Benchmarking Vector databases on Code embeddings

Raymond Chang (rkchang)
rkchang@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Vikram N. Subramanian(vnsubram)
vnsubram@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Yahya Jabary
yahya.jabary@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

1 Problem Area

Vector Embeddings are a critical data structure used in applications that rely on Large Language Models (LLMs). Embeddings are generated by feeding documents into a model. The structure of a vector embedding is essentially an array of floats of length n and would describe a point in an n-dimensional space. The similarity between two embeddings can be compared via strategies such as cosine similarity where you take the cosine of the angle between the vectors [1].

Mature databases are not currently optimized for the storing or querying of these embeddings. Thus a new wave of databases (Vector databases) have emerged that are specifically designed to handle this workload. When querying a vector database, the input would be a vector and the output would be the nearest neighbours of the vector[1]. As this area is fairly new, existing benchmarks of vector databases are not mature. In this project, we seek to expand on existing work on the area by measuring existing metrics such as queries per second with a larger data set of higher dimensional vectors.

2 Promised Deliverables

1. **Benchmarking of Vector databases on a low resource machine:** Being able to support a large number of requests (and consequently users) is a very important requirement of vector databases. To the best of our knowledge, we were not able to find benchmarks of open-source vector databases running on low-resource systems. We hypothesize that a low-resource system will act as a good proxy for a system that is being heavily utilized while also being very easy to saturate with requests and therefore very easy to test the performance under load.
   This bench-marking is also valuable as we can evaluate the possibility of on-device vector databases and querying in mobile and embedded devices. For instance, using a vector database in a mobile app.
2. **Data set of code embeddings:** We identified a gap in data sets used to benchmark vector databases. Several data sets including image data sets have been created and used to benchmark vector databases but no code dataset exists to the best of our knowledge. Code search is a very popular use case for vector databases and code has unique characteristics that differentiate it from regular text when it comes to embeddings produced and how they are distributed in a vector space. One metric we seek to measure is the fraction of retrieved vectors that are near the query vector (Precision). An important issue we face is that given a block of code and its corresponding vector, we must know in our data set what other blocks and their corresponding vector may be semantically similar. Instead of manually labelling blocks, we plan to instead generate new blocks by mutating existing code in the data set such that the mutated code is still semantically the same as the original. An example of a semantics preserving mutation is below:

Listing 1. Original Code

```
int foo(){
    return 2;
}
```

Listing 2. Mutated Code

```
int foo(){
    if (false) {
        return 1;
    }
    return 2;
}
```

3. **Large dataset of high dimensional embeddings**: A popular embedding model currently used is OpenAI's embedding model. It produces embeddings that have 1536 dimensions. To the best of our knowledge, there exists no big dataset with embeddings of that dimensions. The dataset we produce will at least have 500,000 embeddings and have 1536 dimensions.

3 Metrics

We shall benchmark the performance of vector databases using the following metrics

1. **Latency** Average time to respond to requests as the number of requests/second increases
2. **Queries per second:** How many requests the vector database can process before it has to queue requests.
3. **Precision:** To achieve scalability and performance, many vector databases implement approximate nearest neighbour algorithms. These algorithms trade off

a small amount of accuracy (precision) for significant gains in speed and reduced computational load. They do not always guarantee the absolute nearest neighbour but provide a very close approximation. As load increases, precision usually reduces.

1. **Vector Databases:** There exist over 15 open-source vector databases. Popular ones include FAISSM, Qdrant and Elasticsearch. We will evaluate the most popular ones.

4 Schedule

- November 6th: Decide on what vector databases to benchmark
- November 10th: Collect code from github that will serve as original dataset
- November 17th: Generate new code via semantics preserving mutations
- November 24th: Perform benchmarks on new code and original dataset

References

[1] Roie Schwaber-Cohen. What is a Vector Database & How Does it Work? Use Cases + Examples | Pinecone.