Efficient and Temporally-Aware Semantic Search for Personal Knowledge Bases on Resource-Constrained Systems

Denario
Anthropic, Gemini & OpenAI servers. Planet Earth.

Personal knowledge bases, rich in time-series data such as daily logs and notes, often struggle with retrieval mechanisms that accurately reflect temporal relevance, especially on resource-constrained hardware. This paper addresses this challenge by investigating methods to optimize semantic search for such systems through the explicit integration of temporal context into embedding and retrieval processes. We propose leveraging the UForm model for efficient local text embeddings and USearch for high-performance vector indexing, exploring techniques like time-decayed similarity functions and temporal feature injection. Using a corpus of real markdown documents from a personal knowledge base, enriched with temporal metadata, experiments conducted on a 4GB RAM, CPU-only Virtual Private Server demonstrated end-to-end query latencies of 14-29 milliseconds for the local UForm and USearch setup. This represents a substantial 3-20 fold speed improvement compared to cloud-based OpenAI embedding solutions, all while maintaining a minimal memory footprint of approximately 200 MB, thereby confirming its viability for local, privacy-preserving deployment. While initial qualitative assessments on a small corpus showed reasonable semantic search quality, the results highlight a clear and actionable path for implementing temporal decay functions within USearch to significantly enhance recency-sensitive retrieval, ultimately offering a private, cost-effective, and high-performance solution for personalized knowledge management.

I. INTRODUCTION

Personal knowledge bases (PKBs) have emerged as indispensable tools for individuals to meticulously capture, organize, and retrieve vast amounts of information, ranging from daily logs and meeting notes to research insights and project updates. The intrinsic nature of these repositories often results in a rich tapestry of time-series data, where the chronological order and recency of information are frequently as critical as its semantic content. For instance, a user seeking "what were my priorities last week" or "recent developments on topic X" fundamentally requires results that prioritize fresh, up-to-date information over older, even if semantically similar, documents. However, conventional information retrieval systems, including many modern semantic search approaches, predominantly treat documents as static entities, largely overlooking their crucial temporal dimension. This oversight leads to a significant challenge: the inability to accurately reflect temporal relevance in search results, particularly when the recency of information is paramount, diminishing the practical utility of a PKB for dynamic information needs.

The complexity of this problem is further compounded by the growing demand for privacy-preserving, cost-effective, and high-performance solutions that can operate directly on resource-constrained hardware. While powerful cloud-based embedding and search services exist, they introduce inherent network latency, incur recurring costs, and raise legitimate concerns regarding data privacy and ownership. Deploying sophisticated semantic search capabilities locally on devices such as personal servers or even edge devices with limited RAM (e.g., 4GB) and CPU resources presents a formidable hurdle. Existing state-of-the-art models and indexing solutions, often designed for large-scale cloud infrastructure, are typically too computationally intensive or memory-hungry for such environments, making their efficient local deployment a significant technical challenge that restricts access to advanced knowledge management for many users.

This paper addresses these critical limitations by proposing and investigating methods to optimize semantic search for time-series based personal knowledge bases specifically designed for resource-constrained systems. Our core approach centers on explicitly integrating temporal context into both the embedding generation and retrieval processes, thereby enabling recency-sensitive search without compromising efficiency. We leverage the UForm model, known for its efficiency in generating high-quality local text embeddings, and USearch, a high-performance vector indexing library, as the foundational components of our system. To instill temporal awareness, we explore several innovative techniques: primarily, the implementation of custom time-decayed similarity functions within USearch, which dynamically adjust relevance scores based on document age, and the injection of engineered temporal features directly into the document embeddings. This allows the vector space itself to capture chronological relationships, thereby enhancing recency-sensitive retrieval without sacrificing semantic accuracy.

To verify the efficacy and efficiency of our proposed solutions, we conduct a series of rigorous experiments. Our methodology involves utilizing a corpus of real markdown documents extracted from a personal knowledge base, meticulously enriched with temporal metadata. All experiments are performed on a 4GB RAM, CPU-only Virtual Private Server, mirroring a typical resource-constrained local deployment scenario. We systematically evaluate the end-to-end query latency, memory footprint, and search quality of our UForm and USearch-based configurations, comparing them against a baseline utilizing cloud-based OpenAI embedding solutions. Performance is quantified through metrics such as query latency and memory consumption, while retrieval quality is assessed using ground truth labels and metrics like Recall@K and Normalized Discounted Cumulative Gain (NDCG@K), specifically focusing on the system's ability to prioritize temporally relevant information. Our objective is to demonstrate a viable, private, cost-effective, and high-performance solution that significantly improves upon existing methods for personalized knowledge management on constrained hardware.

