MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Personalized Similarity Search for Vector Databases

Advanced Master's Thesis

MGR. ET MGR. MATÚŠ ŠIKYŇA

Brno, Fall 2025

MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Personalized Similarity Search for Vector Databases

Advanced Master's Thesis

MGR. ET MGR. MATÚŠ ŠIKYŇA

Advisor: prof. Ing. Pavel Zezula, CSc.

Department of Machine Learning and Data Processing

Brno, Fall 2025

Signature of Thesis Advisor


FACULTAS ARTIS INFORMATICÆ
UNIVERSITAS MASARYKIANA

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source. During the preparation of this thesis, I used ChatGPT and Grammarly to improve my writing style. I declare that I used the tool in accordance with the principles of academic integrity. I checked the content and take full responsibility for it.

Mgr. et Mgr. Matúš Šikyňa

Advisor: prof. Ing. Pavel Zezula, CSc.

Abstract

Vector databases store large collections of high-dimensional embeddings and make them searchable by building an index around a fixed similarity function, most commonly Euclidean distance. This design enables efficient retrieval at scale, but it also hard-codes a single notion of similarity for all users, even though human similarity judgements are subjective and context-dependent. Changing the similarity model typically implies rebuilding the index, which is impractical when many users share the same collection but require different views of relevance. We address this gap by introducing a similarity search engine that maintains a shared Euclidean index while allowing each user to search through a personalized similarity view, expressed as a learned Mahalanobis metric parameterized by a small matrix. We validate the proposed theory experimentally and show that semantic relevance feedback from user interactions can effectively learn and update user matrices. To translate the relevance–efficiency trade-off into practical deployment guidance, we define operational scenarios that balance personalization quality against computational cost, ranging from configurations that prioritize efficiency and stable rankings to those that maximize retrieval effectiveness. Across multiple feedback-processing strategies and a broad suite of metric learning models, we achieve high personalization gains, increasing the mean average precision by up to 0.211 while keeping scaling factors low, demonstrating that effective and efficient personalized similarity search is possible without modifying the shared index.

Keywords

metric learning, similarity search, vector embeddings, learning user profiles, relevance feedback, vector databases, personalized search

Contents

List of Tables

List of Figures

1 Introduction

Recent advancements in artificial intelligence, particularly in representation learning, have produced models that map text, images, and other modalities into high-dimensional vector embeddings. These embeddings encode rich semantic information, enabling efficient similarity-based retrieval, and are employed across various applications, such as recommender systems [1], clustering [2], classification [3], and information retrieval [4].

To operate with such representations at scale, vector databases are used to store and manage embeddings, enabling efficient similarity search over billions of high-dimensional vectors. Such databases are now commonly used as the retrieval layer in semantic search, recommendation, and retrieval-augmented generation pipelines, where relevant items or documents are retrieved based on similarity before further processing. To achieve low latency at scale, they commonly employ approximate nearest-neighbour (ANN) indexing techniques that trade a small amount of recall for substantial speedups. This efficiency comes from precomputing index structures over the vectors, so the ANN index acts as a candidate generator, narrowing the search space before the final similarity evaluation. Vector databases, therefore, expose similarity search as their core query operation. The quality of the returned results depends not only on the embedding model, but also on how the index generates and orders candidates under the chosen similarity function. In practice, vector databases commonly rely on a fixed distance function (e.g., Euclidean distance) to organize the index and to select the most relevant vectors with respect to a given query or reference vector. However, once the content is embedded and a single distance function is utilized, the candidate selection behaviour is identical across all users, while changing this notion of similarity generally requires rebuilding or substantially reconfiguring the index. This is in sharp contrast with human perception of similarity, which is subjective and context-dependent [5].

This observation connects the indexing issue with a broader problem. In principle, there is a gap between the *global relevance* suggested by general-purpose embeddings and the *personal relevance* experienced by an individual user. What is relevant is not only a property of the

content itself, but also of the user, the current intent, and the situation in which the request is made. A practical answer to this gap is context-aware computing, where the same query can legitimately produce different results for different users, even when operating over the same underlying database.

Such motivation has been visible for a long time in large-scale systems. Web search moved beyond purely universal ranking early on, as users typing identical queries often meant different things and valued different sources. Therefore, industrial systems introduced personalization signals derived from user interactions to adjust rankings accordingly. In a different but closely related setting, services such as Netflix built their user experience around personalization from the start: the catalogue is the same, but what is shown, in what order, and in what context is tailored to the individual. Similar motivations apply in advertising and e-commerce, where even minor improvements in personal relevancy can translate into substantial gains. Because such techniques have commercial potential, the precise system designs and evaluation procedures behind deployed solutions are typically proprietary.

As a result, the research community has introduced a broad spectrum of personalized search methods that exploit behavioural and contextual evidence, including query history, clicks, dwell time, location, and social or topical profiles, to rerank results for each user. The proposed techniques range from probabilistic models and collaborative filtering to deep neural architectures designed to model both short-term and long-term interests together with situational context [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]. Later, representation-based approaches moved this idea into vector spaces by learning neural embeddings for users, queries, items, and entities, so that similarity in the learned space directly encodes user-specific preferences for search or recommendation [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]. More recently, personalization has been extended with knowledge-graph signals, self-supervised objectives, and LLM-based components, including editable or entity-centric memories, motivation-aware embeddings, and multimodal user representations that provide finer-grained and more context-sensitive signals [34, 35, 36, 37]. Despite this progress, embedding-based retrieval systems that rely on a sin-

gle shared index still face the practical constraint that the candidate generation stage is tied to a fixed metric.

In this work, we focus on vector databases and bridge the gap between the shared notion of similarity encoded by general-purpose embeddings stored using the fixed index and the personalized notion of similarity that individual users apply during similarity search. Specifically, we elaborate on an architecture in which all vectors are stored in a single shared index built for a standard metric (Euclidean), while each user queries the same database using a personalized similarity model expressed as a Mahalanobis metric parameterized by a learned matrix. We learn this matrix from user relevance feedback, framing personalization as a metric learning problem [38] in which the geometry of the embedding space is adapted to reflect an individual's notion of similarity. More generally, metric learning uses supervision, such as pairwise or triplet preferences, to learn a distance function that brings relevant items closer together while pushing irrelevant items further apart. Furthermore, our design exploits the lower bounding relationship [39] between the Euclidean and Mahalanobis distances. This relationship provides a formal guarantee that Euclidean-based candidate generation returns a superset of the true Mahalanobis neighbours, enabling personalization in a refinement stage without rebuilding the shared index. To implement a practical *filter and refine* search, the Euclidean index efficiently retrieves a guaranteed superset of candidates, which is then refined using the user-specific Mahalanobis metric.

The approach is investigated through two studies. Our first study formulates relevance feedback as supervision for metric learning, proposes a search engine architecture, and validates the proposed approach using *synthetic* feedback signals. Our second study focuses on *semantic* relevance feedback collected from real user interactions with the returned results, and uses it to learn and update the personalized similarity models in practice. We propose and compare several feedback processing strategies for constructing feedback judgements (in the form of triplets), evaluate them across a range of established metric learning models, and suggest practical system setup strategies under different operational priorities (e.g., trading relevance gains against computational cost, or prioritizing efficiency vs. precision).

The work is organized as follows: Chapter 2 reviews personalized search approaches, ranging from traditional feature-based methods that leverage behavioural and contextual signals to representation-based techniques using embeddings and large language models in vector spaces. Chapter 3 summarizes the achieved results from the published papers. Appendix A and Appendix B contain the full versions of our publications.

2 State of the Art

Personalized search utilizes signals from a user's prior interactions to infer their interests and tailor rankings, ensuring results better fit their current information needs. This chapter surveys two broad families of methods. First, we review *traditional personalization techniques* that rely on engineered behavioural and contextual signals (including queries, clicks, dwell time, time, and location) and incorporate them through relevance feedback, user profiling, topical taxonomies, collaborative signals, and learning-to-rank or neural models. Second, we cover *embedding-based approaches* that represent users, queries, and items in shared vector spaces, enabling personalization through representation learning and similarity-based retrieval, as well as modern transformer and LLM-based encoders with explicit user memory. Finally, we position our approach relative to these lines of work.

2.1 Traditional Personalized Search Approaches

Traditional personalized search methods typically break down the problem into: extracting behavioural and contextual evidence, constructing a user model (e.g., profile, topic distribution, or similarity function), and incorporating that model into ranking through re-ranking, query expansion, or feature augmentation. The literature encompasses (i) interactive relevance feedback loops that update user profiles online, (ii) implicit feedback profiling from queries and clicks, and (iii) ontology-based approaches (e.g. using ODP) that extend personalization to the topic level. To address sparse or noisy individual signals, (iv) group and collaborative methods smooth user models using related users. Furthermore, (v) context-aware and time-aware models capture situational intent and temporal drift across sessions. In the following subsections, we describe these lines of work and highlight representative methods within each category.

2.1.1 Interactive Relevance Feedback and Similarity Learning

Early personalization systems treat personalization as an interactive loop. The system observes feedback (explicit or implicit), updates

a user model online, and decides what to show next. This family includes relevance-feedback-style Bayesian updating and online similarity/metric learning.

PicHunter [6], a content-based image retrieval system was introduced, using a Bayesian relevance feedback loop: after each display, it updates a posterior probability over all images from user actions (e.g., selecting relevant images) using a user model derived from similarity judgment experiments, and selects the next display via an entropy-minimizing strategy, leveraging hidden semantic annotation and pictorial feature vectors (e.g., color). The limitation of the system is that it scales linearly with database size (feature storage and repeated user model evaluations). Therefore, it is not suited for large-scale collections (e.g., hundreds of millions to billions of images).

Chechik et al. [40] introduced OASIS, an online method that learns a bilinear similarity over sparse image features from relative similarity constraints (e.g., images sharing a label or retrieved for the same text query), using a large margin criterion and an efficient hinge loss cost, scaling linearly to millions of images. The learned model is class-independent. Therefore, it is not aware of which queries or categories were shared by two similar images.

2.1.2 Implicit Feedback Profiling

A large body of work personalizes search by building profiles from implicit behavioural signals (e.g., queries, clicks, visited pages, and query reformulations) and uses the resulting profile for re-ranking or for generating query suggestions.

Sugiyama et al. [7] built per-user profiles from user browsing histories (in one day). A limitation of the approach is its dependence on a sufficient and reliable history, as well as sensitivity to the window size (in days) parameters.

Shen et al. [8] proposed a client-side agent, UCAIR, that performs eager implicit feedback, using the immediately preceding query and clickthrough information for query expansion to improve the accuracy of ad-hoc retrieval.

Teevan et al. [9] constructed user profiles implicitly based on past queries, visited pages, and even local documents and emails, treating them as a form of relevance feedback for reranking the results. A

limitation of the approach is that personalization gains are highly parameter- and query-dependent, so it likely requires tuning per query or user and can yield volatile results as profiles and contexts change.

Another work [41] implicitly learned past and current topical preferences from clickthrough logs (queries and visited pages) using a topical ontology. The approach assigns topics to previously visited pages, then updates the user profile by relating the current query to the user's past topic preferences. Search results are then re-ranked using a scoring function that favours pages whose inferred topics best match the topics in the user's profile. A limitation is that the effectiveness of the approach depends on the accuracy of topic classification.

Furthermore, Vu et al. [42] personalized query suggestions for each search session by building two temporal user profiles: one from clicked documents and one from the user's query modification history. A learning-to-rank model (LambdaRank [43]) is used to re-rank the query suggestion list.

Bennett et al. [13] studied how short-term (session) and long-term (historic) behaviour interact in personalization using large-scale query and click logs, and show that historic signals help most at the start of a session, while session signals contribute most of the gains later in longer sessions. They further find that combining historic and session evidence outperforms either alone.

2.1.3 ODP-based Approaches

ODP-based approaches personalize search by mapping user interests onto a fixed topical directory (the Open Directory Project). Personalization then becomes topic-level matching by inferring which ODP categories a user prefers and re-rank results using topic-biased scores.

Qiu and Cho [11] inferred a user's preference from past click history and personalized ranking by selecting Topic-Sensitive PageRank scores over ODP topics for the user and query. Extending the approach to richer signals raises challenges in integrating heterogeneous sources.

Another work [10] built implicit profiles from search-engine side activity (via a Google wrapper), using queries that led to clicks and the clicked results' snippets (titles and summaries), mapping this per-user information to ODP concepts, and re-ranking results accordingly.

However, the approach is constrained to what the site can observe (queries and snippets for clicked items) and depends on the tuning and dynamicality of the concept hierarchy.

Zhongming et al. [44] proposed a client-side personalized categorization system, PCAT, that maps known user interests (e.g., workplace skills) to ODP categories, trains ODP-based text classifiers, and then categorizes search results by those interests on top of a standard engine (e.g., Google). A limitation of the approach is that it assumes reliable, pre-known interest data and accurate mappings into the ODP taxonomy. In practice, users may have unknown or shifting interests, or issue out-of-profile queries (e.g., unrelated to work), so many tasks may not align with any known interest, and personalization cannot help beyond what the system already knows.

Sieg et al. [45] built ontological user profiles by assigning and updating interest scores on concepts in a domain ontology (ODP), using a spreading activation algorithm driven by the user's ongoing implicit behaviour, and then re-rank results using these interest scores. A limitation authors highlight is the need to understand the stability and convergence properties of the profiles to represent user interests reliably.

2.1.4 Group and Collaborative Approaches

When individual histories are sparse or noisy, personalization can be improved by leveraging signals from other individuals, such as similar users, participation in groups, social connections, or crowds performing similar tasks. These methods cluster users, smooth individual profiles with group behaviour, and improve cold-start robustness.

Some works constructed user profiles based on collaborative filtering. Xue et al. [46] modelled personalization with a user language model that combines individual profiles, based on queries and visited page content, with group and global behaviour. Users are clustered into groups, and group models are built. Relevance is computed via a two-step smoothing process: global smoothing for unseen terms, followed by group-based interpolation. However, the effectiveness of the approach depends on the quality of the clustering and the tuned parameters.

Carmel et al. [47] personalized search by re-ranking results using signals from a user's social network, including familiarity-based and similarity-based relations inferred from social activity (e.g., tagging, commenting) and boosting documents by their relationship strength to the user's related people.

Teevan et al. [48] studied groupization, combining an individual's data with that of related users, analyzing similarity in query selections, desktop content, and explicit relevance judgments across people grouped in different ways. They found that the approach is most effective for explicit groups (e.g., task, occupation, or interest-based) on queries related to the group's focus, while implicitly inferred groups and off-theme queries are less cohesive and yield fewer reliable gains.

White et al. [49] moved beyond query matching by mining search logs to build task models, finding other users performing similar tasks, and promoting URLs based on on-task click behaviour. The authors note that the approach requires accurate modelling of search tasks, and for further gains, likely needs richer task features beyond queries or clicks (e.g., success signals or post-click behaviour), and potentially deeper re-ranking results when relevant URLs are not already near the top.

Vu et al. [15] proposed query-dependent dynamic grouping: given an input query, they identify a group of users with similar interests for that query, enrich the user profile (based on LDA [50]) using those users' signals, and re-rank results accordingly.

Furthermore, PerSaDoR [16] constructs a personalized social document representation for each user–document pair from social-tagging (folksonomy) annotations using matrix factorization, and combines this personalized representation with the document's textual ranking score to rerank results. The authors report significant performance gains with query-time costs that scale linearly with the number of matched documents.

2.1.5 Context, Engagement, and Temporal Dynamics

Beyond *what the user likes*, personalization also depends on situational and temporal factors: where the user is, what device they use, how engaged they are with items, and how intent drifts over time. These works incorporate context features (e.g., location, time, or device),

engagement measures (e.g., dwell time), and temporal modelling assumptions.

Lu et al. [12] focused on implicit local-intent queries, using users' physical location to personalize results.

Furthermore, Yi et al. [14] used item-level dwell time to measure the time of user engagement on a specific item. The primary practical challenges of this approach lie in measuring and normalizing dwell time robustly and in designing engagement objectives that more effectively capture long-term user satisfaction.

Zamani et al. [17] shifted context modelling from history-based signals to situational context (e.g., time, location, device) that is independent of query content and user history, proposing neural context-aware ranking models that learn dense representations from sparse contextual features. They evaluated the approach on a large personal search engine, where their model significantly improved the personalization.

Ge et al. [18] exploited sequential structure in user logs using a hierarchical Recurrent Neural Network (RNN). A lower-level RNN models current session (short-term behaviour), a higher-level RNN models dependencies across sessions (long-term behaviour), and a query-aware attention mechanism dynamically weights past sessions to form a query-specific user profile before re-ranking. They observed significant gains over traditional personalization approaches and showed that attention is able to highlight the important parts from previous queries and sessions.

Ma et al. [20] argued that sequential personalization models overlook fine-grained time information associated with user actions, and introduced a new model, PSTIE, to incorporate temporal information. The time-aware LSTM model captures the short-term evolution of query intent and document interest, while long-term re-finding behaviour is modelled via a query-specific Gaussian mixture that drifts over time. They report improvements in the performance of personalized ranking.

2.1.6 Other Approaches

Lu et al. [19] propose KEPS, a knowledge graph enhanced personalized search model that performs personalized entity linking on the

user's queries to form a better intent representation, builds a knowledge enhanced user profile by storing linked entities and predicted intents from search history in a memory network, and uses click feedback to post-adjust entity linking for past queries to correct earlier disambiguation errors. Experiments on the AOL log show that these three components jointly improve ranking accuracy.

Another approach [51] proposes query-specific concept-based user profiles learned from clickthrough logs, explicitly modelling both positive and negative preferences. The approach utilizes Ranking SVM [52] to learn weighted concept vectors that improve personalized query clustering. The study finds that incorporating negative preferences not only improves cluster quality but also increases the separation between similar and dissimilar queries, pushing them into more distant clusters.

2.2 Vector Embeddings for Personalized Search

In contrast to traditional feature-based methods, a growing line of work frames personalization as representation learning in shared embedding spaces. This section organizes embedding-based personalization into four themes: (i) query expansion and reformulation, (ii) user-context injection methods that personalize matching and scoring directly, including self-supervised contrastive pretraining under sparse logs, (iii) product and marketplace search models that jointly embed users, queries, and items and often use sequential transformers over purchase or click histories, and (iv) embeddings that incorporate social graphs, groups, or knowledge graphs to disambiguate intent and improve cold-start robustness. We also briefly note on multimodal personalized retrieval.

2.2.1 Query Expansion and Query Reformulation

These methods personalize at the query level. They use embedding proximity to add terms, substitute terms, or rewrite queries, aiming to resolve vocabulary mismatch and capture user-specific semantics.

Amer et al. [21] used Word2Vec [53] embeddings for query expansion by training the embedding model on the user profile text.

They discovered that sparse or low-quality profile text can yield weak embeddings. Therefore, the personalized variant does not consistently improve effectiveness and may require richer profile evidence or auxiliary signals (e.g., neighbours) to be useful.

Kuzi et al. [22] proposed query expansion where Word2Vec is trained once on the entire target corpus and expansion terms are selected by semantic proximity in the learned vector space, either to the query as a whole or to individual query terms. The selected terms are used either directly to expand the query or are integrated with a pseudo-relevance-feedback relevance model. The authors report significant improvements in performance over using the raw query alone. The method is not personalized, but it can serve as a starting point for personalization by introducing user-specific embedding spaces or combining them with the global space.

Zhang [32] personalized query reformulation with embeddings integrated into both candidate generation and scoring. Candidates are generated via term expansions or substitutions augmented with a user embedding, and then re-ranked using a graphical model enriched by term, user, and topic embeddings to capture semantic consistency and topical dependencies. Limitations include reliance on sufficient query history to form topic profiles and the constrained expressivity of single-term edits, which may miss more complex reformulations.

Bassani et al. [33] proposed a method, PQEWC, for personalized query expansion with contextual word embeddings, addressing redundancy and scalability caused by multiple similar embeddings. Their approach clusters user-term embeddings offline and selects representative expansion terms per cluster, coupled with an approximation scheme that enables sub-millisecond expansion while improving effectiveness and term diversity. Limitations include reliance on cosine similarity (rather than a learned scoring function) and the use of a fixed number of expansion terms, rather than predicting the optimal expansion amount.

2.2.2 Approaches Based on User Context

These approaches personalize ranking by injecting user context, learned from a person's past queries, clicks, or stored memories, directly into the matching and scoring process.

Vu et al. [23] proposed a user profiling method where each profile is represented by a user embedding together with two projection matrices that capture interest-dependent aspects of submitted queries and clicked (relevant) documents, while the user embedding captures the relationship between the queries and documents in this user interest-dependent subspace. The resulting profile is then used directly to re-rank a commercial search engine's results, yielding stable and significant improvements in ranking.

Yao et al. [26] proposed PEPS, which personalizes ranking by learning personal word embeddings so that ambiguous terms acquire user-specific semantics from each person's search history. Queries and documents are encoded using multi-head self-attention to form personalized contextual representations, and a neural matching model (KNRM [54]) is used to score relevance. Experiments on large-scale query logs showed significant performance improvements.

Mysore et al. [35] proposed CtrlCE, a controllable personalized search cross-encoder augmented with an editable user memory built from historical user documents (item-based or concept-based). A calibrated mixing model determines when personalization should be applied.

PSSL [29] proposed a two-stage, self-supervised framework for personalized search that pre-trains both a sentence encoder and a sequence encoder before fine-tuning on ranking. It uses contrastive sampling to mine paired signals from query logs at two levels: within a user (self-contrastive) and across users (user-contrastive). From these, it constructs four kinds of contrastive pairs: query pairs, document pairs, sequence augmentation pairs, and user pairs. Then, it optimizes contrastive losses so that representations of similar queries/documents/behaviour sequences are pulled closer together in the embedding space, yielding more robust representations for personalization under sparse logs.

2.2.3 Product or Marketplace Search and Recommendation

These works focus on commerce retrieval, where short queries, rich item content (e.g., titles or reviews), and strong behavioural signals (e.g., click or purchase) make joint embedding spaces particularly effective. Models typically embed users, queries, and items or products

together, often with sequence models over history to capture evolving intent.

In a field of personalized product search, Ai et al. [24] proposed a hierarchical embedding model that jointly learns latent representations of users, queries, and products from associated language data (e.g., reviews). Experiments on Amazon benchmarks showed consistent performance improvements, while the authors note that gains and the importance of personalization vary substantially across datasets (e.g., with review or query sparsity).

Bi et al. [27] proposed TEM, a transformer-based embedding model that encodes the sequence of the current query, along with the purchase history, to dynamically control how much personalization should influence retrieval. Multi-layer TEM can also model interactions among historical purchases, yielding more informative attention weights and better dynamic representations. Experiments on the Amazon product search dataset show significant performance improvements.

In an applied marketplace setting, Grbovic and Cheng [25] learned Airbnb-specific listing and user embeddings for real-time personalization in search ranking and similar-listing recommendations, explicitly modelling each guest's short-term and long-term interests.

Wang et al. [31] proposed an embedding model that maps users and items into a shared vector space for e-commerce recommendations at eBay. To reduce cold-start, item embeddings rely on content features, while user embeddings are learned from multi-modal onsite behaviour (e.g., item clicking and query searching). Candidates are retrieved via KNN in the embedding space. A limitation noted by authors is the single-vector user representation, which may collapse multiple concurrent interests.

Furthermore, Qin et al. [36] proposed MAPS for personalized product search by incorporating pre-search consultation text to capture user motivations that are not expressed in short queries. Queries and consultations are embedded into a unified semantic space with Large Language Model (LLM). A Mixture of Attention Experts focuses on critical semantics and dual alignment (contrastive and bidirectional attention), bridges consultation text, product features, and user history. While MAPS improves personalized search via query–consultation

semantic alignment, it does not fully address efficiency and real-time scalability and lacks explicit modelling of evolving user behaviour.

2.2.4 Social-augmented and Knowledge-augmented Embeddings

When individual history is sparse or ambiguous, embedding personalization can be improved by injecting structure beyond the user's own logs, such as friend networks, group behaviour, or knowledge graphs.

FNPS [28] is a group-based personalized search model that constructs dynamic group profiles from users' friend networks and historical search behaviour using graph attention and cross-attention mechanisms, and combines these with individual profiles to generate personalized results.

ReBKC [30] treats the knowledge graph as a heterogeneous network and learns unified embeddings that combine users' behaviour signals with knowledge relations. It mines short-term and long-term preferences from historical interactions using a self-attention mechanism, and then leverages graph attention over the knowledge graph neighbourhood to weight different knowledge entities by their user-specific importance. The authors note that their knowledge graph embeddings are primarily learned to reconstruct triples rather than being optimized for recommendation.

Baek et al. [34] recently proposed a lightweight LLM-based personalization framework that constructs an entity-centric knowledge store for each user from their search and browsing histories, aligns these aggregated entities with public knowledge graphs, and then leverages the resulting user-specific context to augment LLM prompts for tasks like contextual query suggestion, yielding more relevant and useful personalized generations.

2.2.5 Multimodal Personalized Retrieval

Finally, in the multimodal domain, Ryan et al. [37] propose POLAR for personalized vision-language retrieval by adapting a CLIP model's language encoder via regularized low-rank adaptation (LoRA) on the final layer, learning new personal concepts from a few images while aiming to preserve general knowledge.

2.3 Our Approach

Most existing personalization models intervene at the representation or ranking stages. They mine user-specific or group-specific signals from a person's own history and context (queries, clicks, dwell time, time, location), and often also from other users (similar users, groups, social graphs), then apply personalization through query expansion/reformulation, user/profile embeddings, or learned ranking functions that re-rank candidates produced by a global retriever. This can deliver substantial gains, but it often couples personalization with data and the ranking model, increasing the amount of user-specific state to maintain (e.g., embeddings, memories, models, features or profiles) and complicating deployment. More broadly, even when personalization is powerful, many pipelines still rely on a shared first-stage retrieval mechanism to generate candidates, then personalize after retrieval. In vector databases, however, the candidate set is determined by an index built around a fixed similarity function (e.g., Euclidean), so a *user's true nearest neighbours under their personalised notion of similarity* may never be considered unless the system rebuilds indices per user.

In contrast, our approach (see Chapter 3 for more details) personalizes search in vector databases at the level of the *similarity function* itself. Specifically, we model personalization as learning a per-user matrix that parameterizes a user-specific similarity function (Mahalanobis distance) over a common embedding space, indexed by a fixed index similarity function (Euclidean distance). Each user is represented by only this small matrix, which encodes how they weigh and correlate the embedding dimensions. The per-user matrix is learned from relevance feedback. Users' judgments about retrieved items, in the form of triplets (*query, positive, negative*), are used to update their matrix, utilizing metric learning [38] to learn a similarity function that brings relevant items closer together while pushing irrelevant items further apart. At retrieval time, the system first *filters* the shared index using the index metric (Euclidean) to retrieve an enlarged candidate set, then *refines* that set using the user's personalized Mahalanobis distance. The enlargement step is justified by a theoretical lower bounding relationship between Euclidean and Mahalanobis distances [39]. This relationship provides a way to expand the index metric search scope

so that the refinement stage guarantees retrieval of the precise nearest neighbours under the user's personalized Mahalanobis view, therefore providing theoretical bounds on the approach. From a practical point of view, we treat the vector index as a black box built for a fixed index metric (Euclidean). Our method requires only that the index support similarity search (range or kNN queries) under this metric. It does not depend on a particular indexing scheme (metric trees, HNSW, LSH, quantization, etc.). Moreover, personalization requires storing only the per-user matrix learned from feedback. Because embeddings and the index remain shared, deployment is straightforward, and the amount of user-specific state is small compared to approaches that maintain per-user profiles, memories, or retrained models.

3 Achieved Results

This chapter summarizes the research results achieved so far.

3.1 Towards Personalized Similarity Search for Vector Databases

In *Towards Personalized Similarity Search for Vector Databases* [55] (full paper in Appendix A), we proposed a personalised similarity search system for vector databases where the index is built with an **Index metric** (e.g., Euclidean distance) over learned embeddings, while users query with a **View metric** given by a Mahalanobis distance. Figure 3.1 shows the architecture and the roles of the metrics.

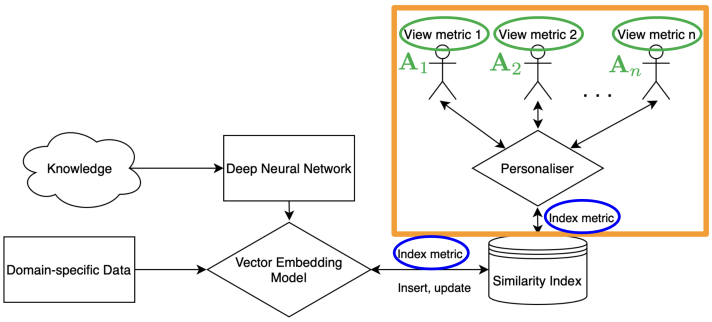


Figure 3.1: Overview of the personalised similarity search system

Personalisation follows the *metric learning* paradigm [38]. Each user is associated with a matrix **A** (initially the identity matrix, so the **View metric** equals Euclidean). During the search, users provide relevance feedback on the returned results, which is used to update **A**, therefore adapting the metric to the user's notion of similarity.

Since the index remains Euclidean, the system must retrieve larger candidate sets by increasing the search range or the number of nearest neighbours to ensure that the Mahalanobis nearest neighbours are included. This guarantee is grounded in the *lower-bounding relationship* between Euclidean and Mahalanobis distances [56]:

- Having all eigenvalues λ_j of $\mathbf{A}$, and given two vectors, their Euclidean distance d_E is always smaller or equal to the scaled Mahalanobis distance $d_M(\mathbf{A})$:

$$\frac{d_M(\mathbf{A})}{\sqrt{min_j(\lambda_j)}} \geq d_E.$$

- This introduces the scaling factor:

$$s_{M,E}(\mathbf{A}) = 1/\sqrt{min_j(\lambda_j)}.$$

In the work, we focused on the *range queries*. Using the scaling factor, we defined the Euclidean range $r_E = r_M \cdot s_{M,E}(\mathbf{A})$, which is used for retrieving the candidate set R_{d_E,r_E} using range query on the Euclidean index. The range r_E can be calculated instantly and is query independent. We further defined the optimum range $r_{\mathrm{OptE}}(\overrightarrow{q})$ as the minimum Euclidean range, such that the answer set $R_{d_E,r_{\mathrm{OptE}}(\overrightarrow{q})}$ retrieved using this range contains the retrieved answer set R_{d_M,r_M} using the Mahalanobis range r_M and the Mahalanobis distance parameterised by $\mathbf{A}$. However, this range cannot be calculated exactly without first having the wanted answer set R_{d_M,r_M}, and it is dependent on the query $\overrightarrow{q}$. Figure 3.2 shows the inclusions between the sets.

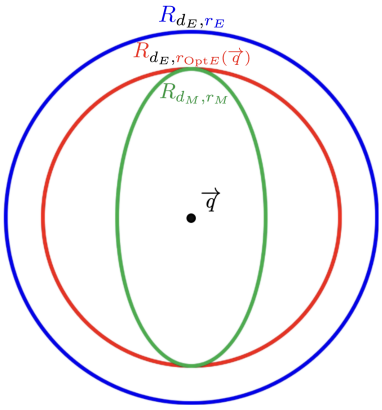


Figure 3.2: Inclusions between the answer sets

We experimentally verified that $R_{d_M,r_M} \subseteq R_{d_E,r_{\mathrm{OptE}}(\overrightarrow{q})} \subseteq R_{d_E,r_E}$. In practice, we suggested the *filter and refine* principle: the engine first

uses Euclidean distance to efficiently retrieve a candidate superset R_{d_E,r_E}, which is refined using the more expensive Mahalanobis distance parameterised by $\mathbf{A}$ to obtain the answer set R_{d_M,r_M}.

For the validation of the approach, we used 226,778 CLIP [57] embeddings from the Profiset [58] dataset. We trained 30 matrices using metric learning models ITML [59] and SDML [60], and different numbers of pairs of vectors synthetically chosen as similar or dissimilar. We abstracted from the semantics of the vectors and learned the matrices purely on a geometric level. We conducted all experiments using 1000 test query vectors randomly selected from the dataset. Figure 3.3 plots the ranges r_E (blue), r_M (green), and $r_{\mathrm{OptE}}(\overrightarrow{q})$ (red) for 6 different matrices $\mathbf{A}$, while the corresponding scaling factor $s_{M,E}(\mathbf{A})$ is shown atop each plot. The y-axis reports range values and the x-axis indexes query vectors $\overrightarrow{q}$, ordered so that r_M is non-decreasing.

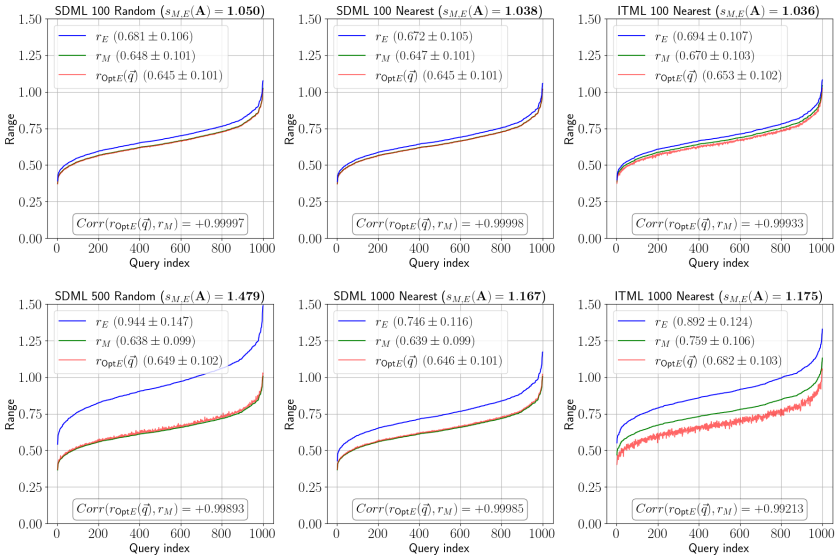


Figure 3.3: r_E, r_M, and $r_{\mathrm{OptE}}(\overrightarrow{q})$ across test queries for six matrices.

As expected, $r_{\mathrm{OptE}}(\overrightarrow{q})$ never exceeds r_E, and moreover $r_{\mathrm{OptE}}(\overrightarrow{q})$ is highly correlated with r_M (Pearson $Corr(r_{\mathrm{OptE}}(\overrightarrow{q}), r_M)$ between $+0.992$ and $+0.99998$), suggesting that estimating $r_{\mathrm{OptE}}(\overrightarrow{q})$ from r_M is a promising direction for future work.

We also examined the sizes of the Euclidean candidate sets retrieved with ranges r_E and $r_{\mathrm{OptE}}(\overrightarrow{q})$. Table 3.1 reports the average

numbers of retrieved vectors, denoted $\#_E$ and $\#_{\mathrm{Opt}E}$, over all test query vectors (each Mahalanobis answer set has 100 vectors).

Table 3.1: Scaling factors $s_{M,E}(\mathbf{A})$ and average candidate set sizes under Euclidean retrieval: $\#_E$ for range r_E and $\#_{\mathrm{Opt}E}$ for range $r_{\mathrm{Opt}E}(\overrightarrow{q})$. The dataset contains 226,778 vectors.

Learned matrix $\mathbf{A}$	$s_{M,E}(\mathbf{A})$	Avg. $\#_E$	Avg. $\#_{\mathrm{Opt}E}$
SDML 100 Random	1.050	317 ± 263	102 ± 2
SDML 500 Random	1.479	$108{,}025 \pm 82{,}569$	115 ± 23
SDML 100 Nearest	1.038	228 ± 110	101 ± 2
SDML 1000 Nearest	1.167	$4{,}110 \pm 10{,}828$	104 ± 5
ITML 100 Nearest	1.036	497 ± 528	128 ± 23
ITML 1000 Nearest	1.175	$70{,}681 \pm 69{,}508$	321 ± 235

For matrices with small scaling factors, Euclidean retrieval with r_E typically returned only hundreds of candidates, which can be efficiently refined. However, with higher scaling factors, the candidate set size under r_E grew by orders of magnitude, whereas using $r_{\mathrm{Opt}E}(\overrightarrow{q})$ remains close to 100–300 candidates. This motivated the investigation of conditions (e.g., other metric learning models, real semantic feedback, or approximation of the optimal range), which could lead to small scaling factors as future work.

3.2 Integrating Relevance Feedback for Effective Personalisation in Vector Search

In *Integrating Relevance Feedback for Effective Personalisation in Vector Search* [61] (full paper in Appendix B), we followed up on our previous work and addressed the limitations: (i) the profile matrices had been learned from synthetic feedback (so real-world efficacy was untested) and (ii) the evaluation covered only two established metric learning models, rather than a broader range of contemporary approaches.

Addressing the first limitation, we introduced a pipeline that captures *semantic* relevance feedback from user interactions and integrates it into personalised vector search as a cyclic loop (Fig. 3.4). In each iteration, the user submits a query using their matrix $\mathbf{A}$. The system returns results via *filter and refine*: it first performs a Euclidean range

search where the radius is enlarged by the scaling factor derived from **A**, and then refines the candidates using the Mahalanobis distance parameterised by **A**. The user then marks irrelevant results (negative), while all unmarked results are treated as relevant (positive). The system converts these judgements into triplets of the form (*query, positive, negative*) and uses them to update **A** via a metric learning model. This loop repeats for subsequent queries, continuously adapting **A** until further updates become impractical due to a large scaling factor.

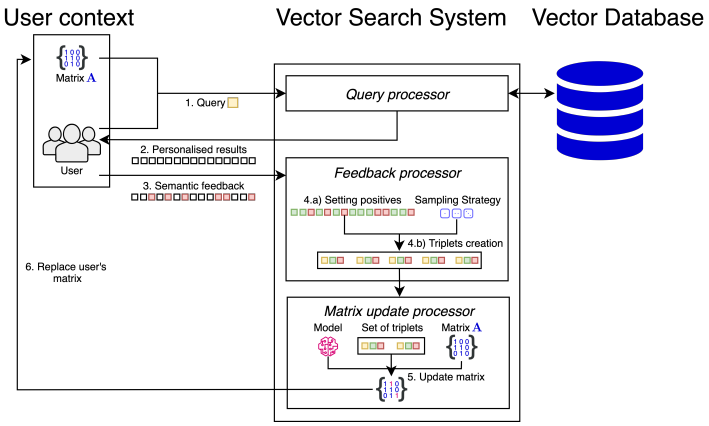


Figure 3.4: Workflow of the semantic-feedback pipeline. The query is shown in yellow, positive results in green, and negative results in red.

A key design choice is how feedback is transformed into triplets for learning. We therefore defined three feedback strategies: (i) repeatedly sampling one positive and one negative to form a single triplet per draw (for $k \in \{1, 2, 4, 8, 16, 32, 64\}$ draws, with or without replacement), (ii) sampling a negative (without replacement) and pairing it with all available positives, with updates applied either as a single batch or sequentially, and (iii) pairing all negatives with all positives, but triggering training only after accumulating feedback from $q \in \{1, 2, 5, 10, 20, 40\}$ queries (one update per batch of q queries).

To translate empirical results into deployment guidance, we defined three operational scenarios based on the trade-off between personalisation quality and efficiency. We measure personalisation benefit as ΔMAP, i.e., the change in mean-average precision when switching the ranking metric from Euclidean to Mahalanobis parameterised by

the learned **A**. The *BPSF* scenario targets a balanced trade-off between ΔMAP and a small scaling factor, *MSAP* minimises the scaling factor while enforcing an acceptable improvement ($\Delta MAP \geq 0.1$), and *MP* prioritises maximising ΔMAP (strongest personalisation). Each scenario is considered in three variants: a *Baseline* (*BAS*) variant, a *Low Learning Time* (*LLT*) variant prioritising training efficiency, and a *Rank Stability* (*RS*) variant favouring stable rankings.

To address the second limitation, We used or re-implemented a broad suite of widely used metric learning methods: OASIS [40], ITML [59], POLA [62], AROMA [63], OMDML [64], MLOML [65], RobustODML [66], SORS [67], AdaSORS [67], and OPML [68].

We ran an extensive hyperparameter grid search to identify the best-performing combination of (i) metric learning model and hyperparameters and (ii) feedback strategy for each operational scenario and its variants. Table 3.2 summarises the best configurations and their outcomes in terms of the scaling factor (FSF), the personalisation gain (ΔMAP), average learning time (ALT), and rank-stability measures (Average Spearman@1000 and Average Kendall's Tau@1000).

Table 3.2: Calculated evaluation measures for the best model given the scenario and its variant. The Model column is in the form: model and hyperparameters. The Feedback column is in the form: feedback strategy and parameters – 1:(Number of pairs, Replacement), 2:(Batch), 3:(Number of queries).

Scenario–Variant	Model	Feedback	FSF	ΔMAP	ALT	AS@1000	AK@1000
BPSF-BAS	OMDML β: 1e-3, C: 4e-3, γ: 7e-3	2 True	1.095	0.203	1.373	0.708	0.532
BPSF-LLT	OASIS C: 7e-2, *enforce_psd*: False	1 1, False	1.147	0.103	0.001	0.621	0.456
BPSF-RS	OMDML β: 7e-2, C: 1e-3, γ: 1e-4	2 False	1.039	0.134	1.416	0.899	0.738
MSAP-BAS	MLOML *nl*: 2, γ: 1e-4, *act*: tanh, λ: 7e-6	3 5	1.017	0.102	18.098	0.863	0.693
MSAP-LLT	OASIS C: 1e-2, *enforce_psd*: False	1 8, False	1.062	0.106	0.008	0.813	0.636
MSAP-RS	OMDML β: 1.0, C: 7e-4, γ: 4e-4	1 64, True	1.028	0.101	1.768	0.939	0.798
MP-BAS	OMDML β: 7e-1, C: 1e-2, γ: 1e-2	2 False	1.148	0.211	0.712	0.566	0.411
MP-LLT	OASIS C: 7e-2, *enforce_psd*: False	1 2, True	1.271	0.130	0.003	0.308	0.211
MP-RS	OMDML β: 4e-2, C: 1e-2, γ: 1e-4	2 True	1.077	0.193	1.104	0.788	0.609

Two clear patterns emerged: *OMDML* dominates when effectiveness or rank stability is prioritised (BPSF-BAS/RS, MP-BAS/RS, MSAP-RS), whereas *OASIS* is consistently selected for fast updates (all LLT variants), with *MLOML* appearing only in MSAP-BAS. Overall, the selected configurations maintain low scaling factors (scaling factor ≤ 1.15 in most cases) while achieving gains up to $\Delta MAP = 0.211$, showing that semantic feedback can improve relevance without substantial candidate set inflation, making the whole proposed approach effective and efficient.

3.3 Author's Publications

Below are the publications relevant to this work. The full texts of these publications are available in Appendix A and Appendix B.

1. Marek Mahrík, **Matúš Šikyňa**, Vladimir Mic and Pavel Zezula. "Towards Personalized Similarity Search for Vector Databases". In: *17th International Conference on Similarity Search and Applications*. SISAP 2024, Providence, RI, USA, pp. 126–139.

    - CORE conference ranking B
    - My contribution: 30%
    - Evaluation of experiments, data visualisation, related work review, drafting and editing the paper, and submission management as corresponding author.

2. **Matúš Šikyňa** and Pavel Zezula. "Integrating Relevance Feedback for Effective Personalisation in Vector Search". In: *18th International Conference on Similarity Search and Applications*. SISAP 2025, Reykjavik, Iceland, pp. 154–162.

    - CORE conference ranking B
    - My contribution: 90%
    - Conceptualisation and schema design, implementation, experiment evaluation and interpretation, data visualisation, related work review, drafting and revising the paper, and managing the submission as corresponding author.

Bibliography

1. ZHAO, Xiangyu; WANG, Maolin; ZHAO, Xinjian; LI, Jiansheng; ZHOU, Shucheng; YIN, Dawei; LI, Qing; TANG, Jiliang; GUO, Ruocheng. Embedding in Recommender Systems: A Survey. *arXiv preprint arXiv:2310.18608*. 2023.
2. ZHOU, Sheng; XU, Hongjia; ZHENG, Zhuonan; CHEN, Jiawei; LI, Zhao; BU, Jiajun; WU, Jia; WANG, Xin; ZHU, Wenwu; ESTER, Martin. A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions. *ACM Computing Surveys*. 2025, vol. 57, no. 3, 69:1–69:38.
3. COSTA, Liliane Soares da; OLIVEIRA, Italo Lopes; FILETO, Renato. Text classification using embeddings: a survey. *Knowledge and Information Systems*. 2023, vol. 65, no. 7, pp. 2761–2803.
4. WANG, Jiajia; HUANG, Jimmy Xiangji; TU, Xinhui; WANG, Junmei; HUANG, Angela Jennifer; LASKAR, Md. Tahmid Rahman; BHUIYAN, Amran. Utilizing BERT for Information Retrieval: Survey, Applications, Resources, and Challenges. *ACM Computing Surveys*. 2024, vol. 56, no. 7, 185:1–185:33.
5. TVERSKY, Amos. Features of Similarity. *Psychological Review*. 1977, vol. 84, no. 4, pp. 327–352.
6. COX, Ingemar J.; MILLER, Matthew L.; MINKA, Thomas P.; PAPATHOMAS, Thomas V.; YIANILOS, Peter N. The Bayesian image retrieval system, PicHunter: theory, implementation, and psychophysical experiments. *IEEE Transactions on Image Processing*. 2000, vol. 9, no. 1, pp. 20–37.
7. SUGIYAMA, Kazunari; HATANO, Kenji; YOSHIKAWA, Masatoshi. Adaptive web search based on user profile constructed without any effort from users. In: *Proceedings of the 13th international conference on World Wide Web*. 2004, pp. 675–684.
8. SHEN, Xuehua; TAN, Bin; ZHAI, ChengXiang. Implicit user modeling for personalized search. In: *Proceedings of the 14th ACM international conference on Information and Knowledge Management*. 2005, pp. 824–831.
9. TEEVAN, Jaime; DUMAIS, Susan T.; HORVITZ, Eric. Personalizing search via automated analysis of interests and activities. In:

Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. 2005, pp. 449–456.

10. SPERETTA, Mirco; GAUCH, Susan. Personalized Search Based on User Search Histories. In: *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence*. 2005, pp. 622–628.
11. QIU, Feng; CHO, Junghoo. Automatic identification of user interest for personalized search. In: *Proceedings of the 15th international conference on World Wide Web*. 2006, pp. 727–736.
12. LU, Yumao; PENG, Fuchun; WEI, Xing; DUMOULIN, Benoıt. Personalize web search results with user's location. In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2010, pp. 763–764.
13. BENNETT, Paul N.; WHITE, Ryen W.; CHU, Wei; DUMAIS, Susan T.; BAILEY, Peter; BORISYUK, Fedor; CUI, Xiaoyuan. Modeling the impact of short- and long-term behavior on search personalization. In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2012, pp. 185–194.
14. YI, Xing; HONG, Liangjie; ZHONG, Erheng; LIU, Nathan Nan; RAJAN, Suju. Beyond clicks: dwell time for personalization. In: *Proceedings of the 8th ACM Conference on Recommender Systems*. 2014, pp. 113–120.
15. VU, Thanh Tien; SONG, Dawei; WILLIS, Alistair; TRAN, Son Ngoc; LI, Jingfei. Improving search personalisation with dynamic group formation. In: *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2014, pp. 951–954.
16. BOUADJENEK, Mohamed Reda; HACID, Hakim; BOUZEGHOUB, Mokrane; VAKALI, Athena. PerSaDoR: Personalized social document representation for improving web search. *Information Sciences*. [N.d.], vol. 369, pp. 614–633.
17. ZAMANI, Hamed; BENDERSKY, Michael; ZHANG, Mingyang; WANG, Xuanhui. Situational Context for Ranking in Personal Search. In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 1531–1540.
18. GE, Songwei; DOU, Zhicheng; JIANG, Zhengbao; NIE, Jian-Yun; WEN, Ji-Rong. Personalizing Search Results Using Hierarchical RNN with Query-aware Attention. In: *Proceedings of the 27th ACM*

International Conference on Information and Knowledge Management. 2018, pp. 347–356.

