# Project שמאטע: Schmate* (Pronounced “SHMAH-teh)

**Extending re-Isearch with vector datatypes for embeddings. Enabling a better dense passage retrieval (DPR) for retrieval augmented generation (RAG) and hybrid search.**

**(*) Schmate means RAG in Yiddish**

Edward C. Zimmermann // re-Isearch.org // 2024

# At its core is Re-Isearch

*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.

https://archive.fosdem.org/2022/schedule/event/lt_re_lsearch/

# (Re-)Isearch History

## The past 30 years

- Isearch was a legendary open-source text retrieval software first developed in 1994 as part of the Isite Z39.50 information framework with support from NSF.
- Development was divided between CNDIR/MCNC (North Carolina, USA) and Bsn (Munich, Germany).
- In 1998 BSn launched a proprietary fork called IB using new algorithms that significantly improved performance and power.
- IB development shifted to BSn’s research arm: NONMONOTONIC Lab. In was deployed in 100s of high profile sites though BSn projects as well as partners.
- In 2011 Bsn/NONMONOTONIC's proprietary fork ceased with Ed Zimmermann leaving his active role in BSn and joining CIB.
- The software moved to the proverbial attic.
- Even ten years on from End-of-Development and despite lack of support some servers were still running!??? Don't break a working system.....

# Who used it? Among others…

The U.S. Patent and Trademark Office (USPTO) patent search, the Federal Geographic Data Clearinghouse (FGDC), the NASA Global Change Master Directory, the NASA EOS Guide System, the NASA Catalog Interoperability Project, the astronomical pre-print service based at the Space Telescope Science Institute, The PCT Electronic Gazette at the World Intellectual Property Organization (WIPO), the SAGE Project of the Special Collections Department at Emory University, Eco Companion Australasia (an environmental geospatial resources catalog), the Open Directory Project, genomic search for the Australian National Genomic Information Service's (ANGIS) human genome project (and its eBiotechnology workbench split-off); the D-A-S-H search portal against racism, antisemitism and exclusion (funded within the framework of the action program "Youth for tolerance and democracy - against right-wing extremism, xenophobia and anti-Semitism", the YOUTH program of the European Community and with additional support from the German Federal Agency for Civic Education); the e-government search (Yeehaw) of the U.S. State of Utah to agronomic cooperation across the Mediterranean region (supported by the EU’s DG). Integrated into a number of CMS platforms it powered search for a number of high volume web sites. It was also used as a database accelerator by a number of eCommerce shops.

# Re-Isearch. Its algorithms

Reborn in 2020 in the middle of the global Covid19 pandemic as Project re-Isearch catalysed by a grant from NInet/ NGI-Zero. Like the original it is not just about textual words but pushes the envelope.

- Based roughly on pat-arrays going back to the original work of Gonnet, Baeza-Yates, Uri Mamber et al.
- The algorithm is accelerated by a simple trick: In contrast to depending just on the addresses it also stores a cache of the first X characters and the address range in the index to significantly save on system calls and I/O. Since this can be mapped into memory it super-scales to multiple processes accessing the same indexes.
- Each word has a composite address made up from the address (key) of the file it is contained in, the file offset to the start of record (a file can contain multiple records) and the offset from the start of record to the start of the word.
- Each field (container) instance encountered gets added in a separate file its start and end address stored. We use multiple files to better exploit individual file system features, configurations and the potential tuning thereof..
- The vector of the pairs <word, address> is sorted by word. The address column is dumped to disk, the addresses whose words match the first and/or last X characters is dumped as < word fragement, start offset, end offset> where are offsets are into the index above.
- The address of every word is stored and the address range of every field instance is stored.
- An index file tracks the correlation of composite address to file and record.
- Should a path/field/node be of an extended datatype, a handler parses the contents and adds it to a datatype optimised index (date, numerical, geospatial etc.).

# What does this deliver?

## Unlimited term length, unlimited fields, wildcards and deep search

- For every word we have an address and can open the file and read it. The word could have been encoded. No general need for an intermediate file store (such as JSON). How it was encoded we know from index time and can have classes that are responsible for handling the document format do the right magic.
- Since we know the addresses in all our fields (paths) we can find the paths for any word.
- This lets us, for example, not just search for words in a specific field, a specific path but also for words in the same container without knowing its name or path.
- Since we can read the original file we can also search for unlimited length literals using any wildcard or regex scheme of our dreams—even applied to field names and paths.
- We can even go the other way and ask: what words are in a given field or path instance?
- Since the hierarchical structure of a document is modelled we can walk up (parent) and down (children) from any point in its tree without need to re-parse.
- Since we know all the words and the structure we can even at search reconstitute an alternative encoding without needed to re-parse the original. Its just load and translate using the extracted structure and addresses from the indexing stage.
- Throw in multiple polymorphic datatypes with their own search and ranking algorithms…

# Exploiting Structure
## (Explicit and Implicit)

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 vector datatypes tuned for embeddings and using structure not just for chunking but for a more optimal passage retrieval.**

# 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
    B. Chunk size and chunking strategy
    C. Robustness and scalability
2. Search and Retrieval
3. Data Security

# Ingest

## A) 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).

Our ODF parser is, for example, the absolute state-of-the-art: the fastest and most accurate (created in coordination with the OASIS ODF TC).

# Ingest

## B) Chunking

Our approach is variant of context-aware chunking that also takes the hierarchical structure of text into account. It contains a computationally more modest approach than “agenic” (which uses a LLM to determine chunks) to find better context matches by using the found passages and seeing them in the hierarchical context of the document. In contrast to recursive we view the edge nodes of a document tree as the fine level chunks and determine the optimal passage on the distribution and relation of the retrieved chunks in the tree.

# Ingest

## C) Robustness and Scalability

In order to be robust and scale one needs a modular and distributed system. The re-Isearch design has been proven over many years in a large number of large scale highly visible deployments.

Beyond the internal document formats (DOCTYPES) there is a mechanism for dynamic loadable plugins.

In the future a plug-in design will also be implemented to extend re-Isearch to support additional ANN (approximate nearest neighbour) algorithms for dense vector retrieval ranking as well as embedding generators.

The 64-bit index (32-bit file IDs) supports a max Input of 16384 Tbytes (Total of all files) and 1 million records per physical index which, in turn, can be aggregated on demand at search-time with up to 254 additional indexes to create a searchable virtual index (max 255 million records). This can be significantly increased by compiling for 64-bit file IDs. In the other direction it can be compiled to produce 32-bit indexes to save on storage.

# Search and Retrieval

## The power of re-Isearch as a hybrid

**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).

# Sustainability
## Energy and Computational resource demands.

**It's one thing to have a good search but it must also be economically feasible, efficient and sustainable.**

While the standard re-Isearch powered node can run on pretty much any platform, even small embedded systems, LLM inferencing (currently) demands a bit more resources. But instead of costly GPU data-servers (where popular the NVIDIA H100 costs $30k each—even the cut-price Asian market 96GB H20 variant costs $12k—and consumes alone 700w and are often deployed for inferencing in 8x constellations) we aim to target accessible platforms.  While companies such as Google sing praise about their massive context length (large context or LC) models they are heavily data-center based and energy intensive.  By constraining computational requirement we can leverage existing hardware and adhere to our aim to eliminate the need for costly data centers and reduce the environmental impact for search. Just as with the standard version, it is intended that users will be able to host their own data under their own well defined conditions.

We have opted for a flexible architectural design to support multiple vector stores including, of course, HNSW. The basic idea is to view them as data types and within their handlers pipelines to selectable/extendable embeddings models. We have found, for example, that some algorithms work for some data really well but for others sub-optimally and visa-versa. Another aspect of our design, however, is to enable a simple pipeline where documents handed over to the ingest get processed by the appropriate document handler and data types detected, structure exploited for chunks, embeddings created and stored in an appropriate vector data store.

# Development Organisation

**Under re-Isearch.org (Munich) in association with ExoDAO Network Association (Zurich)**

- re-Isearch.org is a newly formed organisation to centre the development of the re-Isearch engine. It is based in Munich, Germany.
- ExoDAO.net is a Swiss Association established in 2022 in Zurich, Switzerland around the Student Project House (SPH) of the Swiss Federal Institute of Technology in Zurich (ETH-Z). It has been tasked with a “moonshot” to develop a next generation Internet search. Its founding was catalysed by a grant from the Open Data Switzerland/Mercator Foundation Switzerland.
- Through the founding of re-Isearch.org the development of re-Isearch is disentangled from ExoDAO protocol development. It is intended as a clear signal that one can use re-Isearch without participating in the ExoDAO Network.

# Main Repository

https://github.com/re-lsearch/Schmate