שמאטע (Rag in Yiddish)

http://exodao.net

SCHMATE (pronounced SHMAH-teh) : Extending re-Isearch with vector datatypes for embeddings.

August 2024 Draft 1.16 Verbose // Edward C. Zimmermann

Re-Isearch is a 100% open source (Apache 2.0) novel multimodal search and retrieval engine using mathematical models and algorithms different from the all-too-common inverted index. It is a kind of hybrid between full-text, XML, object and graph noSQL-db that natively ingests a wide range of document types and formats. It has been open-sourced through a grant from Nlnet/NGI-Zero Search. See our talk from FOSDEM '22: A lightning intro to re-Isearch.

The re-Isearch engine exploits document structure, both implicit (XML and other markup) and explicit (visual groupings such as paragraph), to zero in on relevant sections of documents, not just links to documents. These are a heterogeneous mix of text, data (a large number of datatypes including: numerical, computed, range, date, time, geo, boolean etc. as well as a number of hashes including several phonetic), network objects and databases. These datatypes have their own, for their individual datatypes, storage and retrieval algorithms (including relevant ranking and similarity methods). Project Schmate intends to extend re-Isearch with a flat vector datatype tuned for embeddings.

This new datatypes are intended to provide a powerful alternative to popular vector databases like FAISS for Dense Passage Retrieval (DPR) in, among other domains, Retrieval Augmented Generation (RAG) applied to popular open source context constrained large language models. LLaMa/LLaMa2/LLaMa3 have, for example, a relatively small 2k, resp. generally 4k and 8k context while Mistral-7B has a context length of 32k, This is still too small to be able to include the whole local or updated corpus but only some bits (passages). RAG (Retrieval Augmented Generation) is a means to try to maneuver out of this constraint but because of the fixed unit of retrieval in typical vector databases used for DPR (Dense Passage Retrieval) they demand a prior segmentation of content into size constrained blobs or passages.

This is where re-Isearch and its proposed new datatypes and extensions enter. Since re-Isearch has a fully dynamic unit of retrieval, definable at search time or by heuristic, its simplifies the creation and maintenance of DPR systems and provides a significant advantage for, among other applications, RAG. There are, of course, a myriad of other uses.

While the current re-Isearch can be used for vector search of embeddings without an internal datatype optimized for the function using a bi-encoder architecture to encode queries and nodes into the desired dense vectors its performance is sub-optimal on this specific task to some of the vector engines such as Meta's FAISS with its dedicated vector store. The herein proposed datatype should provide significant performance improvements at least en-par with, for example, FAISS or uSearch.

Because of the "curse of dimensionality" and our design to support higher dimensioned vectors we are restricted to approximate nearest neighbor (ANN) rather than k nearest neighbor (K-NN) as the later can get quite expensive. We selected to focus on graph based rather than partition-based indexes—like LSH, IVF or SCANN)— as they are both fast and may be incrementally updated.

While the Hierarchical Navigable Small World (HNSW) algorithm is easy to implement and deploy—there is a popular C++ header based implementation—its excellent performance degrades when the sets are large and stored on disk rather than RAM. More optimal structures as sets get large must build on flat rather than hierarchical graphs. We ultimately need a number of algorithms.

Under the hood these are all indeed flat indexes—long used by re-Isearch for caching record presentations—encoding objects (here vectors) of a fixed size and so may be linearly addressed on disk. As with the core engine itself, we shall use memory mapping to exploit the virtual memory system rather than file I/O intro intermediate buffers. This memory itself may still be compressed.

Appendium: Typical RAG challenges or why Re-Isearch+Schmatte for DPR?

While creating a prototype RAG application is these days comparatively easy thanks to sites like LangChain and HuggingFace, making it work well much less performant, robust, or scalable to a large knowledge corpus has proven for many organizations as quite difficult.

1. **Ingest**

   (a) Data Extraction

   Extracting data from diverse types of documents, such as emails, PDFs and office files such as ODF can be challenging. Documents have generally both explicit and implicit structure (text formats of what is typically called unstructured is not really without structure). The re-Isearch engine already understands these (and more than 80 base types and with these 100s more and with a plugin-in architecture easily extended to support new formats). Formats whose contents can be directly addressed are natively indexed without need for an intermediate (such as JSON or XML). Because of our algorithms we don't have any restrictions on word length, word frequency, number of fields or paths and support a polymorphic indexing to more than 30 data types (date fields, for example, can be indexed and searched as both dates, using any of a number of detected format conventions, as well as the words or phases in its expression). Because of our design ingest is not just extremely powerful but also extremely fast and also resource efficient.

   (b) Chunk Size and Chunking strategy

   i. Finding the optimal chunk size for dividing documents into manageable parts for passage retrieval is generally a challenge. Larger chunks may contain more relevant information but can reduce retrieval efficiency and increase processing time. Finding the optimal balance is crucial.

   ii. Most RAG system demand careful consideration on deciding how to partition the data into chunks choosing typically between size, sentence-based or paragraph-based chunking. This does not properly exploit contextual scope and often leads to misaligned context.

   (c) Robustness and scalability

   In order to be robust and scale one needs a modular and distributed system.

2. Search/Retrieval

   (a) Because of their chunking size and strategy retrieved passages may sometimes been misaligned to context. Because we support multi-indexes, have a scope aware passage retrieval system-- we retrieve

not just the chunks in the similarity top-k but view these in their contextual scope to retrieve a scope that is more inclusive of probable context.

(b) We also do, among many other things, query augmentation to help contextualize and generate accurate responses, We can also deploy query routing to the most appropriate domain (index or even virtual collection of indexes).

3. Data Security

While Retrieval Augmented text Generation (RAG[1]) addresses the issues of pre-training, allowing for (some) on-the-fly changes they depend upon the ability of the underlying DPR system to retrieve quality context fragments.

Instead of purely using the retrieved passages and feeding them into the LLM we look at the scope of the retrieved passages to determine a more optimal passage response. Our approach is to exploit data (document) structure to define the chunks. On the search retrieval side (full-text retrieval) alongside the search for similar chunks we have a retrieval of its own model of relevant bits using its ability to fully retrieve structural elements without need to re-parse: dynamic unit of retrieval. This approach maintains the original author's organization of content and helps keep the text coherent. It builds on the idea that documents are implicitly structured to be understood by humans using either explicit markup or implicit structure such as lines, sentences or paragraphs. It also builds on the notion that meaning is communicated through also structure so needs to be viewed in the context of structure. Rather than just use the chunk we can use the contextual scope—with, of course, a size constraint definable at search time—and also, when desired or warranted provide the traceback.

The re-Isearch engine can at search-time respond to geo-location, user rights or other issues which may define what constitutes as “inappropriate” as it is just toggling a single bit. This feature can be quite useful for retrieval augmentation.

While we would delight in developers joining our ExoDAO moon-shot, this project is focused on re-Isearch and is fully de-coupled from a dependence upon participation and 100% useable in and of itself.

1 Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks