GSI's Elasticsearch k-NN Plugin Application Brief

Introduction

The GSI Elasticsearch k-NN plugin allows you to perform nearest neighbor vector similarity search using Elasticsearch's dense_vector type. The plugin provides a high-performance, low-latency, low-power, billion-scale vector similarity search solution that allows users to combine traditional Elasticsearch text filters with vector search queries for a more advanced search.

Elasticsearch was originally designed as a text and document search engine. The GSI Elasticsearch k-NN plugin expands Elasticsearch's ability to search beyond just text. The plugin opens the door to other data types like images, video, audio—any data type that can be represented as a compact, semantically rich numeric vector. Vectors can be used to search for the most similar items (nearest neighbors) to a query and can accelerate applications such as visual search, face recognition, natural language processing (NLP), and recommendation systems.

Scales to Billions of Documents

Core Elasticsearch uses a computationally-expensive exhaustive match_all which makes it too slow to handle the large-scale initial retrieval step in a vector similarity search pipeline. This limits core Elasticsticsearch to scoring documents on a small, filtered set of vectors.

Instead of using an exhaustive match_all search, the GSI plugin performs an approximate nearest neighbor vector similarity search. This allows the GSI plugin to scale to billions of documents and handle the important initial retrieval step in a search pipeline.

Supports Multimodal Search

Multimodal search, where images and text are combined to form a powerful search, is a rapidly emerging trend. Retail segments, such as fashion and home design, are one particular driver of multimodal search because they rely heavily on visual search since style is often difficult to describe using text.

In addition to visual search, text search is also a required part of the solution because product information, such as item description, item title, category, and brand are generally used to filter the results that are returned as part of the visual search. Thus, what is needed is a solution that allows for multimodal search.

Open-source search libraries do not handle multimodal search well. The popular open-source vector search libraries, such as FAISS and NMSLIB, are good at nearest neighbor vector search, but lack support for efficient filtering of data. For example, it would be difficult for FAISS to use textual filters to filter the results previously returned from a nearest neighbor vector search.

The GSI Elasticsearch plugin allows users to perform multimodal searches. For example, as seen in **Figure 1** below, a user could first perform a vector similarity search to find similar images (vector) to a query image and then filter those results using a color (text) filter.

```
{
  "query": {
     "bool": {
        "must": [{
             "gsi_similarity": {
                "field": "data",
                "vector": [1.21439, 3.10212, -1.16291],
                "topk": 20
             }
          }
        ],
        "filter": [{
             "term": {
                "color": "red"
             }
          }
        ]
     }
  }
}
```

Figure 1: Multimodal Query Example
A user first finds the closest 20 neighbors (topk:20) to the query "vector" and then applies a text filter using product information to find a subset of those items that are the color "red".

No Need to Reindex Documents

Other plugins require custom field type(s) and codecs to support vector search, so a user who already has indices using the dense_vector field type needs to change his index mapping and reindex their documents to use those other plugins. This reindexing is costly since most other vector plugins use graph-based approaches for search, which suffer from slow indexing and memory-intensive indexing.

As seen in **Figure 2** below, the GSI Elasticsearch k-NN plugin uses the core Elasticsearch dense_vector field type and index mapping, so there is no need to reindex.

```
{
 "mappings": {
  "properties": {
   "my_vector1": {
    "type": "dense_vector",
    "dims": 2
   }
  }
 }
}
```

Figure 2: GSI Elasticsearch k-NN Plugin Example
The GSI Elasticsearch plugin requires no reindexing because
the mapping for the dense_vector field type is the same as core Elasticsearch's.

Provides a Simple, Ready-for-Production Vector Search Architecture

Popular open-source vector search libraries, such as FAISS and NMSLIB, provide good results for nearest neighbor benchmarks, but they are not easily integrated into a production search system. For example, a user would still need to address the difficult tasks of building a distributed system that scales and managing the complex indexing of those distributed systems.

With Elasticsearch, that is already taken care of because two of its key strengths are horizontal scaling and index management. The GSI Elasticsearch k-NN plugin leverages those strengths and extends the power of Elasticsearch even further by integrating nearest neighbor vector similarity search directly into Elasticsearch. The plugin uses the existing Elasticsearch query interface, resulting in a simple-to-use, ready-for-production vector similarity search solution.

High Performance with Low Power

Benchmark #1

Based on: https://github.com/jobergum/dense-vector-ranking-performance

GIST-960-Euclidean Results

Engine	QPS	Average Latency (ms)	P95 Latency (ms)	Recall@10
Elastic 7.6	0.57	1752.74	1850.74	1
Vespa 7.190.14	1.32	756.61	955.63	1
GSI Plugin	8.1	123	135	1

GSI Plugin - APU Power: 16.1W, CPU Power: 225W-----Total: 241.1W

CPU Power: 325W (1 x Intel Xeon E5-2680 v3 2.50GHz Haswell)

SIFT-128-Euclidean Results

Engine	QPS	Average Latency (ms)	P95 Latency (ms)	Recall@10
Elastic 7.6	3.29	303.96	337.89	1
Vespa 7.190.14	9.14	109.33	148.9	1
GSI Plugin	15.8	63.3	64.7	1

GSI Plugin - APU Power: 15.85W, CPU Power: 225W-----Total: 240.85W

CPU Power: 325W (1 x Intel Xeon E5-2680 v3 2.50GHz Haswell)

Benchmark #2: Demo

DB=Fashion (700K)

Dataset	Size	Engine	Platform	QPS	Average Latency (sec) Per Query
Fashion	700K	Elastic 7.8.0	Xeon Gold 5115	0.8	3.31
Fashion	700K	GSI Plugin	APU: One Query	20	0.05
Fashion	700K	GSI Plugin	APU: Batch of 10 Queries	78.13	0.0119

DB=ImageNet (14M)

Dataset	Size	Engine	Platform	QPS	Average Latency (sec) Per Query
ImageNet	14M	Elastic 7.8.0	Xeon Gold 5115	0.30	3.31
ImageNet	14M	GSI Plugin	APU: One Query	10	0.1
ImageNet	14M	GSI Plugin	APU: Batch of 10 Queries	54	0.0185

Contact us at elasticsearch@gsitechnology.com for a customized demo.