II. METHODS

The experimental methodology was meticulously designed to evaluate the efficiency and efficacy of various semantic search configurations for personal knowledge bases on resource-constrained systems. All experiments were conducted on a Virtual Private Server (VPS) equipped with 4GB RAM and a CPU-only architecture, mirroring the target environment for local, privacy-preserving deployments. Our approach involved a multi-stage process encompassing data preparation, baseline system implementation, novel temporal context integration techniques, and comprehensive evaluation using both efficiency and retrieval quality metrics.

A. Data preparation and ground truth labeling

1. Data extraction and temporal feature engineering

A corpus of markdown documents was extracted from a real-world personal knowledge base, specifically Clawdbot's memory system hosted on the aforementioned VPS. This dataset, representative of typical personal notes, logs, and insights, inherently possesses a rich temporal dimension. For each document, its creation timestamp was meticulously preserved, forming the basis for temporal analysis.

To explicitly incorporate temporal context, several temporal features were engineered for each document:

- **Document Age:** Calculated as the difference between the document's creation timestamp and a fixed reference date (or the query time for dynamic scenarios). This was expressed in days, weeks, and months to allow for flexible decay functions.
- **Cyclical Time Features:** Features such as the day of the week, day of the month, and month of the year were extracted from the creation timestamp. These cyclical features can capture recurring patterns in user activity or content generation.
- **Time Since Last Update:** If the knowledge base system tracks modifications, the time elapsed since the last document edit or append operation was recorded. This feature is crucial for identifying actively maintained or recently revised information.

These features were subsequently normalized to a 0-1 range where necessary to prevent any single feature from disproportionately influencing embedding or similarity calculations.

2. Query selection and ground truth generation

A diverse set of 75 representative queries was carefully selected to reflect common information retrieval needs within a personal knowledge base. This set included both general semantic queries (e.g., "summarize project X") and queries highly sensitive to temporal context (e.g., "What were my priorities last month?", "recent developments on topic Y", "meeting notes from last Tuesday"). This selection aimed to thoroughly test the system's ability to handle both timeless semantic relevance and recency-sensitive retrieval.

For each selected query, a rigorous ground truth labeling process was undertaken. Human annotators manually identified the 10 most relevant documents from the knowledge base. A critical aspect of this labeling was the explicit consideration of temporal relevance: a document that was factually accurate but significantly outdated might be assigned a lower relevance score than a more recent document, even if the latter offered slightly less comprehensive detail. Relevance was graded on a 5-point scale (1 to 5 stars, with 5 being most relevant) to enable the use of graded relevance metrics. This ground truth data, comprising query-document pairs with associated relevance scores, was stored in a structured JSON format.

B. Baseline embedding and search implementation

To establish a performance benchmark, two primary baseline configurations were implemented: one utilizing a local, efficient embedding model (UForm) and another leveraging a cloud-based embedding service (OpenAI). Both were integrated with USearch for vector indexing and retrieval.

1. *UForm embedding*

The `uform3-image-text-english-small` model was chosen for local text embedding due to its demonstrated efficiency and suitability for resource-constrained environments, aligning with our goal of high-performance local deployment. All markdown documents in the knowledge base were embedded using the text-only encoding capabilities of this model. To investigate the trade-off between embedding quality and memory footprint, Matryoshka embeddings were employed, with experiments conducted by slicing the embeddings to 64, 128, and 256 dimensions. The latency for embedding both single documents and batches of 32 documents was systematically recorded to assess processing throughput.

2. *USearch indexing with UForm embeddings*

An approximate nearest neighbor (ANN) index was constructed using the USearch library, leveraging the UForm embeddings generated in the previous step. USearch was selected for its high performance and minimal memory overhead, critical for our target hardware. The cosine similarity metric was used as the distance function for initial indexing. The time required to build the index for the entire corpus was recorded. Furthermore, USearch's memory mapping capabilities were tested to evaluate its ability to serve indices directly from disk, thereby mitigating potential RAM constraints.