19. LU, Shuqi; DOU, Zhicheng; XIONG, Chenyan; WANG, Xiaojie; WEN, Ji-Rong. Knowledge Enhanced Personalized Search. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 709–718.
20. MA, Zhengyi; DOU, Zhicheng; BIAN, Guanyue; WEN, Ji-Rong. PSTIE: Time Information Enhanced Personalized Search. In: *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. 2020, pp. 1075–1084.
21. AMER, Nawal Ould; MULHEM, Philippe; GÉRY, Mathias. Toward Word Embedding for Personalized Information Retrieval. 2016.
22. KUZI, Saar; SHTOK, Anna; KURLAND, Oren. Query Expansion Using Word Embeddings. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 1929–1932.
23. VU, Thanh; NGUYEN, Dat Quoc; JOHNSON, Mark; SONG, Dawei; WILLIS, Alistair. Search Personalization with Embeddings. In: *39th European Conference on Information Retrieval*. 2017, vol. 10193, pp. 598–604.
24. AI, Qingyao; ZHANG, Yongfeng; BI, Keping; CHEN, Xu; CROFT, W. Bruce. Learning a Hierarchical Embedding Model for Personalized Product Search. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2017, pp. 645–654.
25. GRBOVIC, Mihajlo; CHENG, Haibin. Real-time Personalization using Embeddings for Search Ranking at Airbnb. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2018, pp. 311–320.
26. YAO, Jing; DOU, Zhicheng; WEN, Ji-Rong. Employing Personal Word Embeddings for Personalized Search. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 1359–1368.
27. BI, Keping; AI, Qingyao; CROFT, W. Bruce. A Transformer-based Embedding Model for Personalized Product Search. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 1521–1524.

28. ZHOU, Yujia; DOU, Zhicheng; WEI, Bingzheng; XIE, Ruobing; WEN, Ji-Rong. Group based Personalized Search by Integrating Search Behaviour and Friend Network. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 92–101.
29. ZHOU, Yujia; DOU, Zhicheng; ZHU, Yutao; WEN, Ji-Rong. PSSL: Self-supervised Learning for Personalized Search with Contrastive Sampling. In: *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 2021, pp. 2749–2758.
30. HUI, Bei; ZHANG, Lizong; ZHOU, Xue; WEN, Xiao; NIAN, Yuhui. Personalized recommendation system based on knowledge embedding and historical behavior. *Applied Intelligence*. 2022, vol. 52, no. 1, pp. 954–966.
31. WANG, Tian; BROVMAN, Yuri M.; MADHVANATH, Sriganesh. Personalized Embedding-based e-Commerce Recommendations at eBay. *CoRR*. 2021, vol. abs/2102.06156.
32. ZHANG, Xiaojuan. Improving personalised query reformulation with embeddings. *Journal of Information Science*. 2022, vol. 48, no. 4, pp. 503–523.
33. BASSANI, Elias; TONELLOTTO, Nicola; PASI, Gabriella. Personalized Query Expansion with Contextual Word Embeddings. *ACM Transactions on Information Systems*. 2023, vol. 42, no. 2, pp. 1–35.
34. BAEK, Jinheon; CHANDRASEKARAN, Nirupama; CUCERZAN, Silviu; HERRING, Allen; JAUHAR, Sujay Kumar. Knowledge-Augmented Large Language Models for Personalized Contextual Query Suggestion. In: *Proceedings of the ACM Web Conference 2024*. 2024, pp. 3355–3366.
35. MYSORE, Sheshera; DHANANIA, Garima; PATIL, Kishor; MCCALLUM, Andrew; KALLUMADI, Surya; ZAMANI, Hamed. Bridging Personalization and Control in Scientific Personalized Search. In: *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2025, pp. 2309–2319.
36. QIN, Weicong; XU, Yi; YU, Weijie; SHEN, Chenglei; HE, Ming; FAN, Jianping; ZHANG, Xiao; XU, Jun. MAPS: Motivation-Aware Personalized Search via LLM-Driven Consultation Alignment.

In: *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*. 2025, pp. 3039–3051.

37. RYAN, Fiona; SIVIC, Josef; HEILBRON, Fabian Caba; HOFFMAN, Judy; REHG, James M.; RUSSELL, Bryan C. Improving Personalized Search with Regularized Low-Rank Parameter Updates. In: *Proceedings of the Computer Vision and Pattern Recognition Conference 2025*. 2025, pp. 19748–19757.
38. BELLET, Aurélien; HABRARD, Amaury; SEBBAN, Marc. *Metric Learning*. 2015. Synthesis Lectures on Artificial Intelligence and Machine Learning. ISBN 978-3-031-00444-5.
39. CIACCIA, Paolo; PATELLA, Marco. Searching in metric spaces with user-defined and approximate distances. *ACM Transactions on Database Systems*. 2002, vol. 27, no. 4, pp. 398–437.
40. CHECHIK, Gal; SHALIT, Uri; SHARMA, Varun; BENGIO, Samy. An Online Algorithm for Large Scale Image Similarity Learning. In: *Proceedings of the 23rd International Conference on Neural Information Processing Systems*. 2009, pp. 306–314.
41. STAMOU, Sofia; NTOULAS, Alexandros. Search personalization through query and page topical analysis. *User Modeling and User-Adapted Interaction*. 2009, vol. 19, no. 1-2, pp. 5–33.
42. VU, Thanh; WILLIS, Alistair; KRUSCHWITZ, Udo; SONG, Dawei. Personalised Query Suggestion for Intranet Search with Temporal User Profiling. In: *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*. 2017, pp. 265–268.
43. BURGES, Christopher J. C.; RAGNO, Robert; LE, Quoc Viet. Learning to Rank with Nonsmooth Cost Functions. In: *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*. 2006, pp. 193–200.
44. MA, Zhongming; PANT, Gautam; SHENG, Olivia R. Liu. Interest-based personalized search. *ACM Transactions on Information Systems*. 2007, vol. 25, no. 1, p. 5.
45. SIEG, Ahu; MOBASHER, Bamshad; BURKE, Robin. Ontological user profiles for personalized web search. In: *Proceedings of the 5th Workshop on Intelligent Techniques for Web Personalization*. 2007, pp. 84–91.
46. XUE, Gui-Rong; HAN, Jie; YU, Yong; YANG, Qiang. User language model for collaborative personalized search. *ACM Transactions on Information Systems*. 2009, vol. 27, no. 2, pp. 1–28.

47. CARMEL, David; ZWERDLING, Naama; GUY, Ido; OFEK-KOIFMAN, Shila; HAR'EL, Nadav; RONEN, Inbal; UZIEL, Erel; YOGEV, Sivan; CHERNOV, Sergey. Personalized social search based on the user's social network. In: *Proceedings of the 18th ACM conference on Information and Knowledge Management*. 2009, pp. 1227–1236.
48. TEEVAN, Jaime; MORRIS, Meredith Ringel; BUSH, Steve. Discovering and using groups to improve personalized search. In: *Proceedings of the Second ACM International Conference on Web Search and Data Mining*. 2009, pp. 15–24.
49. WHITE, Ryen W.; CHU, Wei; AWADALLAH, Ahmed Hassan; HE, Xiaodong; SONG, Yang; WANG, Hongning. Enhancing personalized search by mining and modeling task behavior. In: *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 1411–1420.
50. BLEI, David M.; NG, Andrew Y.; JORDAN, Michael I. Latent Dirichlet Allocation. *Journal of Machine Learning Research*. 2003, vol. 3, pp. 993–1022.
51. LEUNG, Kenneth Wai-Ting; LEE, Dik Lun. Deriving Concept-Based User Profiles from Search Engine Logs. *IEEE Transactions on Knowledge and Data Engineering*. 2010, vol. 22, no. 7, pp. 969–982.
52. JOACHIMS, Thorsten. Optimizing search engines using clickthrough data. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2002, pp. 133–142.
53. MIKOLOV, Tomás; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient Estimation of Word Representations in Vector Space. In: *1st International Conference on Learning Representations*. 2013.
54. XIONG, Chenyan; DAI, Zhuyun; CALLAN, Jamie; LIU, Zhiyuan; POWER, Russell. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In: *Proceedings of the 40th International Conference on Research and Development in Information Retrieval*. 2017, pp. 55–64.
55. MAHRÍK, Marek; ŠIKYŇA, Matúš; MIC, Vladimir; ZEZULA, Pavel. Towards Personalized Similarity Search for Vector Databases. In: *17th International Conference on Similarity Search and Applications*. 2024, pp. 126–139.

56. CIACCIA, Paolo; PATELLA, Marco. Searching in metric spaces with user-defined and approximate distances. *ACM Transactions on Database Systems*. 2002, vol. 27, no. 4, pp. 398–437.
57. RADFORD, Alec; KIM, Jong Wook; HALLACY, Chris; RAMESH, Aditya; GOH, Gabriel; AGARWAL, Sandhini; SASTRY, Girish; ASKELL, Amanda; MISHKIN, Pamela; CLARK, Jack; KRUEGER, Gretchen; SUTSKEVER, Ilya. Learning Transferable Visual Models From Natural Language Supervision. In: *International Conference on Machine Learning, ICML 2021*. 2021, vol. 139, pp. 8748–8763.
58. BUDÍKOVÁ, Petra; BATKO, Michal; ZEZULA, Pavel. Evaluation Platform for Content-Based Image Retrieval Systems. In: *International Conference on Theory and Practice of Digital Libraries, TPDL 2011*. 2011, vol. 6966, pp. 130–142.
59. DAVIS, Jason V.; KULIS, Brian; JAIN, Prateek; SRA, Suvrit; DHILLON, Inderjit S. Information-theoretic metric learning. In: *International Conference on Machine Learning, ICML 2007*. 2007, vol. 227, pp. 209–216.
60. QI, Guo-Jun; TANG, Jinhui; ZHA, Zheng-Jun; CHUA, Tat-Seng; ZHANG, Hong-Jiang. An efficient sparse metric learning in high-dimensional space via l_1-penalized log-determinant regularization. In: *International Conference on Machine Learning, ICML 2009*. 2009, vol. 382, pp. 841–848.
61. ŠIKYŇA, Matúš; ZEZULA, Pavel. Integrating Relevance Feedback for Effective Personalisation in Vector Search. In: *18th International Conference on Similarity Search and Applications*. 2025, pp. 154–162.
62. SHALEV-SHWARTZ, Shai; SINGER, Yoram; NG, Andrew Y. Online and batch learning of pseudo-metrics. In: *ICML 2004*. 2004, vol. 69.
63. CRAMMER, Koby; CHECHIK, Gal. Adaptive Regularization for Weight Matrices. *arXiv preprint arXiv:1206.4639*. 2012.
64. WU, Pengcheng; HOI, Steven C. H.; ZHAO, Peilin; MIAO, Chunyan; LIU, Zhiyong. Online Multi-Modal Distance Metric Learning with Application to Image Retrieval. *IEEE Transactions on Knowledge and Data Engineering*. 2016, vol. 28, no. 2, pp. 454–467.
65. LI, Wenbin; LIU, Yanfang; HUO, Jing; SHI, Yinghuan; GAO, Yang; WANG, Lei; LUO, Jiebo. A Multilayer Framework for Online

Metric Learning. *IEEE Transactions on Neural Networks and Learning Systems*. 2023, vol. 34, no. 10, pp. 6701–6713.

66. ZABIHZADEH, Davood; TUAMA, Amar; KARAMI-MOLLAEE, Ali; MOUSAVIRAD, Seyed Jalaleddin. Low-rank robust online distance/similarity learning based on the rescaled hinge loss. *Applied Intelligence*. 2023, vol. 53, no. 1, pp. 634–657.
67. YAO, Dezhong; ZHAO, Peilin; YU, Chen; JIN, Hai; LI, Bin. Sparse Online Relative Similarity Learning. In: *2015 IEEE International Conference on Data Mining*. 2015, pp. 529–538.
68. LI, Wenbin; GAO, Yang; WANG, Lei; ZHOU, Luping; HUO, Jing; SHI, Yinghuan. OPML: A one-pass closed-form solution for online metric learning. *Pattern Recognition*. 2018, vol. 75, pp. 302–314.

A Appendix

Marek Mahrík, **Matúš Šikyňa**, Vladimir Mic and Pavel Zezula. "Towards Personalized Similarity Search for Vector Databases". In: *17th International Conference on Similarity Search and Applications*. SISAP 2024, Providence, RI, USA, pp. 126–139.

Towards Personalized Similarity Search for Vector Databases

Marek Mahrík[1], Matúš Šikyňa[1( )], Vladimir Mic[2], and Pavel Zezula[1]

[1] Faculty of Informatics, Masaryk University, Brno, Czech Republic
492727@mail.muni.cz, {xsikyna,zezula}@fi.muni.cz
[2] Department of Computer Science, Aarhus University, Aarhus, Denmark
v.mic@cs.au.dk

Abstract. The importance of similarity search has become prominent in the fast-evolving vector databases, which apply content embedding techniques on complex data to produce and manage large collections of high-dimensional vectors. Processing of such data is only possible by using a similarity function for storage, structure, and retrieval. However, if multiple users access the collection, their views on similarity can differ as similarity, in general, is subjective and context-dependent. In this article, we elaborate on the problem of a similarity search engine implementation, where users use a common index but search with personalised views of similarity, implemented by a possibly different similarity model. Specifically, we define a foundational theoretical framework and conduct experiments on real-life data to confirm the viability of such an approach. The experiments also indicate future research directions needed to propose and implement an effective and efficient personalised similarity search engine.

Keywords: Similarity search · Personalized similarity · Vector databases

1 Introduction

Current AI models produce vector embeddings that carry semantic information capable of performing context-based reasoning. The concept of similarity is central to this process, given its critical role in both learning and teaching. Such vector embedding models produce high-dimensional vectors, the collections of which are structured into partitions of a vector database index to facilitate efficient similarity querying. Typically, vector databases utilise Euclidean distance to structure the index and identify the most similar vectors in response to a given query vector.

This work was supported by the research grants (VIL50110) from VILLUM FONDEN, and by the Open Calls for Security Research 2023–2029 (OPSEC) program granted by the Ministry of the Interior of the Czech Republic under No. VK01010147 - Automated digital data forensics lab for complex crime detection.

E. Chávez et al. (Eds.): SISAP 2024, LNCS 15268, pp. 126–139, 2024.
https://doi.org/10.1007/978-3-031-75823-2_11

Table 1. Notation used throughout this paper

$D; X \subseteq D$	domain of the searched vectors; searched dataset
$\overrightarrow{q} \in \Theta \subseteq D$	query vector $\overrightarrow{q}$ and the set of tested query vectors Θ from D
$\overrightarrow{x}, \overrightarrow{y} \in D$	vectors from the domain of the searched vectors
δ	the dimensionality of vectors in the domain D
$\mathbf{A}$	positive semi-definite matrix used in the Mahalanobis distance function
d_E; $d_E(\overrightarrow{x}, \overrightarrow{y})$	Euclidean distance function; the Euclidean distance of $\overrightarrow{x}$ and $\overrightarrow{y}$
d_M	Mahalanobis distance function
$d_M(\overrightarrow{x}, \overrightarrow{y}, \mathbf{A})$	the Mahalanobis distance of $\overrightarrow{x}$ and $\overrightarrow{y}$ computed with matrix $\mathbf{A}$
$s_{M,E}(\mathbf{A})$	the scaling factor defining relation between functions d_E and d_M (Eq. 1)
r_M	search range given for Mahalanobis distance function d_M
r_E	search range defined for $d_E$by r_M and scaling factor $s_{M,E}(\mathbf{A})$ (Eq. 3)
$r_{\mathrm{Opt}E}(\overrightarrow{q})$	smallest search range defined in the Euclidean space which for given $\overrightarrow{q}$ returns a superset of $R_{d_M,r_M}(\overrightarrow{q})$ (Eq. 5)
$Q(\overrightarrow{q}, < f >, < r >)$	the range query with distance function <f> and search range <r>
$R_{<f>,<r>}(\overrightarrow{q})$	the (precise) answer of the range query $Q(\overrightarrow{q}, < f >, < r >)$
$\mathcal{P}_E$	the precision of answer $R_{d_E,r_E}(\overrightarrow{q})$ with respect to $R_{d_M,r_M}(\overrightarrow{q})$
$\mathcal{P}_{\mathrm{Opt}E}$	the precision of answer $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$ with respect to $R_{d_M,r_M}(\overrightarrow{q})$
$\#_E$	the number of vectors in $R_{d_E,r_E}(\overrightarrow{q})$ for a given $\overrightarrow{q}$
$\#_{\mathrm{Opt}E}$	the number of vectors in $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$ for a given $\overrightarrow{q}$

Undoubtedly, current AI models, trained on billions of sample objects, are capable of producing high-quality content embeddings. However, once these vectors are structured in an index using a fixed distance measure (typically, the Euclidean one), the process of partitioning and filtering remains the same for all database users. Such conditions invariably persist until a global, presumably costly, reorganisation is performed. This approach is in sharp contrast with human perception of similarity, which is subjective and context-dependent [13,24]. Contemporary approaches to similarity indexing include but are not limited to the metric space [26], LSH [6], graph-based (e.g., HNSW [15]), and quantization-based (e.g., OPQ [9]) concepts.

