Building a Semantic Search Engine with Vectorized Databases

Introduction:
The objective of this project is to design and implement an indexing system for a semantic search database. The system should be able to efficiently retrieve information based on vector space embeddings. The project will focus on building an indexing mechanism for a vector column only.

Semantic Search:
Semantic search is a technology that enables search engines to understand the meaning of the search queries and provide relevant results based on the context and intent of the searcher. Unlike traditional keyword-based search methods, which rely solely on matching keywords, semantic search uses natural language processing and machine learning algorithms to analyse the relationships between words, phrases, and concepts, and retrieve results that match the underlying intent of the search query. This means that even if a student enters a query using different words or phrasing than what's contained in the database, the semantic search engine can still return relevant results because it understands the context and intent of the search. For example, if a student searches for "What are the best ways to study effectively?", a semantic search engine would return results related to studying tips, time management strategies, and productivity techniques, rather than simply returning results that contain the individual keywords "study," "effectively," and "ways." Please read this page for more about semantic search and why it is used.

Here is a full scenario for semantic search, example of images search engine:
From the user's perspective:
The user wants to search for images that are similar to the one they have. They start by uploading the image and waiting for the search results. The system responds with a set of 10 images that are related to the original image. If the user doesn't find what they're looking for in this initial batch, they can click through to the next page to see another 10 images. This process continues until the user finds what they need, at which point they can exit the session.
From the system's perspective:
To enable this functionality, the system must first crawl the entire internet to gather all available images. It then passes each image through an AI model that converts it into a vector representation, which captures the essence of the image. The system stores both the image and its corresponding vector in a database for future reference. When a user submits a new image, the system passes it through the same AI model to generate a vector representation. It then performs a search using this vector to identify the top 10 vectors that are most similar to it. Finally, the system retrieves the images associated with these vectors and displays them to the user. If the user requests additional pages of results, the system repeats this process to find the top 20 similar images, and so on.

Project Scope:
The project will require the development of an indexing system that can efficiently store and retrieve data based on vector space embeddings. The following requirements must be met:

- The database must have only two columns (ID, embedding -vector of dim 70-)
- The database must have an indexing system that retrieve the top_k most similar rows to the given input
- The indexing system must be able to handle large datasets (up to 20 million).
- The indexing system must be able to efficiently retrieve the top 'k' results for a given query (k is up to 10).
- The system must response in a reasonable time

Here's a comprehensive overview of the project:
We (TAs) will provide you with vectors, as if they were generated by an AI model. Your task is to store them and create an index for the vector column. Next, we'll give you a vector, as if it's the image to search for, along with the number of desired vectors to retrieve (k). Your job is to retrieve the most similar vectors to the given one. Cosine similarity will be used to calculate the similarity between vectors. We'll assess the quality of the retrieved vectors by comparing them to the actual most similar vectors. While approximations are acceptable, accuracy is important. We'll also evaluate the time it takes to retrieve the vectors.

Deliverables:

- Design Document: A detailed design document outlining the proposed indexing system, including its architecture, data structures, and algorithms. The document should also discuss the reasoning behind the design choices and any trade-offs made during the development process.
- Implementation: An implemented indexing system that meets the requirements listed above. The implementation should be written in Python and you shouldn't use any database management system (e.g., MySQL, PostgreSQL).

Evaluation Criteria:
The project will be evaluated based on the following criteria:

1. Accuracy (Recall): The indexing system must accurately retrieve the top 'k' results for a given query.
2. Efficiency: The indexing system must efficiently retrieve the results, with a reasonable query time and memory usage.
3. Scalability: The indexing system must be able to handle large datasets and scale appropriately.
4. Documentation: The design document must be well-written and provide sufficient detail for someone to understand the indexing system.

Timeline:

- Week 7: Submit the initial version of the design document.
- Week 11: Submit final deliverables (final version of the design document and the implementation). Further instructions of how to submit the code will be announced later

This project document provides a general outline of the project requirements and expectations. The details of the project may change based on the needs and goals of the team, but the core objectives remain the same. Good luck with your project!

bnt7asb 3la wa2t bta3 al retrieve m4 al insert

al retrieve of the k vectors with some probability of each

no evaluation on the memory but without memory crash on colab

m4 bnt7asb 3la al disk storage