HW1 Lab 094295 - Report

Amit Zalle, Guy Shoef, Omer Yom Tov

July 2024

A link to github repo (click me!)

1 Faiss

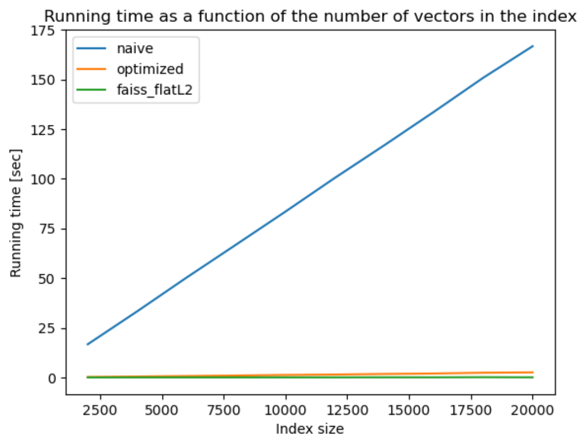

Figure 1: (1.1.1)

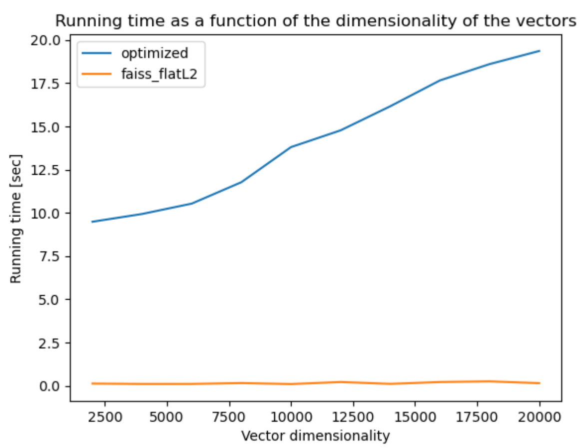

Figure 2: (1.1.2)

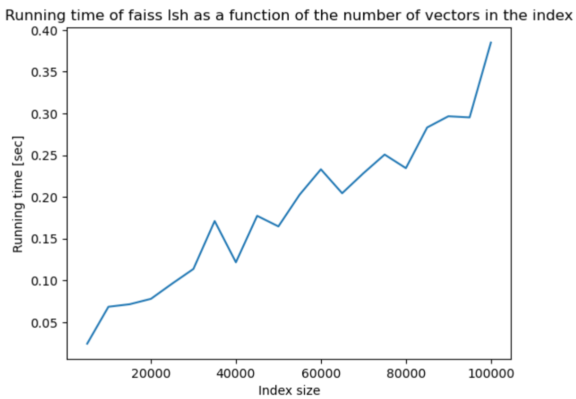

Figure 3: (1.2.1)

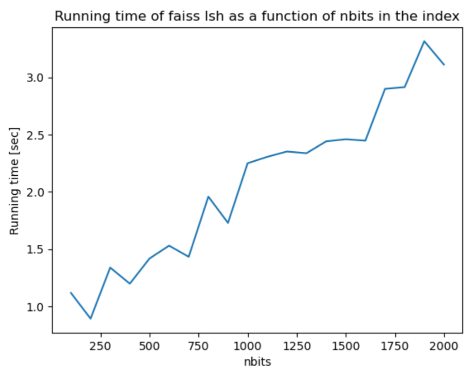

Figure 4: (1.2.2)

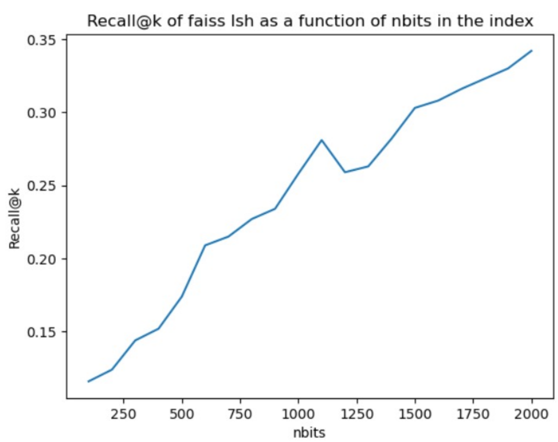

Figure 5: (1.2.3)

2 Implementing an Index

2.1 LSH vs. Naive Exhaustive Search

- Running time of the **semi_optimized_exhaustive_search** function: 13.4 *sec*

- Running time of creating **faiss_LSG_index**: 1.33 *sec*
- Running time of faiss search over query vectors with **faiss_LSH_index**: 7.43 *ms*
- Recall@10 for **faiss_LSH_index**: 0.138

2.2 Custom Index Algorithm

2.2.1 Explanation of Indexing Algorithm

We started with a simple LSH we learned in class. using hyper-planes set uniformly we devide the data into cells in hash table. however, the uniform sampling result in all vectors in the same cell, so we switched to partition based PCA and then normal sampling. the last one show good result, but since it was random was unstable. to fix that we created 50 indexes, and chose the one that spread the data among the cells the best. this stableized the time issue, but still wasnt perfect in terms of recall.

so, we displayed the data using PCA(in dim 2) and thats when we saw that it is clearley clustered into 10 clusters. using that we converted the data into PCA of dimension 2 and performed clustering using MiniBatchKmeans we found online.