In this paper, we investigate the feasibility of a search engine architecture where a single similarity index accelerates query execution, even when personalised (i.e. modified or different) similarity measurements are used. We first review the related work in Sect. 2. Next, we outline the principal idea in Sect. 3 and introduce the theoretical foundations of a personalised similarity search engine in Sect. 4. In Sect. 5, the correctness of the theory and feasibility of the proposal are verified by experiments on contemporary data. Through this approach, we aim to better understand the design potential and identify its limitations and research challenges. The effort should lead to the proposal of an effective and efficient personalised search system. The specific findings are summarised in Sect. 6, and the paper concludes in Sect. 7.

2 Related Work

There have been many significant efforts to incorporate personalisation into search systems. One such approach is to utilise relevance feedback [19]. Traditional relevance feedback methods require explicit user feedback, which imposes a significant burden on users by requiring them to actively indicate the relevance of search results [12]. As opposed to explicit feedback, methods based on implicit feedback gather data such as past searches and click history [8,17,21] to improve the ranking of search results tailored to individual users. For instance, Shen et al. [20] explore short-term user search context given by previous queries for query expansion.

While implicit feedback focuses on individual user behaviour, another approach to enhancing personalisation is through collaborative filtering. Sugiyama et al. [22] derive users' preferences from their browsing history while utilising modified collaborative filtering to predict missing term weights in user profiles. Moreover, mining the search behaviour of similar users can further improve personalisation [23,25].

Deep learning techniques have shown significant potential in capturing complex patterns and user preferences. Ma et al. [14] presented time-aware LSTM architectures to model the evolving query intent and document interest over time, capturing both short-term preferences and long-term user interests. PSSL [27] utilises contrastive sampling methods to generate paired self-supervised data from users' query logs to improve the personalised results.

Personalisation can also be used for browsing. Bartolini et al. [2] presented PIBE, an adaptive image browsing system that allows users to customise a hierarchical organisation of images without the need for costly global reorganisations.

Our approach is different as it considers personalised instances of Mahalanobis distance functions adapted by users' feedback to capture the subjective similarity perception. Moreover, we leverage the relationship between Euclidean and Mahalanobis distances to personalise search results using an efficient Euclidean-based indexing structure without the need for global index reorganisation.

3 Principal Idea and the Approach

The idea of the proposed personalised similarity search system is based on the following two paradigms: (1) the *metric learning* [3], where individual subject similarity views are modelled by instances of the class of Mahalanobis distances, and (2) the *lower bounding relationship* between the Euclidean and the Mahalanobis distances: given two vectors, their Euclidean distance is always smaller or equal to the scaled Mahalanobis distance [5].

We assume that the index of the database is constructed from vectors typically extracted using the *deep learning* technology, which utilises the best possible reference knowledge available. After the index construction by the Euclidean distance, users start querying using the Mahalanobis distance function with the identity square matrix $\mathbf{A}$ – this distance function equals the Euclidean.

Each user owns a specific matrix **A**, initially the identity matrix. While searching, users provide feedback on result satisfaction, which is used to modify matrix **A**, reflecting the personalised views of users by establishing proper correlations between specific pairs of dimensions. The database index, structured with the Euclidean distance function, still executes queries using the Euclidean distances. Provided a user's matrix is not the identity matrix, the index needs to retrieve larger result sets so that the nearest neighbours defined by the Mahalanobis distances are included. Depending on the query type, this is achieved by using a proper extension, i.e., a larger search range or a larger number of nearest neighbours.

However, the problem is the size of the extension - too small extension does not guarantee full inclusion, i.e., it can decrease the recall as some nearest neighbours defined by the Mahalanobis distances can be missing, and too large extension provides a bulky result, i.e., it decreases the precision as many not-qualifying objects are also retrieved by the index.

From a technical point of view, we assume the *filter and refine* principle – the search engine first efficiently filters out most of the not-qualifying objects using the Euclidean distances and retrieves a superset of the answer, which is consequently refined by the Mahalanobis distances specified for individual users by their personalised matrix **A**. According to our experiments, the computation of Mahalanobis distances is approximately 2.5 times slower than the computation of Euclidean distances. It motivates us to investigate the search range that provides 100 per cent recall with the maximum possible precision.

In summary, we investigate the feasibility of a similarity search engine architecture – see the general model in Fig. 1 – where individual users see the vector database (i.e., *Similarity index*) and express their queries through a personalised instance of the Mahalanobis distance function (i.e., the *View metric*) while the database is structured and the index executes queries using the Euclidean distance measure (i.e., the *Index metric*). Performed experiments aim to identify the advantages and limitations of such a solution.

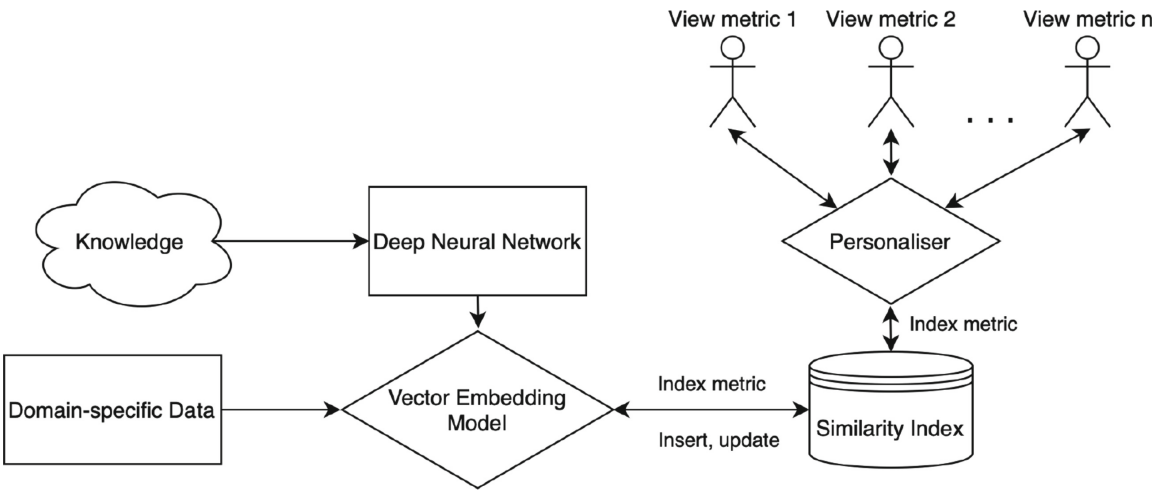


Fig. 1. Personalized similarity search system architecture

4 Theoretical Foundations

A summary of notation used in the paper is in Table 1. We use the Euclidean distance $d_E(\overrightarrow{x}, \overrightarrow{y})$, defined for two δ-dimensional vectors $\overrightarrow{x}, \overrightarrow{y}$ as the Euclidean norm of their difference:

$$d_E(\overrightarrow{x}, \overrightarrow{y}) = \|\overrightarrow{x} - \overrightarrow{y}\|.$$

The Mahalanobis distance $d_M(\overrightarrow{x}, \overrightarrow{y}, \mathbf{A})$ of vectors $\overrightarrow{x}$ and $\overrightarrow{y}$ is defined for any positive semi-definite matrix $\mathbf{A}$:

$$d_M(\overrightarrow{x}, \overrightarrow{y}, \mathbf{A}) = \sqrt{(\overrightarrow{x} - \overrightarrow{y})^T \mathbf{A} (\overrightarrow{x} - \overrightarrow{y})}.$$

Having all eigenvalues λ_j of $\mathbf{A}$, article [5] defines the lower bound on the Mahalanobis distance given by the Euclidean distance:

$$\frac{d_M(\overrightarrow{x}, \overrightarrow{y}, \mathbf{A})}{\sqrt{min_j(\lambda_j)}} \geq d_E(\overrightarrow{x}, \overrightarrow{y}) \tag{1}$$

Value $s_{M,E}(\mathbf{A}) = 1/\sqrt{\min_j\{\lambda_j\}}$ is the ***scaling factor*** *between the Mahalanobis and Euclidean distances*. Consequently, if the scaling factor $s_{M,E}(\mathbf{A})$ is 1, i.e., the smallest eigenvalue of matrix $\mathbf{A}$ is 1, then the Mahalanobis distance is lower-bounded by the Euclidean distance: $d_M(\overrightarrow{x}, \overrightarrow{y}, \mathbf{A}) \geq d_E(\overrightarrow{x}, \overrightarrow{y})$.

Having the domain D of δ-dimensional vectors, we define the goal of the personalised similarity search in a dataset $X \subseteq D$ as follows. Having a query vector $\overrightarrow{q} \in D$ and matrix $\mathbf{A}$, we want to evaluate a range query $Q(\overrightarrow{q}, d_M, r_M)$, i.e., to retrieve the *Mahalanobis answer set* $R_{d_M,r_M}(\overrightarrow{q})$:

$$R_{d_M,r_M}(\overrightarrow{q}) = \{\overrightarrow{x} \mid d_M(\overrightarrow{x}, \overrightarrow{q}, \mathbf{A}) \leq r_M\} \subseteq X \tag{2}$$

Due to the property given by Eq. 1, range r_E defined as:

$$r_E = r_M \cdot s_{M,E}(\mathbf{A}) \tag{3}$$

defines the range query $Q(\overrightarrow{q}, d_E, r_E)$ that provides the Euclidean answer set $R_{d_E,r_E}(\overrightarrow{q})$:

$$R_{d_E,r_E}(\overrightarrow{q}) = \{\overrightarrow{x} \mid d_E(\overrightarrow{x}, \overrightarrow{q}) \leq r_E\} \subseteq X \tag{4}$$

that contains the Mahalanobis answer set formalised by Eq. 2 as its subset.

Notice that r_E only depends on r_M and matrix $\mathbf{A}$ according to Eq. 3, i.e., is independent of a query vector $\overrightarrow{q}$. Therefore, besides range r_E, we define the *optimum range* $r_{\mathrm{Opt}E}(\overrightarrow{q})$ for a given $\overrightarrow{q}$ as the minimum Euclidean range such that the Euclidean answer set $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$ contains Mahalanobis answer set $R_{d_M,r_M}(\overrightarrow{q})$, i.e.:

$$\begin{aligned} R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q}) = \{\overrightarrow{x} \mid d_E(\overrightarrow{x}, \overrightarrow{q}) \leq r_{\mathrm{Opt}E}(\overrightarrow{q})\} \supseteq R_{d_M,r_M}(\overrightarrow{q}) \\ \wedge \quad \forall r < r_{\mathrm{Opt}E}(\overrightarrow{q}) : R_{d_E,r}(\overrightarrow{q}) \not\supseteq R_{d_M,r_M}(\overrightarrow{q}) \end{aligned} \tag{5}$$

Then, the Euclidean range r_E expresses the maximum $r_{\mathrm{Opt}E}(\overrightarrow{q})$ over all possible δ-dimensional vectors $\overrightarrow{q}$ [5,10]. Finally, we emphasise that:

- for any $\overrightarrow{q} \in D$: $R_{d_M,r_M}(\overrightarrow{q}) \subseteq R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q}) \subseteq R_{d_E,r_E}(\overrightarrow{q})$,
- range $r_{\mathrm{Opt}E}(\overrightarrow{q})$ depends on $\overrightarrow{q}$, and inherently on r_M and $\mathbf{A}$ (see Eq. 5),
- r_E only depends on r_M and $s_{M,E}(\mathbf{A})$ (see Eq. 3),
- different matrices $\mathbf{A}$ can have the same scaling factor $s_{M,E}(\mathbf{A})$ as it is given just by the smallest eigenvalue of $\mathbf{A}$.

The first bullet implies the recalls 1 of the Euclidean answer sets $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$ and $R_{d_E,r_E}(\overrightarrow{q})$ concerning the Mahalanobis answer set $R_{d_M,r_M}(\overrightarrow{q})$.

The *precision* of each of the Euclidean answer sets is consequently defined:

$$\mathcal{P}_E = \frac{|R_{d_M,r_M}(\overrightarrow{q})|}{|R_{d_E,r_E}(\overrightarrow{q})|} \tag{6}$$

and

$$\mathcal{P}_{\mathrm{Opt}E} = \frac{|R_{d_M,r_M}(\overrightarrow{q})|}{|R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})|} \tag{7}$$

Notice they differ just in the search ranges r_E and $r_{\mathrm{Opt}E}(\overrightarrow{q})$.

5 Concept Verification

In this section, we experimentally verify the validity of theory from Sect. 4 by investigating the relations between ranges r_M, $r_{\mathrm{Opt}E}(\overrightarrow{q})$, and r_E. We also investigate the precisions $\mathcal{P}_{\mathrm{Opt}E}$ and $\mathcal{P}_E$ of the search with ranges $r_{\mathrm{Opt}E}(\overrightarrow{q})$ and r_E, respectively, to reveal whether the usage of these ranges leads to small answer sets $R_{d_E,r_E}(\overrightarrow{q})$ and $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$ which are necessary for efficient refinement with the Mahalanobis distance function d_M.

Dataset We use a set of 226,778 images that were randomly selected from the *Profiset* image collection [4]. These images are represented by 768-dimensional vector embeddings generated by the CLIP ViT-L/14 model [18]. The vectors are normalised to unit vectors and form the searched dataset X.

Learning Personalized Matrices In this article, we address the similarity search purely on a geometric level, i.e., we abstract from the semantics of original images. Conversely, our goal is to investigate the relations of the Euclidean search ranges r_E (Eq. 3), $r_{\mathrm{Opt}E}(\overrightarrow{q})$ (Eq. 5), and Mahalanobis search range r_M. Specifically, the contributions of the indexing with Euclidean distances to the searching with the Mahalanobis distances are proportional to the size of the answer set $R_{d_E,r_E}(\overrightarrow{q})$, or possibly $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$ – the smaller, the better. Besides the number of vectors in the Euclidean answer sets, we also report their precisions $\mathcal{P}_E$ and $\mathcal{P}_{\mathrm{Opt}E}$.

The focus on a geometric level of image representations allows us to skip user relevance feedback and simulate it with various matrices **A** learnt for the Mahalanobis distance function. We learn matrices **A** with two *metric learning* techniques, namely *Information Theoretic Metric Learning (ITML)* [7] and *Sparse Distance Metric Learning (SDML)* [16]. Both techniques to learn **A** use pairs of vectors that are marked either as *similar* or *dissimilar*. The learning process aims to minimise the Mahalanobis distances of similar vectors and maximise the distances of dissimilar vectors at the same time.

We get the similar and dissimilar pairs of vectors in two different ways denoted as *random* and *nearest*. The random selection means that we work with n vector pairs selected in random from the dataset. In the nearest selection, we control a mutual closeness of vectors $(\overrightarrow{x}, \overrightarrow{y})$ in n pairs: vector $\overrightarrow{x}$ is always selected at random, but $\overrightarrow{y}$ is chosen randomly from a set of 1000 nearest neighbours of a given $\overrightarrow{x}$ from the dataset, as defined by Euclidean distances. In both cases, we randomly assess whether vectors in each pair are mutually similar or dissimilar. Consequently, approximately half of the pairs (i.e., $n/2$) are denoted as similar.

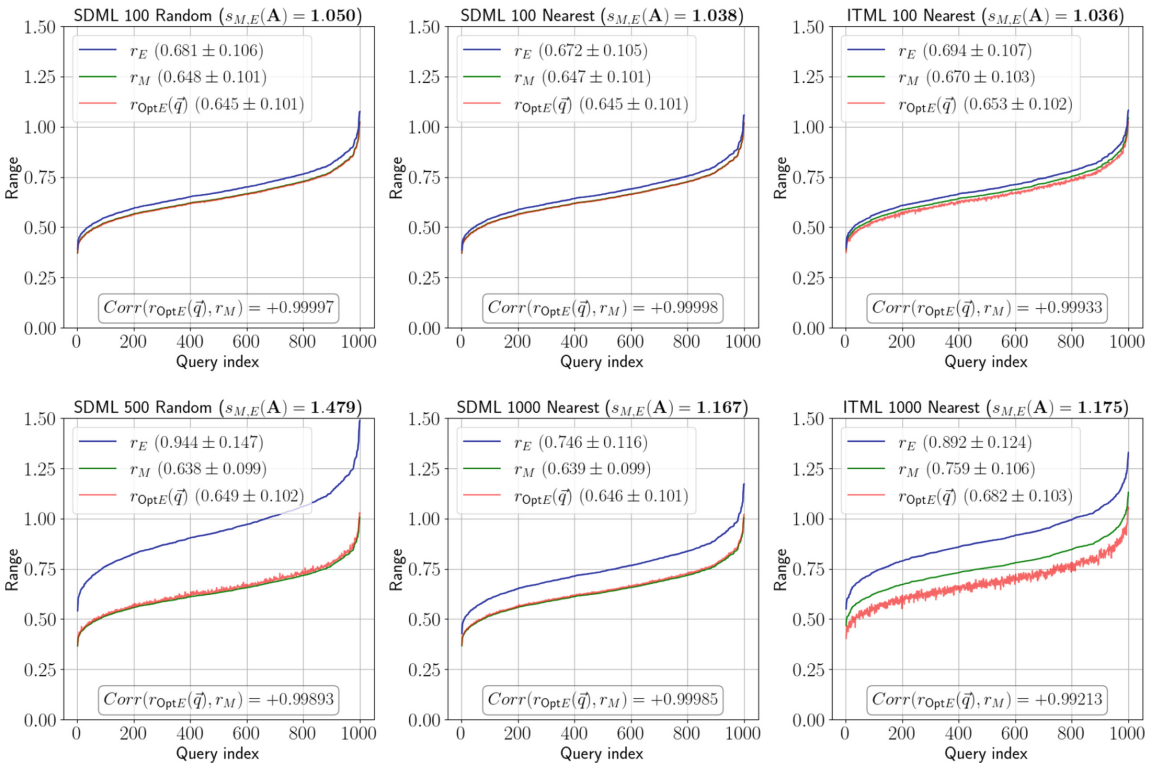


Fig. 2. Ranges r_M, $r_{\mathrm{OptE}}(\overrightarrow{q})$, and r_E, their averages and standard deviations for six learned matrices **A**. The *x-axis* expresses query indices – query vectors $\overrightarrow{q}$ are sorted to make r_M (i.e., the green curve) non-decreasing. Scaling factors $s_{M,E}(\mathbf{A})$ are at the top of each plot, and the Pearson correlations $Corr(r_{\mathrm{OptE}}(\overrightarrow{q}), r_M)$ are at the bottom.

We have investigated about 30 matrices $\mathbf{A}$ and picked 6 representatives for a presentation in the article. All the results we have observed are consistent with the presented findings. The matrices are denoted "<method> n <selection>" where <method> is either *SDML* or *ITML*, n is the number of vector pairs used in $\mathbf{A}$ learning, and <selection> denotes the way how the vector pairs were sampled, i.e., either *Random* or *Nearest*.

Matrices *SDML 100 Nearest* and *ITML 100 Nearest* have almost the same scaling factor $s_{M,E}(\mathbf{A})$, and we demonstrate sizes of the corresponding Euclidean answer sets created as the super sets of the Mahalanobis answer sets with these $\mathbf{A}$. In total, we present results for 6 matrices where 3 matrices were learnt with 100 vector pairs (i.e., *SDML 100 Random*, *SDML 100 Nearest*, and *ITML 100 Nearest*), and the other 3 were learnt in the same way just with more vector pairs (*SDML 500 Random*, *SDML 1000 Nearest*, and *ITML 1000 Nearest*). This selection of matrices helps us to demonstrate the consequences of an increasing number of learning vector pairs. We conduct all experiments with 1000 query vectors selected randomly from the dataset as the query set Θ. These query vectors were not used to learn any $\mathbf{A}$.

Lower Bounding Relationship Verification For the following experiments, we need to choose the Mahalanobis search ranges r_M. We consider each $\overrightarrow{q} \in \Theta$ independently, and compute $d_M(\overrightarrow{q}, \overrightarrow{x}, \mathbf{A})$ for all $\overrightarrow{x} \in X$. For each $\overrightarrow{q}$, we define $r_M = d_M(\overrightarrow{q}, \overrightarrow{x_{100}}, \mathbf{A})$ which is the Mahalanobis distance of $\overrightarrow{q}$ to its 100th nearest neighbour $\overrightarrow{x_{100}}$ from X. These ranges r_M define the Mahalanobis answer sets $R_{d_M,r_M}(\overrightarrow{q})$ which each contains 100 vectors[1].

The following experiments verify the key result of the theoretical Sect. 4, i.e., the relation of the answer sets defined by Eqs. 2, 4, and 5:

$$R_{d_M,r_M}(\overrightarrow{q}) \subseteq R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q}) \subseteq R_{d_E,r_E}(\overrightarrow{q}) \tag{8}$$

We start with the Mahalanobis ranges r_M, and compute the optimum range $r_{\mathrm{Opt}E}(\overrightarrow{q})$ for each $\overrightarrow{q}$ as the maximum Euclidean distance $d_E(\overrightarrow{x}, \overrightarrow{q})$, $\overrightarrow{x} \in R_{d_M,r_M}(\overrightarrow{q})$. The optimum range $r_{\mathrm{Opt}E}(\overrightarrow{q})$ enables us to compute the Euclidean answer set $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$ to confirm that it is the superset of the Mahalanobis answer set $R_{d_M,r_M}(\overrightarrow{q})$, i.e.:

$$\forall \overrightarrow{q} \in \Theta \ : \ R_{d_M,r_M}(\overrightarrow{q}) \subseteq R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q}) \tag{9}$$

Therefore, the recall of $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$ concerning $R_{d_M,r_M}(\overrightarrow{q})$ is always 1.

We continue with computations of ranges r_E according to Eq. 3 and confirm that $r_{\mathrm{Opt}E}(\overrightarrow{q}) \leq r_E$ for each $\overrightarrow{q}$ and $\mathbf{A}$. This is the result consistent with Eq. 5. For each $\overrightarrow{q} \in \Theta$, both ranges $r_{\mathrm{Opt}E}(\overrightarrow{q})$ and r_E apply to the same Euclidean space. The smaller search range thus implies a smaller answer set, formally:

$$\forall \overrightarrow{q} \in \Theta \ : \ R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q}) \subseteq R_{d_E,r_E}(\overrightarrow{q}) \tag{10}$$

[1] We did not observe any equality $d_M(\overrightarrow{q}, \overrightarrow{x_{100}}, \mathbf{A}) = d_M(\overrightarrow{q}, \overrightarrow{x_{101}}, \mathbf{A})$ for any $\overrightarrow{q} \in \Theta$.

Eqs. 9 and 10 are thus witnesses of Eq. 8, which we want to demonstrate. Finally, the recall of $R_{d_E,r_E}(\overrightarrow{q})$ concerning $R_{d_M,r_M}(\overrightarrow{q})$ is also always 1, as a consequence of the transitivity of Eq. 9 and Eq. 10.

We have conducted all these experiments with all 30 matrices $\mathbf{A}$ and a query set Θ with 1,000 vectors. All the results have been consistent with the claims above. We report in detail the results for 6 selected matrices $\mathbf{A}$ in Fig. 2 and Table 2. Curves in Fig. 2 illustrate the ranges r_E (in blue), r_M (in green), and $r_{\mathrm{Opt}E}(\overrightarrow{q})$ (in red). Each plot is made for a different matrix $\mathbf{A}$, and the scaling factors $s_{M,E}(\mathbf{A})$ are presented at the top of each plot. The values of the ranges are on the y-axis, and the x-axis depicts the index of the query vector $\overrightarrow{q}$. The queries are ordered to make ranges r_M (i.e., the green curves) non-decreasing.

Table 2. Scaling factors $s_{M,E}(\mathbf{A})$; Averages of precisions $\mathcal{P}_E, \mathcal{P}_{\mathrm{Opt}E}$; Averages of retrieved vector counts $\#_E$ and $\#_{\mathrm{Opt}E}$. Dataset X contains 226,778 vectors.

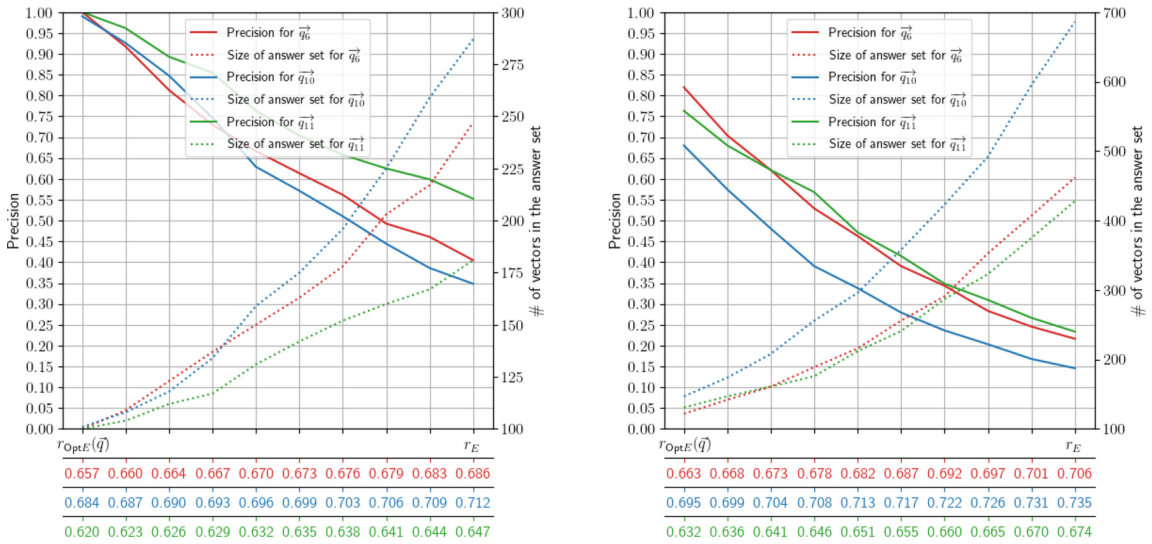
Learned matrix $\mathbf{A}$	$s_{M,E}(\mathbf{A})$	Avg. $\mathcal{P}_E$	Avg. $\#_E$	Avg. $\mathcal{P}_{\mathrm{Opt}E}$	Avg. $\#_{\mathrm{Opt}E}$
SDML 100 Random	1.050	0.410	317 ± 263	0.982	102 ± 2
SDML 500 Random	1.479	0.008	$108{,}025 \pm 82{,}569$	0.887	115 ± 23
SDML 100 Nearest	1.038	0.506	228 ± 110	0.990	101 ± 2
SDML 1000 Nearest	1.167	0.120	$4{,}110 \pm 10{,}828$	0.965	104 ± 5
ITML 100 Nearest	1.036	0.310	497 ± 528	0.804	128 ± 23
ITML 1000 Nearest	1.175	0.012	$70{,}681 \pm 69{,}508$	0.408	321 ± 235

Besides theoretically justified results, mainly $r_{\mathrm{Opt}E}(\overrightarrow{q}) \leq r_E$, Fig. 2 compares ranges r_M (green curves) and $r_{\mathrm{Opt}E}(\overrightarrow{q})$ (red curves). In all plots, the ranges are strongly correlated – see the Pearson correlations $Corr(r_{\mathrm{Opt}E}(\overrightarrow{q}), r_M)$ at the bottom of each plot in Fig. 2 that are from +0.992 to +0.99998. This is a very promising observation that should be further addressed to try to efficiently estimate the optimum search range $r_{\mathrm{Opt}E}(\overrightarrow{q})$ from r_M.

Precisions and Sizes of Euclidean Answer Sets We investigate the precisions $\mathcal{P}_{\mathrm{Opt}E}$ and $\mathcal{P}_E$ (defined by Eq. 7 and Eq. 6), as well as the sizes of Euclidean answer sets $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$ and $R_{d_E,r_E}(\overrightarrow{q})$ (Eq. 5 and Eq. 4) to reveal the viability of the proposal. Contributions of the indexing with Euclidean distances to the searching with the Mahalanobis distances are proportional to the size of the answer set $R_{d_E,r_E}(\overrightarrow{q})$, or possibly $R_{d_E,r_{\mathrm{Opt}E}(\overrightarrow{q})}(\overrightarrow{q})$. Specifically, at least the optimum range $r_{\mathrm{Opt}E}(\overrightarrow{q})$ should provide a sufficiently small answer set for each $\overrightarrow{q}$ to make the proposal viable. If also answer sets $R_{d_E,r_E}(\overrightarrow{q})$ are small enough, then even better, as the search range r_E is directly given by the Mahalanobis range r_M by Eq. 3.

For each matrix $\mathbf{A}$ and each query vector $\overrightarrow{q} \in \Theta$, we calculate $\mathcal{P}_E$, $\mathcal{P}_{\mathrm{Opt}E}$, and numbers of vectors in corresponding Euclidean answer sets denoted as $\#_E$ and $\#_{\mathrm{Opt}E}$. We remind that each Mahalanobis answer set $R_{d_M,r_M}(\overrightarrow{q})$ contains

100 vectors. Averages of the measured values over query vectors $\overrightarrow{q} \in \Theta$ are in Table 2. Specifically, there are: (1) the scaling factors $s_{M,E}(\mathbf{A})$, (2) the average precisions $\mathcal{P}_E$ and $\mathcal{P}_{\text{Opt}E}$, and (3) average numbers of vectors $\#_E$ and $\#_{\text{Opt}E}$ with standard deviations, all for 6 matrices $\mathbf{A}$. We only discuss the sizes of the retrieved answer sets $\#_{\text{Opt}E}$ and $\#_E$ in detail because the precisions are related to them. We also remind the dataset size of 226,778 vectors.



(a) SDML 100 Nearest, $s_{M,E}(\mathbf{A}) = 1.038$ (b) ITML 100 Nearest, $s_{M,E}(\mathbf{A}) = 1.036$

Fig. 3. Precision of range queries with three selected query vectors $\overrightarrow{q}$ (correspond to colours) evaluated with two matrices $\mathbf{A}$ (correspond to sub-figures). The x-axis depicts the search ranges r that cover the interval $[r_{\text{Opt}E}(\overrightarrow{q}), r_E]$ for each $\overrightarrow{q}$. The primary y-axis expresses the precision of range queries $Q(\overrightarrow{q}, d_E, r)$ and applies to the solid lines. The secondary y-axis depicts the number of vectors in the Euclidean answer set $R_{d_E,r}(\overrightarrow{q})$ and applies to the dashed lines.

Search ranges r_E provide answer sets with just hundreds of vectors in case of all matrices learned with 100 vector pairs, i.e., *SDML 100 Random*, *SDML 100 Nearest* and *ITML 100 Nearest* – see the fourth column in Table 2. While these are very promising results, experiments with matrices learnt with more vector pairs suggest a very quick growth of the answer set $R_{d_E,r_E}(\overrightarrow{q})$ – see all other matrices in the fourth column in Table 2. We thus conclude that the conditions leading to small answer sets $R_{d_E,r_E}(\overrightarrow{q})$ should be properly investigated since, in some cases, an index built with Euclidean distances and queried with range r_E provides small answer sets $R_{d_E,r_E}(\overrightarrow{q})$ that can be efficiently refined even with an expensive Mahalanobis distance function. But sometimes, Euclidean search with range r_E returns by orders of magnitude larger answer set than desired – see the matrices learned with 1000 or 500 vector pairs in Table 2.

Optimum search range $r_{\text{Opt}E}(\overrightarrow{q})$ leads to the retrieval of just slightly more than 100 candidates that need to be refined by expensive Mahalanobis distances, especially when the *SDML* technique to learn $\mathbf{A}$ is used (see the last column in

Table 2). While range $r_{\mathrm{Opt}E}(\overrightarrow{q})$ clearly leads to smaller answer sets than r_E, it cannot be directly derived, so its efficient estimation from r_M forms a challenge for future work. Overall, Table 2 illustrates the very good viability of the proposal and reveals the future challenges.

Figure 3 illustrates the search ranges for three representative query vectors $\overrightarrow{q} \in \Theta$ depicted in three different colours. Curves show how for each given $\overrightarrow{q}$ and an increasing Euclidean search range from $r_{\mathrm{Opt}E}(\overrightarrow{q})$ to r_E decreases the precision of the Euclidean answer set from $\mathcal{P}_{\mathrm{Opt}E}$ to $\mathcal{P}_E$. Correspondingly, the number of vectors in the Euclidean answer set increases. Figure 3a and 3b differ only in the matrix $\mathbf{A}$. Specifically, they depict the results for the same triplet of query vectors $\overrightarrow{q}$. Even though both examined matrices, *SDML 100 Nearest* in Fig. 3a and *ITML 100 Nearest* in Fig. 3b have nearly the same scaling factors $s_{M,E}(\mathbf{A})$ (1.038 and 1.036, respectively), the sub-figures demonstrate a different influence of matrices on different query vectors $\overrightarrow{q}$.

6 Final Discussion and Future Research Objectives

The experiments reveal the following facts:

- The theory is valid and the Euclidean search with ranges $r_{\mathrm{Opt}E}(\overrightarrow{q})$ and r_E always results in recall 1 with respect to the Mahalanobis search with range r_M. Both ranges can be effectively used for retrieving the superset of the Mahalanobis answer set $R_{d_M,r_M}(\overrightarrow{q})$, which can be further refined using Mahalanobis distance d_M.
- Range $r_{\mathrm{Opt}E}(\overrightarrow{q})$ is usually much smaller than r_E and the recall is still 1.
- Ranges $r_{\mathrm{Opt}E}(\overrightarrow{q})$ and r_M are surprisingly strongly correlated in our experiments; specifically, we observed the Pearson correlations around +0.999 in cases of 5 matrices out of 6 examined, and +0.992 for the last one.
- Experiments suggest that for a given matrix learning $< method >$ and $< selection >$, the precisions $\mathcal{P}_E$ and $\mathcal{P}_{\mathrm{Opt}E}$ decrease with increasing number of vector pairs n used for the $\mathbf{A}$ learning, which also increases the scaling factor $s_{M,E}(\mathbf{A})$.
- Given two different matrices $\mathbf{A}$ with nearly the same scaling factor $s_{M,E}(\mathbf{A})$, the precision $\mathcal{P}_E$ can be distinct even for the same query vector $\overrightarrow{q}$ with the search range r_E derived from r_M according to Eq. 3.
- Precision $\mathcal{P}_{\mathrm{Opt}E}$ is usually much higher than precision $\mathcal{P}_E$ because $r_{\mathrm{Opt}E}(\overrightarrow{q})$ is usually much lower than range r_E even when the scaling factor $s_{M,E}(\mathbf{A})$ is constant.
- Scaling factor $s_{M,E}(\mathbf{A})$ much depends on the way how the matrix $\mathbf{A}$ for the Mahalanobis distance function is learnt.

To develop a personalised similarity search engine, the following challenges need to be properly explored:

- The search range r_E guaranteed by Eq. 3 to cover the Mahalanobis answer set for given matrix $\mathbf{A}$ is usually too large. Given an extreme Pearson correlation

between Mahalanobis ranges r_M and optimum Euclidean ranges $r_{\mathrm{Opt}E}(\overrightarrow{q})$, a heuristic to estimate $r_{\mathrm{Opt}E}(\overrightarrow{q})$ from r_M should be proposed. The heuristic could be *approximate*, i.e., without guarantees on its correctness.
- We need theories and paradigms which would lead to a proper definition of the personalised matrix $\mathbf{A}$. This metric learning should be fast, simple, but effective from the user's point of view. At the same time, it should produce matrices $\mathbf{A}$ with a small scaling factor $s_{M,E}(\mathbf{A})$ as they still can lead to small Euclidean answer sets $R_{d_E,r_E}(\overrightarrow{q})$.
- Range r_E is linearly dependent on the scaling factor $s_{M,E}(\mathbf{A})$, but the number of vectors in the Euclidean answer set $R_{d_E,r_E}(\overrightarrow{q})$ is not. In practice, the k-NN queries are preferred over the range queries because they do not require knowledge of proper search ranges. Estimating the proper extension of the k value to efficiently execute the k-NN queries according to the schema proposed in this article is thus challenging. Moreover, also the position of a specific query vector $\overrightarrow{q}$ should be considered [1,11].
- Real-life applications deal with divergence and plenty of subjective similarity perceptions as well as with the diversity of searched datasets. This opens a space for different search engine architectures and implementations. Performance models of such systems would allow for convenient decisions that would best suit the specifics of applications.

7 Conclusion

Traditional research in similarity searching follows the effectiveness or efficiency complementary objectives, possibly both. To respect the subjectivity and individual views of users, we have elaborated on the personalised similarity search engine for vector databases. We have formalised a paradigm and tested the feasibility on a real-life dataset. Encouraging results indicate research directions that must be explored to create a truly usable personalised search system.

References

1. Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M.E., Kawarabayashi, K.i., Nett, M.: Estimating local intrinsic dimensionality. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 29–38. KDD '15 (2015)
2. Bartolini, I., Ciaccia, P., Patella, M.: Adaptively browsing image databases with PIBE. Multim. Tools Appl. **31**(3), 269–286 (2006)
3. Bellet, A., Habrard, A., Sebban, M.: Metric Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning (2015)
4. Budíková, P., Batko, M., Zezula, P.: Evaluation platform for content-based image retrieval systems. In: Research and Advanced Technology for Digital Libraries—International Conferences on Theory and Practice of Digital Libraries, TPDL 2011, Germany. Proceedings. LNCS, vol. 6966, pp. 130–142 (2011)
5. Ciaccia, P., Patella, M.: Searching in metric spaces with user-defined and approximate distances. ACM Trans. Database Syst. **27**(4), 398–437 (2002)

6. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: 20th Annual Symposium on Computational Geometry (SCG), pp. 253–262 (2004)
7. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: Machine Learning, Proceedings of the 24th International Conference (ICML 2007), Oregon, USA. ACM International Conference Proceeding Series, vol. 227, pp. 209–216 (2007)
8. Dou, Z., Song, R., Wen, J.: A large-scale evaluation and analysis of personalized search strategies. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, 8–12 May 2007. pp. 581–590. ACM (2007)
9. Ge, T., He, K., Ke, Q., Sun, J.: Optimized product quantization. IEEE Trans. Pattern Anal. Mach. Intell. **36**(4), 744–755 (2014)
10. Hafner, J.L., Sawhney, H.S., Equitz, W., Flickner, M., Niblack, W.: Efficient color histogram indexing for quadratic form distance functions. IEEE Trans. Pattern Anal. Mach. Intell. **17**(7), 729–736 (1995)
11. Houle, M.E.: Dimensionality, discriminability, density and distance distributions. In: 13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, TX, USA, 7-10 Dec 2013, pp. 468–473 (2013)
12. Kelly, D., Teevan, J.: Implicit feedback for inferring user preference: a bibliography. SIGIR Forum **37**(2), 18–28 (2003)
13. Krenková, M., Mic, V., Zezula, P.: Similarity search with the distance density model. In: Similarity Search and Applications—15th International Conference, SISAP 2022, Bologna, Italy, 5-7 Oct 2022, Proceedings. Lecture Notes in Computer Science, vol. 13590, pp. 118–132 (2022)
14. Ma, Z., Dou, Z., Bian, G., Wen, J.: PSTIE: time information enhanced personalized search. In: CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, 19–23 Oct 2020. pp. 1075–1084. ACM (2020)
15. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. arXiv (1603.09320) (2018)
16. Qi, G., Tang, J., Zha, Z., Chua, T., Zhang, H.: An efficient sparse metric learning in high-dimensional space via l_1-penalized log-determinant regularization. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Canada. ACM International Conference Proceeding Series, vol. 382, pp. 841–848 (2009)
17. Qiu, F., Cho, J.: Automatic identification of user interest for personalized search. In: Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, 23-26 May 2006, pp. 727–736 (2006)
18. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: Proceedings of the 38th International Conference on Machine Learning, ICML 2021, Virtual Event. Proceedings of Machine Learning Research, vol. 139, pp. 8748–8763 (2021)
19. Salton, G., Buckley, C.: Improving retrieval performance by relevance feedback. J. Am. Soc. Inf. Sci. **41**(4), 288–297 (1990)
20. Shen, X., Tan, B., Zhai, C.: Implicit user modeling for personalized search. In: Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, 31 Oct–5 Nov 2005, pp. 824–831. ACM (2005)

21. Speretta, M., Gauch, S.: Personalized search based on user search histories. In: 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005), Sept 19–22 2005, Compiegne, France, pp. 622–628 (2005)
22. Sugiyama, K., Hatano, K., Yoshikawa, M.: Adaptive web search based on user profile constructed without any effort from users. In: Proceedings of the 13th International Conference on World Wide Web, WWW 2004, New York, NY, USA, 17-20 May 2004, pp. 675–684. ACM (2004)
23. Teevan, J., Morris, M.R., Bush, S.: Discovering and using groups to improve personalized search. In: Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, 9–11 Feb 2009, pp. 15–24 (2009)
24. Tversky, A.: Features of similarity. Psychol. Rev. **84**(4), 327–352 (1977)
25. White, R.W., Chu, W., Awadallah, A.H., He, X., Song, Y., Wang, H.: Enhancing personalized search by mining and modeling task behavior. In: 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, 13–17 May 2013, pp. 1411–1420 (2013)
26. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity search—the metric space approach. Adv. Database Syst. **32** (2006)
27. Zhou, Y., Dou, Z., Zhu, Y., Wen, J.: PSSL: self-supervised learning for personalized search with contrastive sampling. In: CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia 1–5 Nov 2021. pp. 2749–2758 (2021)

B Appendix

Matúš Šikyňa and Pavel Zezula. "Integrating Relevance Feedback for Effective Personalisation in Vector Search". In: *18th International Conference on Similarity Search and Applications*. SISAP 2025, Reykjavik, Iceland, pp. 154–162.

Integrating Relevance Feedback for Effective Personalisation in Vector Search

Matúš Šikyňa( ) and Pavel Zezula

Faculty of Informatics, Masaryk University, Brno, Czech Republic
{xsikyna,zezula}@fi.muni.cz

Abstract. Vector search systems typically rely on fixed indexing methods and standard similarity measures, such as Euclidean distance, assuming a universal notion of semantic similarity. This assumption overlooks the subjective and context-dependent nature of human similarity perception. Prior approaches proposed personalised querying via Mahalanobis distance parameterised by learned user-profile matrices, but did not address strategies for acquiring real user relevance feedback. We present an end-to-end pipeline designed to integrate semantic feedback directly from user interactions. We explore three feedback strategies, evaluating them across ten established metric learning models. Through an empirical analysis, we assess how these strategies affect key performance indicators, such as retrieval precision. We also outline three practical operational scenarios – balancing precision and computational costs, prioritising computational efficiency, and optimising for precision – and provide concrete model and system configuration recommendations. Experiments show that the recommended setups make search efficient while improving retrieval precision, showing the practicality of the approach.

Keywords: Personalised search · Relevance feedback · Vector databases

1 Introduction

Modern vector databases use high-dimensional vector embeddings to capture semantics for retrieval and recommendation. To accelerate queries, these embeddings are stored in fixed-index structures with standard metrics (e.g., Euclidean distance). However, this static approach imposes a one-size-fits-all notion of similarity, overlooking its inherently subjective, context-dependent nature [13].

Addressing this challenge, Mahrík et al. [8] proposed a vector search architecture that stores all embeddings in a Euclidean index but personalises queries with a Mahalanobis metric defined by a learned user-profile matrix, exploiting the Euclidean–Mahalanobis lower bounding relationship. Because their profile matrices were trained with synthetic feedback, the method's real-world efficacy remains untested. Building on this foundation, we make two contributions:

G. Amato et al. (Eds.): SISAP 2025, LNCS 16134, pp. 154–162, 2026.
https://doi.org/10.1007/978-3-032-06069-3_13

– We present a pipeline that includes semantic feedback from user interactions with vector search system, define three feedback strategies, and evaluate their impact on multiple metrics across ten widely used metric learning models.
– Based on empirical analysis and defined operational scenarios, we offer practical recommendations for configuring the semantic feedback strategies and metric learning models in personalised vector search systems, guiding practitioners in selecting parameter settings tailored to their operational priorities.

2 Background and Related Work

2.1 Lower Bounding Relationship

Mahrík et al. [8] proposed a range query search engine that leverages the well-known lower bounding relationship, stating that for any two vectors x and y, and a positive semi-definite (PSD) matrix $\mathbf{A}$, the following relationship holds:

$$\frac{d_M(x, y, \mathbf{A})}{\sqrt{\min_j(\lambda_j)}} \geq d_E(x, y), \tag{1}$$

where $d_E(x, y) = \|x - y\|$ represents the Euclidean distance. The Mahalanobis distance $d_M(x, y, \mathbf{A})$ is expressed as: $d_M(x, y, \mathbf{A}) = \sqrt{(x-y)^T \mathbf{A}(x-y)}$, and the eigenvalues λ_j are derived from the matrix $\mathbf{A}$. This inequality introduces a scaling factor $s_{M,E}(\mathbf{A}) = 1/\sqrt{\min_j\{\lambda_j\}}$. In this paper, we use the term Mahalanobis distance to denote a general quadratic form distance parameterised by a learned PSD matrix, as commonly done in metric learning literature [5–7,16].

Given a specified Mahalanobis radius r_M, the corresponding Mahalanobis answer set for any query vector q and all z from the dataset is defined as:

$$R_{d_M, r_M}(q) = \{z \mid d_M(z, q, \mathbf{A}) \leq r_M\}. \tag{2}$$

Utilising the lower bounding relationship (Eq. 1), a corresponding Euclidean radius r_E can be computed as $r_E = r_M \cdot s_{M,E}(\mathbf{A})$, leading to an alternative, larger Euclidean answer set using the Euclidean distance:

$$R_{d_E, r_E}(q) = \{z \mid d_E(z, q) \leq r_E\}. \tag{3}$$

Mahrík et al. [8] experimentally demonstrated that the Euclidean answer set defined by Eq. (3) inherently includes the Mahalanobis answer set defined by Eq. (2). The search adopts a two-stage *filter and refine* approach: Euclidean distance more efficiently retrieves candidates, which are then refined by applying the more precise yet costlier Mahalanobis distance, parametrised by the user-specific matrix $\mathbf{A}$. Clearly, an increase of the scaling factor $s_{M,E}(\mathbf{A})$ directly corresponds to higher computational costs during query evaluation.

2.2 Personalisation Approaches in Search Systems

Personalised search systems utilise implicit and explicit user feedback to adapt search results to individual users' preferences. Implicit feedback, collected passively as users interact with the system, includes: clicks on results and submitted searched queries [15], or usage behaviour characteristics such as dwell time [18]. In contrast, explicit feedback is provided directly by users. One common approach is to provide positive relevance judgments, as in the case of PicHunter [3]. Vadicamo et al. [14] enhanced this approaches by utilising negative relevance judgments. To adjust the search relevance to users, search systems employ variety of personalisation techniques. Common approaches include query expansion [20] or result adaptation [12]. Furthermore, some approaches learn the similarity function using metric learning models [2,6,16].

3 Our Approach to Semantic User Feedback

In this section, we introduce an end-to-end pipeline that captures semantic user feedback and integrates it into a personalised vector search system, define three relevance feedback strategies used for learning the user-profile matrix, and elaborate on various operational scenarios tailored to different operational objectives.

3.1 Workflow of the Personalised Vector Search Pipeline

The end-to-end pipeline is a cyclic sequence of user actions and system operations, illustrated in Fig. 1. Each loop continuously refines the search space in response to explicit user relevance feedback until the scaling factor is too large.

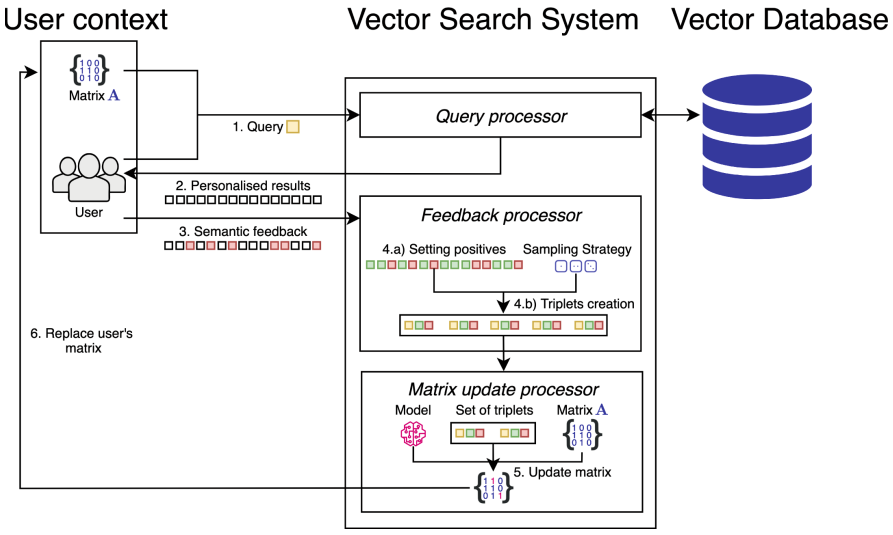


Fig. 1. The workflow of the pipeline. The query appears in yellow, positive results are in green, and negative results are in red boxes. (Color figure online)

The step-by-step workflow proceeds as follows:

1. **User Query**: User submits a query, using their user-profile matrix **A**.
2. **Personalised Results Retrieval**: The system returns results based on the *filter and refine* approach it first runs a Euclidean range search with the radius enlarged by the scaling factor from **A**, then filters those candidates using the Mahalanobis metric parameterised by **A** before returning the results.
3. **User Semantic Feedback**: User selects irrelevant (negative) results.
4. **Feedback Processing**: Results not marked as irrelevant are set as relevant (positive). Using a selected feedback (sampling) strategy, the system arranges these *feedback judgements* into a set of triplets *(query, positive, negative)*.
5. **Matrix Update**: This set of triplets is used for the metric learning model training. The model updates the user-profile matrix **A**, aligning it with personalised semantic preferences captured during the recent feedback session.
6. **Personalised Querying Loop**: The updated user-profile matrix **A** is subsequently used for the future queries iterations.

3.2 Relevance Feedback Strategies

Here, we detail the *feedback processing* step – creating the set of triplets. We propose three feedback strategies for constructing such a set for a specific query:

- **Strategy 1**: each iteration draws randomly one positive and one negative result to form a single triplet; we run $k \in \{1, 2, 4, 8, 16, 32, 64\}$ such iterations, either with or without replacement.
- **Strategy 2**: a randomly chosen negative result (selected without replacement) is paired with every available positive result, producing a set of triplets that is fed to the model either as one whole batch or sequentially.
- **Strategy 3**: every negative result is paired with every positive result, but training is triggered only after feedback from $q \in \{1, 2, 5, 10, 20, 40\}$ queries has been accumulated; once the batch of q queries is complete, the aggregated triplets are used for a single update before the next batch begins.

3.3 Operational Scenarios

For the practical deployment, we present three operational scenarios – *Balanced Precision–Scaling Factor (BPSF)*, *Minimum Scaling Factor with Acceptable ΔMAP (MSAP)*, and *Maximum ΔMAP (MP)*. The ΔMAP is defined as the change in mean-average precision (MAP) obtained when the ranking metric is switched from Euclidean (baseline) to Mahalanobis distance parameterised by the learned user-profile matrix **A** in the ranking function; **A** is first learned using the full query set, and ΔMAP is then evaluated over that same set of queries.

The *BPSF* scenario balances between a low scaling factor and high retrieval precision. *MSAP* focuses on minimising the scaling factor to maximise efficiency, while still enforcing an acceptable precision increase (ΔMAP is at least 0.1 – arbitrarily chosen). *MP* prioritises highest attainable ΔMAP, resulting in maximum personalisation. Each scenario has three variants: (i) *Baseline (BAS)* variant, (ii) *Low Learning Time (LLT)* variant that minimises model learning time, and (iii) *Rank Stability (RS)* variant that maximises rank correlation coefficients.

4 Experimental Evaluation

4.1 Setting

Dataset and Queries. We use the publicly available *Profiset* [1] database of around 20 million images, each encoded as a 768D CLIP ViT-L/14 [10] embedding normalised to unit length, supporting simultaneous text-image similarity search. We created 280 textual queries. For each, the 20 most similar images were retrieved via an image-text search engine [9] and manually labelled relevant or irrelevant, yielding the ground truth for learning user-profile matrices. The query set spans seven thematic categories representing multiple semantic layers – from concrete entities through abstract notions, dynamic events and static scenes.

Metric Learning Models. For learning the user-profile matrices, we re-implemented most of the well-known metric learning models: OASIS [2], ITML [5], POLA [11], AROMA [4], OMDML [16], MLOML [7], RobustODML [19], SORS [17], AdaSORS [17], and OPML [6]. Most of these models maintain PSD property of the matrix, although it is not needed for the personalisation purposes [2,17]. OMDML can be used with more modalities, but was adjusted to use one. We also performed an extensive hyperparameter grid search to find the best settings.

Evaluation Metrics. We define and use the following evaluation metrics:

- **Final Scaling Factor (FSF)** – the scaling factor $s_{M,E}(\mathbf{A})$ derived after learning the matrix $\mathbf{A}$ for all queries incrementally.
- **Precision Difference (ΔMAP)** – as defined in Sect. 3.3.
- **Average Learning Time (ALT)** – average time (in seconds) needed for integration of a single set of triplets in the metric learning model's learning.
- **Average Spearman@k (AS@k)** – the average Spearman rank correlation (across all queries) between two ranked lists – the top k results retrieved with Euclidean distance and the top k results retrieved with the learned Mahalanobis distance – computed after both lists are mapped onto the union of their items, assigning any item missing from a list the worst rank, which equals the size of the union. The tested k are: {20, 100, 1000}.
- **Average Kendall's Tau@k (AK@k)** – like AS@k, using Kendall's Tau.
- **Average Jaccard Sim@k (AJ@k)** – like AS@k, using Jaccard similarity.

For the operational scenarios (*BPSF*, *MSAP*, *MP*) and their variants (*BAS*, *LLT*, *RS*), we define additional measures. These measures were selected based on experimental analysis aimed at finding the suitable metrics and their associated constants with the goal of effectively capturing the distinct characteristics and operational objectives of the scenarios and variants. The measures:

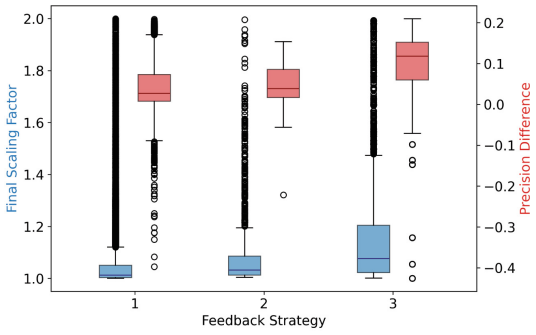
- **BPSF:**
  $\mathrm{BAS} = \Delta MAP/\mathrm{FSF}^2$, $\mathrm{LLT} = \Delta MAP/(\mathrm{FSF}^2 \cdot \mathrm{ALT})$,
  $\mathrm{RS} = \Delta MAP/[\mathrm{FSF}^2 \cdot \big(1 + 4 \cdot (1 - \mathrm{AS@1000}) + 4 \cdot (1 - \mathrm{AK@1000})\big)]$.

– **MSAP:** $\text{BAS} = \text{FSF}^{-8}$, $\text{LLT} = \text{FSF}^{-8}/\text{ALT}$,
$\text{RS} = \text{FSF}^{-8}/[1 + 4 \cdot (1 - \text{AS@1000}) + 4 \cdot (1 - \text{AK@1000})]$.
– **MP:** $\text{BAS} = \Delta MAP^3$, $\text{LLT} = \Delta MAP^3/\text{ALT}$,
$\text{RS} = \Delta MAP^3/[1 + 4 \cdot (1 - \text{AS@1000}) + 4 \cdot (1 - \text{AK@1000})]$.

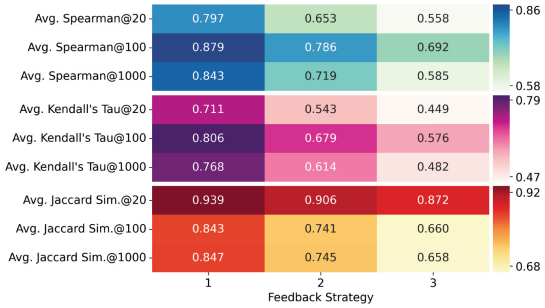
4.2 Results

We limit the analysis to models and hyperparameter settings resulting in Final Scaling Factor ≤ 2, as prior work [8] shows larger scaling factors degrade search efficiency. We then focus on a single query category as preliminary tests revealed that the other six categories yield similar results.

Feedback Strategies Comparison. We examine the comparison between the feedback strategies. Figure 2a shows the boxplots of all learned models (matrices) for the feedback strategies, while the Final Scaling Factor is on the main y-axis, and the Precision Difference is on the secondary y-axis. Figure 2b shows the heatmap of the means of Average Spearman, Kendall's Tau and Jaccard Similarity metrics at 20, 100, and 1000 across all learned models for each strategy.



(a) Scaling Factor and Precision



(b) Spearman, Kendall's Tau and Jaccard

Fig. 2. Comparison of the feedback strategies using Final Scaling Factor, Precision Difference, rank correlation coefficients and Jaccard Similarity.

The boxplots indicate higher Final Scaling Factors and Precision Differences for Feedback Strategies 2 and 3, peaking with Strategy 3. The heatmaps reveal the highest Average Spearman, Kendall's Tau and Jaccard Similarity are for Feedback Strategy 1, followed by Feedback Strategies 2 and 3. These observations imply that with a higher number of feedback inputs, the global ranking using the Mahalanobis distance for ordering deteriorates against the Euclidean distance ranking, while increasing scaling factors and precisions.

Finding Best Models for Operational Scenarios. For the utilisation of the approach in practice, we explore the best models (and their hyperparameters) and

Table 1. Calculated evaluation measures for the best model given the scenario and its variant. The Model column is in the form: model and hyperparameters. The Feedback column is in the form: feedback strategy and parameters – 1:(Number of pairs, Replacement), 2:(Batch), 3:(Number of queries).

Scenario–Variant	Model	Feedback	FSF	ΔMAP	ALT	AS@1000	AK@1000
BPSF-BAS	OMDML β: 1e-3, C: 4e-3, γ: 7e-3	2 True	1.095	0.203	1.373	0.708	0.532
BPSF-LLT	OASIS C: 7e-2, *enforce_psd*: False	1 1, False	1.147	0.103	0.001	0.621	0.456
BPSF-RS	OMDML β: 7e-2, C: 1e-3, γ: 1e-4	2 False	1.039	0.134	1.416	0.899	0.738
MSAP-BAS	MLOML *nl*: 2, γ: 1e-4, *act*: tanh, λ: 7e-6	3 5	1.017	0.102	18.098	0.863	0.693
MSAP-LLT	OASIS C: 1e-2, *enforce_psd*: False	1 8, False	1.062	0.106	0.008	0.813	0.636
MSAP-RS	OMDML β: 1.0, C: 7e-4, γ: 4e-4	1 64, True	1.028	0.101	1.768	0.939	0.798
MP-BAS	OMDML β: 7e-1, C: 1e-2, γ: 1e-2	2 False	1.148	0.211	0.712	0.566	0.411
MP-LLT	OASIS C: 7e-2, *enforce_psd*: False	1 2, True	1.271	0.130	0.003	0.308	0.211
MP-RS	OMDML β: 4e-2, C: 1e-2, γ: 1e-4	2 True	1.077	0.193	1.104	0.788	0.609

feedback strategies for all the operational scenarios and their variants. The results of the best models based on the metrics defined in Sect. 4.1 are in Table 1.

Table 1 shows two patterns: OMDML dominates when precision or rank stability matter, whereas OASIS excels when learning time is low, with MLOML surfacing only in the *MSAP BAS* variant. Final Scaling Factors are low, indicating effectiveness. Within the BPSF scenario, the *OMDML BAS* variant should be adopted by default; the *OASIS LLT* variant should be selected for fast model updates, and the *OMDML RS* variant should be chosen for rank stability. For an efficiency-focused MSAP scenario, the *OASIS LLT* variant should be preferred for minimal training overhead; if additional learning time can be allowed to gain a lower scaling factor and higher Spearman and Kendall's Tau, the *MLOML BAS* or *OMDML RS* configurations should be selected. When requiring the maximum precision – MP variants, the *OMDML BAS* gives the highest Precision Difference, the *OMDML RS* variant keeps ranks steadier, and the *OASIS LLT* provides faster learning but with a high scaling factor.

5 Conclusion

In this paper, we presented an end-to-end pipeline for integrating real semantic feedback into personalized vector search. We introduced three distinct feedback strategies, evaluated across ten established metric learning models. Through extensive experiments, we pinpointed optimal model–strategy configurations that balance precision gains (up to +21%) against computational cost (scaling factors $\leq$ 1.15). Our practical guidelines enable practitioners to quickly tailor vector search systems for efficiency, stability, or maximal relevance improvement.

Acknowledgments. This work was supported by the Open Calls for Security Research 2023–2029 (OPSEC) program granted by the Ministry of the Interior of the Czech Republic under No. VK01010147 – Automated digital data forensics lab for complex crime detection. Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Budikova, P., Batko, M., Zezula, P.: Evaluation platform for content-based image retrieval systems. In: Gradmann, S., Borri, F., Meghini, C., Schuldt, H. (eds.) TPDL 2011. LNCS, vol. 6966, pp. 130–142. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24469-8_15
2. Chechik, G., Shalit, U., Sharma, V., Bengio, S.: An online algorithm for large scale image similarity learning. In: NeurIPS 2009, vol. 22, pp. 306–314 (2009)
3. Cox, I.J., Miller, M.L., Minka, T.P., Papathomas, T.V., Yianilos, P.N.: The bayesian image retrieval system, pichunter: theory, implementation, and psychophysical experiments. IEEE Trans. Image Process. **9**, 20–37 (2000)
4. Crammer, K., Chechik, G.: Adaptive regularization for weight matrices. arXiv preprint arXiv:1206.4639 (2012)
5. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: ICML 2007, vol. 227, pp. 209–216 (2007)
6. Li, W., Gao, Y., Wang, L., Zhou, L., Huo, J., Shi, Y.: Opml: a one-pass closed-form solution for online metric learning. Pattern Recogn. **75**, 302–314 (2018)
7. Li, W., et al.: A multilayer framework for online metric learning. IEEE Trans. Neural Netw. Learn. Syst. **34**(10), 6701–6713 (2023)
8. Mahrík, M., Šikyňa, M., Mic, V., Zezula, P.: Towards personalized similarity search for vector databases. In: SISAP 2024, pp. 126–139 (2024)
9. Novak, D., Batko, M., Zezula, P.: Large-scale image retrieval using neural net descriptors. In: ACM SIGIR 2015, pp. 1039–1040 (2015)
10. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: ICML 2021, vol. 139, pp. 8748–8763 (2021)
11. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: ICML 2004, vol. 69 (2004)
12. Stamou, S., Ntoulas, A.: Search personalization through query and page topical analysis. User Model. User-Adap. Inter. **19**(1–2), 5–33 (2009)
13. Tversky, A.: Features of similarity. Psychol. Rev. **84**(4), 327–352 (1977)
14. Vadicamo, L., Scotti, F., Dearle, A., Connor, R.: Comparative analysis of relevance feedback techniques for image retrieval. In: MMM 2025, pp. 206–219 (2025)
15. Vu, T., Willis, A., Kruschwitz, U., Song, D.: Personalised query suggestion for intranet search with temporal user profiling. In: CHIIR 2017, pp. 265–268 (2017)
16. Wu, P., Hoi, S.C.H., Zhao, P., Miao, C., Liu, Z.: Online multi-modal distance metric learning with application to image retrieval. IEEE Trans. Knowl. Data Eng. **28**(2), 454–467 (2016)
17. Yao, D., Zhao, P., Yu, C., Jin, H., Li, B.: Sparse online relative similarity learning. In: 2015 IEEE International Conference on Data Mining, pp. 529–538 (2015)

18. Yi, X., Hong, L., Zhong, E., Liu, N.N., Rajan, S.: Beyond clicks: dwell time for personalization. In: RecSys 2014, pp. 113–120 (2014)
19. Zabihzadeh, D., Tuama, A., Karami-Mollaee, A., Mousavirad, S.J.: Low-rank robust online distance/similarity learning based on the rescaled hinge loss. Appl. Intell. **53**(1), 634–657 (2023)
20. Zhang, X.: Improving personalised query reformulation with embeddings. J. Inf. Sci. **48**(4), 503–523 (2022)