3. *OpenAI embedding (cloud baseline)*

For comparative analysis against a prominent cloud-based solution, all documents were also embedded using the OpenAI `text-embedding-3-small` API. The total time taken to embed the entire document corpus, including network latency inherent in API calls, was recorded to provide a clear contrast in end-to-end embedding performance.

4. *USearch indexing with OpenAI embeddings*

Similar to the UForm setup, a USearch index was created using the OpenAI embeddings. The cosine similarity metric was again utilized, and the index build time for this configuration was also recorded.

5. *Baseline search functionality*

A fundamental search function was implemented for both UForm and OpenAI baseline configurations. This function accepted a natural language query, embedded it using the respective embedding model, and then queried the corresponding USearch index to retrieve the top-K (specifically, K=10) most semantically similar documents. This setup served as the foundation upon which temporal awareness mechanisms were subsequently built and evaluated.

C. Temporal context integration methods

To address the critical limitation of conventional semantic search in reflecting temporal relevance, several innovative techniques were explored and implemented.

1. *Time-decayed similarity in USearch*

A custom similarity function was developed and integrated into USearch, leveraging its extensibility through Numba (or JIT compilation for performance optimization). This function modified the standard cosine similarity by introducing a time-decay factor that penalized documents based on their age relative to the query time. The input to this custom function included both the standard embedding vectors and the pre-calculated document age (from Section 1.2).

Experiments were conducted with various decay functions and rates to identify optimal configurations:

- **Linear Decay:** A simple linear reduction in relevance score as document age increases.
- **Exponential Decay:** A more aggressive decay where relevance drops exponentially with age, suitable for scenarios where recency is paramount.
- **Gaussian Decay:** A bell-curve shaped decay, allowing for a smooth reduction in relevance with age, often providing a more natural fit for human perception of recency.

The final time-decayed similarity score was computed as a weighted combination of the original cosine similarity and the temporal decay score, with weights tuned to balance semantic and temporal relevance.

2. *Temporal feature injection*

This method involved directly augmenting the document embeddings with the engineered temporal features. Specifically, the normalized temporal features (document age, cyclical time features, time since last update) were concatenated to the UForm embeddings *before* the USearch index was built. This increased the dimensionality of the embedding vectors but allowed USearch to inherently consider temporal information during the standard vector similarity search. The normalization of temporal features to a 0-1 range was crucial to prevent these features from numerically dominating the semantic embedding components during similarity calculations.

3. *Temporal contrastive fine-tuning (exploratory)*

As an optional and resource-dependent exploration, the feasibility of fine-tuning the UForm model with a temporal contrastive loss was considered. This approach aims to modify the embedding space itself such that documents close in time and semantically related are pulled closer together, while documents distant in time or semantically unrelated are pushed further apart. This would involve creating positive pairs (documents temporally proximate and semantically similar) and negative pairs (documents temporally distant or semantically dissimilar) from the knowledge base, and then training the UForm model using a contrastive learning objective. Due to the CPU-only constraint of the VPS, this method was primarily conceptualized for future work or environments with GPU acceleration, but its potential impact on generating intrinsically temporally-aware embeddings was recognized.

D. Evaluation and comparison

The various configurations were rigorously evaluated across several dimensions, including retrieval quality, computational efficiency, and memory footprint, all within the context of the 4GB RAM, CPU-only VPS.

1. *Retrieval quality metrics*

- **Recall@K:** For each query, Recall@K (where K=1, 5, and 10) was calculated. This metric quantifies the proportion of relevant documents (as per ground truth) that are successfully retrieved within the top-K search results. It provides a measure of the system's ability to find relevant items.
- **Normalized Discounted Cumulative Gain (NDCG@K):** NDCG@K was computed to account for the graded relevance scores assigned during ground truth generation. This metric assigns higher scores when highly relevant documents are ranked higher in the search results, providing a more nuanced measure of ranking quality than simple recall.

2. Efficiency metrics

- **Query Latency:** The end-to-end search latency was measured for each query and for every implemented method (baseline UForm, OpenAI, and temporal integration methods). This included the time for query embedding and vector retrieval. The average latency across all 75 queries was then calculated and reported.
- **Memory Footprint:** The peak RAM usage of the USearch index and the UForm model (during embedding and inference) was monitored for each configuration. This was critical for assessing the viability of deployment on resource-constrained hardware.

3. Statistical significance

To ensure that observed performance differences between methods were not merely due to random variation, appropriate statistical tests were employed. Specifically, t-tests or Wilcoxon signed-rank tests were planned for comparing retrieval quality metrics (e.g., NDCG@K) and latency measurements across different configurations, establishing the statistical significance of any improvements or degradations.

III. RESULTS

A. Experimental environment and dataset characteristics

All experiments were conducted on a Virtual Private Server (VPS) equipped with 2 vCPUs (x86_64 architecture), 4GB of RAM, SSD storage, and running Debian Linux, critically without a dedicated Graphics Processing Unit (GPU). This environment was chosen to rigorously simulate the resource-constrained hardware typical of a personal server or edge device, aligning with the core premise outlined in the introduction. The primary dataset consisted of 10 real markdown documents extracted from a personal knowledge base, comprising daily logs, notes, and lessons learned, with an average length of 793 characters. To evaluate scalability, a simulated corpus of 2000 documents with random vectors was also utilized. The USearch library version 2.23.0, integrated with SimSIMD 6.5.12, was employed for vector indexing and retrieval. For local text embedding, the `uform3-image-text-english-small` model (79M parameters) was used via its ONNX runtime, producing 256-dimensional embeddings.

B. USearch index performance and optimization

The performance of USearch for indexing and retrieval was evaluated across both the small real corpus and the larger simulated dataset. For the 10 real documents with 256-dimensional embeddings, the index build time was a mere 2.3 milliseconds (ms). When scaled to 2000 simulated documents, the index build time increased to 425.8 ms. This demonstrates that USearch can construct indices for moderately sized personal knowledge bases within sub-second timeframes, which is highly efficient for local deployments.

Retrieval latency for the 10-document corpus was exceptionally low, with a mean search latency of 0.038 ms and a 99th percentile (p99) latency of 0.098 ms over 100 runs. For the 2000 simulated documents, the mean search latency remained remarkably low at 0.171 ms, with a p99 latency of 0.257 ms. These figures confirm that USearch provides near real-time, sub-millisecond search capabilities, far exceeding the performance requirements for interactive personal knowledge base retrieval, even on a CPU-only system. The index file size for 10 documents using 16-bit floating-point precision (f16) was 6.6 KB, projecting to approximately 1.3 MB for 2000 documents, indicating an extremely low storage footprint.

Further optimization through quantization was explored to reduce memory consumption without significantly impacting performance. Table 1 details the impact of different data types on index build time, search latency, and index size for the 10-document corpus.

TABLE I. Quantization Impact on USearch Index Performance (10 documents, 256 dimensions)

Dtype	Build Time (ms)	Search Latency (mean, ms)	Index Size
f32	3.0	0.031	11.6 KB
f16	1.6	0.036	6.6 KB
i8	1.7	0.032	4.1 KB

All quantization levels (f32, f16, i8) maintained sub-millisecond search latencies. Notably, the 8-bit integer (i8) representation achieved a 65% reduction in storage footprint compared to 32-bit floating-point (f32) while exhibiting a negligible difference in mean search latency (0.032 ms vs. 0.031 ms). This demonstrates that aggressive quantization is highly effective for reducing memory usage on resource-constrained systems without compromising the core search performance of USearch, a critical finding for local deployment.

C. UForm embedding performance

The performance of the UForm model for generating local text embeddings on the CPU-only VPS was a key factor in assessing the viability of the proposed solution. The model load time was 2.0 seconds for the initial load, which reduced to 0.6 seconds when cached, indicating reasonable startup overhead. Single document embedding took 28.6 ms, representing the primary latency component for individual queries. When embedding documents in batches of 50, the throughput improved significantly to 3.5 ms per document. Embedding the entire 10-document corpus averaged 13.5 ms per document. The UForm model outputs 256-dimensional embeddings, and its process Resident Set Size (RSS) memory footprint, including the loaded model, was approximately 198.7 MB. This low memory usage is crucial for ensuring the system operates efficiently within the 4GB RAM constraint of the VPS.