Then for each new query vector we found the cluster it fits, and applied brute force on all vectors in this cluster. which result highly both in time and recall, on all datas and k values.

The logic is that if the data is spread in clusteres, when a query fit into one cluster, it is most likely all vectors for it will come from the same cluster since it is close to that vector.

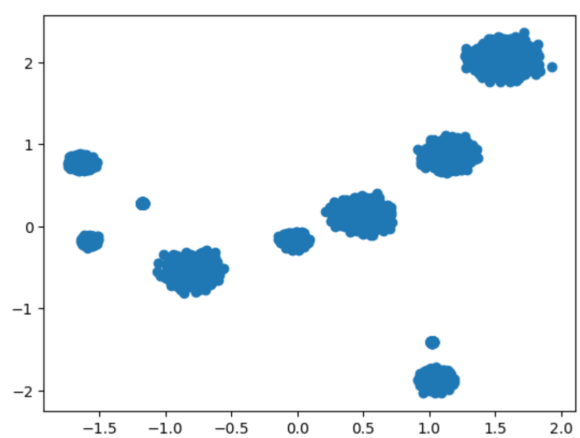

Figure 6: PCA representation of the data, for efficient clustering in indexing algorithm

2.2.2 Results

- Running time of creating **custom_indexing_algorithm**: 255 *ms*
- Running time of searching over query vectors with **custom_index_search**: 1.27 *sec*
- Recall@10 for **custom_index_search**: 1

3 Pincone vectorDB and RAG

3.1 Dataset Selection

We selected SQuAD (Stanford Question Answering Dataset) Dataset. It's a reading comprehension and question answering dataset from Stanford, which consists of many general knowledge and trivia questions and documents that contains answers for those questions, mostly from Wikipedia. We chose this dataset, as it consists of many topics, some might be a weakness of our LLM.

3.2 Failure Examples

- Q: *"what can the process of squering a num can be reduced to?"*
- A: "The process of squaring a number can be reduced to exponentiation. When you square a number, you are essentially raising it to the power of two. So, instead of writing "n squared" or n^2 you can write it as n^2 or simply "n to the power of 2." This notation is commonly used in mathematics and computer science to represent exponentiation."
- Q: *"$\{xx|x \in binary_strings\}$can be solved in what time?"*
- A: "The given regular expression is xx — x is any binary string. To solve this, we can break it down as follows:

  - "xx" represents a pair of binary digits, which can be either 00, 01, 10, or 11.
  - "—" is the alternation operator, indicating that we have a choice between the options before and after the "—".
  - "x is any binary string" means we can have any combination of 0's and 1's for the value of x.

  Therefore, the regular expression can be simplified as:

  (00 + 01 + 10 + 11)x

  This expression allows for any repetition of the pairs 00, 01, 10, or 11, which essentially covers all possible binary strings.

  So, the regular expression xx — x is any binary string can be solved in linear time, as we can process each pair of binary digits in constant time and repeat this process for any length of the binary string."
- Q: *"give me two well-known complexity resources"*
- A: Here are two well-known complexity resources:

  1. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: This book is widely considered a standard text on algorithms and includes a comprehensive discussion of complexity analysis and a range of algorithms.

  2. "The Design and Analysis of Computer Algorithms" by Aho, Hopcroft, and Ullman: Another classic text that covers algorithm design paradigms, analysis techniques, and a wide range of algorithms, with a focus on providing a rigorous treatment of the subject.

  These books provide in-depth explanations of various complexity classes, algorithm design techniques, and analysis methods, making them valuable resources for anyone studying or working in the field of computer science.

3.3 Prompts and Outputs

3.3.1 LLM prompts with context

"Using the contexts below, answer the query. Contexts: {source_knowledge} If the answer is not included in the source knowledge - say that you don't know. Query: {query}"

3.3.2 Outputs

- Q: *"what can the process of squering a num can be reduced to?"*
- A: The process of squaring a number can be reduced to repeated multiplication.
- Q: "$\{xx|x \in binary\_strings\}$*can be solved in what time?"*
- A: The context states that the language xx — x is any binary string can be solved in linear time on a multi-tape Turing machine.
- Q: *"give me two well-known complexity resources"*
- A: The two most well-known complexity resources are time and space.

3.4 RAG pipeline insights

We chose specifically questions that comes from the domain of complexity theory - a branch of computer science and mathematics, which are known as difficult domains for LLMs to deal with coherently. While the LLM alone gave wrong answers, after using the RAG pipeline, it was able to answer somewhat correctly. We assume that the RAG gives the LLM context that contains the actual answers, and when using it in the generation, the LLM manages to use the context to answer correctly. We also assume that it helps the LLM focus on isoteric semantic connections between terms that the LLM did not learned well in pre-training, like being able to state that there is not a known solution for a specific problem.