed a new method for out-of-core similarity search of dense vectors. We mainly target those who need to scale over large amounts of data using an integrated search framework based on a standard search engine. Compared to the state of the art, we improved the performance of surrogate text-based techniques that had the major limitation of working with codebooks constrained by the dimensionality of the dense vectors to be searched.

A key aspect of our approach entails the combination of vector partitioning technique with existing approaches allowing us to expand the codebook used for indexing and thus better fine-tune performance. In the near future, we plan to try to improve our technique by using artificial intelligence-based approaches to learn vector sparsification without sacrificing too much search accuracy.

Acknowledgements This work was partially funded by AI4Media - A European Excellence Centre for Media, Society, and Democracy (EC, H2020 n. 951911).

References

1. Amato, G., Bolettieri, P., Carrara, F., Debole, F., Falchi, F., Gennaro, C., Vadicamo, L., Vairo, C.: The VISIONE video search system: exploiting off-the-shelf text search engines for large-scale video retrieval. J. Imaging **7**(5), 76 (2021)
2. Amato, G., Bolettieri, P., Carrara, F., Falchi, F., Gennaro, C.: Large-scale image retrieval with elasticsearch. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 925–928 (2018)
3. Amato, G., Bolettieri, P., Carrara, F., Falchi, F., Gennaro, C., Messina, N., Vadicamo, L., Vairo, C.: Visione at video browser showdown 2022. In: MultiMedia Modeling. pp. 543–548. Springer International Publishing, Cham (2022)
4. Amato, G., Carrara, F., Falchi, F., Gennaro, C., Vadicamo, L.: Large-scale instance-level image retrieval. Inf. Process. Manage. **57**(6), 102100 (2020)
5. Aumüller, M., Bernhardsson, E., Faithfull, A.: ANN-Benchmarks: a benchmarking tool for approximate nearest neighbor algorithms. Inf Syst . **87**, 101374 (2020)
6. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: Vldb. vol. 97, pp. 426–435 (1997)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186 (2019)
8. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
9. Gennaro, C., Amato, G., Bolettieri, P., Savino, P.: An approach to content-based image retrieval based on the lucene search engine library. In: International Conference on Theory and Practice of Digital Libraries. pp. 55–66. Springer (2010)
10. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. Pattern Anal. Mach. Intell. **33**(1), 117–128 (2010)
11. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2019)
12. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. Pattern Anal. Mach. Intell. **42**(4), 824–836 (2018)
13. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)
14. Revaud, J., Almazan, J., Rezende, R., de Souza, C.: Learning with average precision: Training image retrieval with a listwise loss. In: International Conference on Computer Vision. pp. 5106–5115. IEEE (2019)
15. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York, NY, USA (1986)
16. Shang, W., Sohn, K., Almeida, D., Lee, H.: Understanding and improving convolutional neural networks via concatenated rectified linear units. In: Proceedings of the 33rd International Conference on Machine Learning. ICML 2016, vol. 48, pp. 2217–2225. JMLR.org (2016)