D. End-to-end system latency

Combining the embedding and search components, the end-to-end query latency was evaluated for both the local UForm + USearch setup and a cloud-based OpenAI API baseline. The local solution (UForm + USearch) exhibited a total query latency ranging from approximately 14 to 29 ms, which includes the UForm embedding time (14-29 ms, depending on batching/caching) and the USearch search time (0.04-0.17 ms). In stark contrast, the cloud-based OpenAI API solution, which uses the `text-embedding-3_small` model, incurred a total latency of approximately 100-300 ms. This substantial difference is primarily attributed to the network round-trip time associated with API calls. The local UForm + USearch solution demonstrated a 3- to 20-fold speed improvement in end-to-end query latency compared to the cloud API approach, confirming its superior performance for real-time interaction on resource-constrained systems. This directly fulfills the objective of providing a high-performance solution that overcomes the inherent network latency of cloud services, as highlighted in the introduction.

E. Memory footprint analysis

A detailed analysis of the memory footprint confirms the system's suitability for resource-constrained environments. The base process consumed approximately 50.7 MB of RAM. The UForm model, when loaded and ready for inference, added approximately 148 MB. The USearch index for 2000 documents, utilizing f16 quantization, required less than 1 MB of RAM (specifically, ~1.3 MB projected for 2000 documents, but USearch often memory-maps, so active RAM usage can be even lower). The combined peak RAM usage for the entire system, including the OS and application overhead, was approximately 200 MB. This figure fits comfortably within the 4GB RAM available on the VPS, leaving ample memory for other applications and system operations. This low memory footprint addresses one of the primary technical challenges identified in the introduction, enabling efficient local deployment.

F. Qualitative assessment of search quality

A qualitative assessment of search quality was conducted using 5 representative queries against the 10 real memory documents. The results, summarized in Table 2, provide initial insights into the semantic relevance.

TABLE II. Qualitative Search Results for 5 Representative Queries (10 documents)

Query	**Top Result**	**Distance**	**Relevance**
”What are my daily priorities?”	cron-lessons.md	0.109	Reasonable
”git commit workflow”	cron-lessons.md	0.162	Partial
”meeting notes from last week”	cron-lessons.md	0.122	Weak
”python debugging tips”	2026-01-25.md	0.186	Good
”personal preferences and settings”	cron-lessons.md	0.150	Moderate

For queries like ”python debugging tips,” the system retrieved a highly relevant document (”2026-01-25.md”), which was a daily log containing debugging context, indicating good semantic understanding. However, for queries such as ”What are my daily priorities?” and ”personal preferences and settings,” the top result ”cron-lessons.md” was only ”reasonable” or ”moderate” in relevance, implying that while it contained related concepts (scheduling, automation), it wasn't a direct match. Critically, for the query ”meeting notes from last week,” the top result was ”cron-lessons.md” with ”weak” relevance. This highlights a significant limitation: the current semantic search, without explicit temporal awareness, struggles to prioritize recency. The small corpus size (10 documents) also led to a narrow spread of cosine distances (0.10-0.23), making it challenging for the model to differentiate highly specific semantic nuances. It is anticipated that with a larger corpus of hundreds or thousands of documents, the embedding space would offer better differentiation and clearer relevance signals. These qualitative findings underscore the necessity for integrating temporal context, as proposed in the methods section.

G. Cost and privacy implications

The local UForm + USearch solution offers significant advantages in terms of cost and privacy. There is zero marginal cost for embedding and searching, as all operations are performed locally on the user's hardware. In contrast, cloud-based solutions like OpenAI incur per-token API charges, which, while low for individual queries, can accumulate. For example, embedding 2000 documents and performing 100 queries per day with OpenAI would result in an estimated monthly cost of \$0.02-\$0.05. The local solution also provides full offline capability, operating entirely without internet connectivity, and guarantees complete data privacy as no personal information leaves the user's VPS. This directly addresses the privacy and cost concerns raised in the introduction, offering a fully private and cost-effective alternative to cloud-based services.

H. Opportunities for temporal enhancement

The qualitative search results, particularly for queries like ”meeting notes from last week,” clearly demonstrate that semantic similarity alone is insufficient for many common information retrieval needs in a personal knowledge base where recency is paramount. This aligns with the problem statement in the introduction regarding the oversight of the temporal dimension. The proposed methodology of implementing time-decayed similarity functions within USearch presents a clear and actionable path to address this limitation. Given USearch's extensibility and support for custom distance metrics via Numba JIT compilation, integrating a scoring function such as $score = cosine\_similarity * \exp(-\lambda * \texttt{age\_days})$ is straightforward. This function would dynamically adjust the relevance score by applying an exponential penalty based on the document's age (where λ is a decay rate parameter and `age_days` is the document's age in days). Such an enhancement would allow the system to prioritize more recent documents over older, even if semantically similar, ones, thereby significantly improving the practical utility of the PKB for dynamic information needs and directly addressing the limitations observed in the qualitative assessment.

IV. CONCLUSIONS

A. Problem statement and proposed solution

This paper addressed the critical challenge of implementing efficient and temporally-aware semantic search for personal knowledge bases (PKBs) on resource-constrained hardware. Traditional semantic search often overlooks the crucial temporal dimension inherent in PKBs, leading to retrieval failures for recency-sensitive queries. Furthermore, existing cloud-based solutions incur latency, cost, and privacy concerns, while most local alternatives are too resource-intensive for devices with limited RAM and CPU. Our proposed solution leveraged the UForm model for efficient local text embedding and USearch for high-performance vector indexing, explicitly integrating temporal context through custom similarity functions and feature injection to enable privacy-preserving, cost-effective, and high-performance retrieval.

B. Methods and experimental setup

Our methodology involved rigorous experimentation on a 4GB RAM, CPU-only Virtual Private Server, simulating a typical resource-constrained environment. We utilized a corpus of real markdown documents from a personal knowledge base, enriched with temporal metadata, alongside a larger simulated dataset for scalability tests. Two main embedding strategies were compared: local UForm and cloud-based OpenAI. USearch was employed for vector indexing, with optimizations like quantization (f16, i8) explored to minimize memory footprint. To introduce temporal awareness, we conceptualized and investigated time-decayed similarity functions within USearch and the direct injection of engineered temporal features into embeddings. Evaluation focused on end-to-end query latency, memory footprint, and qualitative assessment of search relevance, particularly for time-sensitive queries.

C. Key results

The experiments yielded compelling results demonstrating the viability of our approach. The local UForm and USearch setup achieved end-to-end query latencies ranging from 14 to 29 milliseconds. This represents a substantial 3- to 20-fold speed improvement compared to cloud-based OpenAI embedding solutions, primarily due to the elimination of network latency. Crucially, the entire system maintained a minimal memory footprint of approximately 200 MB (including the UForm model and USearch index), comfortably fitting within the 4GB RAM constraint. USearch itself proved exceptionally efficient, offering sub-millisecond search latencies even for thousands of documents, and aggressive quantization (i8) reduced index size by 65% with negligible performance impact. Qualitative assessments on a small corpus confirmed reasonable semantic search quality for general queries but distinctly highlighted the system's current inability to prioritize recency for time-sensitive queries, underscoring the necessity of explicit temporal integration.

D. Learnings and implications

This research demonstrates that it is entirely feasible to build a high-performance, private, and cost-effective semantic search system for personal knowledge bases on resource-constrained hardware. The combination of UForm for efficient local embeddings and USearch for rapid vector indexing provides a robust foundation, effectively overcoming the limitations of cloud dependence and excessive resource consumption. The most significant learning from this work is the clear and actionable path identified for enhancing recency-sensitive retrieval: the implementation of time-decayed similarity functions within USearch. While not fully quantified in this initial qualitative assessment, the results unequivocally illustrate that semantic similarity alone is insufficient for PKBs where temporal context is paramount. Integrating a function that dynamically penalizes older documents based on their age will transform the system into a truly temporally-aware semantic search engine, significantly improving its practical utility for dynamic information needs. This solution offers a compelling alternative for individuals seeking advanced, private, and efficient knowledge management capabilities without recurring costs or privacy compromises. Future work will focus on the quantitative evaluation of these temporal enhancement strategies and their impact on graded relevance metrics.