OPEN ACCESS Freely available online

Research Article

Vol. 1, Issue 1, January to June 2026

Journal of Artificial Intelligence and Digital Health

Hybrid SE-FAISS: A Novel Semantic Embedding with Facebook Artificial Intelligence Similarity Search-Based Deep Learning Framework for Similar Text Detection

Bita Ghasemkhani[1*], Alper Ozdemir[1], Mesut Ekinci[1], Deniz Kilinc[2]

[1]*R&D Department, Probel Yazılım ve Bilişim Sistemleri A.Ş., Izmir, Turkey*
[2]*Department of Computer Engineering, Izmir University Bakircay, Izmir, Turkey*

[*]**Correspondence to:** Bita Ghasemkhani, R&D Department, Probel Yazılım ve Bilişim Sistemleri A.Ş., Izmir, Turkey, E-mail: bita.ghasemkhani@probel.com.tr

Received: May 04, 2026; **Manuscript No:** JAID-26-6304; **Editor Assigned:** May 07, 2026; **PreQc No:** JAID-26-6304(PQ); **Reviewed:** May 18, 2026; **Revised:** May 25, 2026; **Manuscript No:** JAID-26-6304(R); **Published:** June 18, 2026

Citation: Ghasemkhani B, Ozdemir A, Ekinci M, Kilinc D (2026). Hybrid SE-FAISS: A Novel Semantic Embedding with Facebook Artificial Intelligence Similarity Search-Based Deep Learning Framework for Similar Text Detection. J. Artif. Intell. Digit. Health. Vol.1 Iss.1, June (2026), pp:46-58.

ABSTRACT

Efficiently identifying similar textual information is a critical task in modern information technology service desk systems, where ticket volume and redundancy can impede timely responses. In this study, we propose Hybrid SE-FAISS, a deep learning–based framework that combines Semantic Embedding (SE) using a sentence transformer with Facebook Artificial Intelligence Similarity Search (FAISS) for fast and accurate detection of similar textual data. Service desk tickets are transformed into dense numerical vectors using a deep learning model to capture semantic relationships between tickets. These embeddings are indexed using FAISS, a high-performance similarity search technique, enabling rapid retrieval of the most semantically similar text data. The framework was evaluated on a real-world dataset of IT service tickets, including approximately 85,000 entries from healthcare project management systems provided by Probel Yazılım ve Bilişim Sistemleri AŞ, a software company in İzmir, Turkey. The model achieved 80% accuracy in detecting similar tickets based on cosine similarity between embedding vectors. Hybrid SE-FAISS demonstrates that integrating deep learning embeddings with optimized similarity search algorithms provides an effective and scalable solution for text similarity detection, with potential applications in ticket deduplication, automated response recommendation, and knowledge management systems. By rapidly identifying tickets similar to new inputs, the framework reduces response times and enables faster processing through reusable solutions.

Keywords: Artificial Intelligence; Machine Learning; Deep Learning; Text Similarity Detection; Semantic Embedding; Sentence Transformer; Facebook Artificial Intelligence Similarity Search; Service Desk Tickets; Information Technology Helpdesk

INTRODUCTION

Artificial intelligence (AI) has emerged as a transformative technology across diverse domains, enabling systems to perform tasks that traditionally require human intelligence [1]. Among its many subfields, machine learning (ML) allows computers to learn patterns from data, while deep learning (DL), a subset of machine learning, utilizes neural networks to model complex, high-dimensional relationships [2-3]. In particular, deep learning-based natural language processing (NLP) techniques have shown remarkable success in understanding and representing textual information [4]. These advances have made it possible to convert text data into numerical representations, known as embeddings, which capture the semantic meaning of sentences or documents [5]. Such representations form the foundation for automated text similarity detection, enabling systems to efficiently identify semantically related content in large datasets.

Deep learning models, particularly those designed for natural language processing, have confirmed an exceptional ability to capture the semantic meaning of textual data. Unlike traditional machine learning approaches that rely on manually engineered features, deep learning models automatically learn rich, high-dimensional demonstrations from raw text [6]. Among these models, sentence transformers have gained prominence for generating sentence-level embeddings that encode semantic relationships between texts. These embeddings allow systems to measure similarity between sentences or documents in a continuous vector space, facilitating efficient and accurate detection of related or duplicate content [7]. By leveraging such embeddings, organizations can significantly improve tasks such as information retrieval, document clustering, and automated ticket resolution in service desk systems.

Automated text similarity detection has become an essential tool for managing large volumes of textual data, enabling

systems to efficiently identify semantically related content. Central to this process are semantic embeddings, numerical vector representations of text that capture contextual and semantic meaning beyond mere keyword matching. By transforming sentences or documents into high-dimensional vectors, embeddings allow for precise comparison between texts using mathematical similarity measures, such as cosine similarity. This approach aids rapid detection of similar entries, reduces redundancy, and improves decision-making in applications ranging from ticket management to knowledge retrieval in enterprise systems [8].

Efficient retrieval of similar texts from large datasets requires more than just high-quality embeddings; it also demands optimized similarity search algorithms. Facebook artificial intelligence similarity search (FAISS) is a high-performance methodology designed to handle large-scale vector searches, enabling rapid identification of nearest neighbors among millions of embeddings [9]. By indexing embeddings and performing similarity searches using methods such as inner product or cosine similarity, FAISS meaningfully reduces query time compared to naive pairwise comparisons [10]. Integrating FAISS with deep learning–based embeddings offer a practical solution for applications where real-time or near-real-time text similarity detection is critical, such as in IT service desk systems managing thousands of incoming tickets.

Building upon the strengths of deep learning embeddings and FAISS, we propose Hybrid SE-FAISS, a novel framework implemented for efficient detection of similar text entries in IT service desk systems. In this approach, service desk tickets are first transformed into dense semantic embeddings using the sentence transformer model, catching the contextual meaning of each ticket. These embeddings are then indexed using FAISS, allowing rapid retrieval of the most semantically similar tickets when a new ticket is submitted. By leveraging this combination, SE-FAISS not only identifies redundant or related tickets quickly but also supports faster response times, automated knowledge retrieval, and potential ticket deduplication, making it highly suitable for healthcare project management systems with large volumes of textual data.

The remainder of this paper is organized as follows: Section 2 reviews related works on text similarity detection, deep learning embeddings, and large-scale similarity search. Section 3 details the dataset, preprocessing procedures, and the proposed Hybrid SE-FAISS framework, including semantic embedding generation and FAISS-based similarity retrieval. Section 4 presents the experimental setup, evaluation metrics, and results. Section 5 discusses the practical implications of the framework for IT service desk systems and knowledge management. Finally, Section 6 concludes the paper and highlights avenues for future research.

Related Works

The growing need to automatically identify semantically similar text has led to significant research across natural language processing, information retrieval, and deep learning. This section reviews the major methodological directions relevant to the proposed SE-FAISS framework. We first discuss traditional text similarity approaches that rely on lexical or statistical representations. Then, we examine their limitations, particularly in capturing semantic meaning and scaling to large datasets. Subsequently, we outline advancements in deep learning based embedding models and large-scale vector search methods, such as those enabled by FAISS. Finally, we highlight existing applications in helpdesk and enterprise ticketing systems and identify the research gaps that motivate our proposed deep learning-based similarity detection framework.

Early research on text similarity primarily relied on corpus-based linguistic and statistical techniques [11]. Methods such as the bag-of-words (BoW) model and term frequency inverse document frequency (TF-IDF) represent text as high-dimensional sparse vectors where each dimension corresponds to a word in the vocabulary [12-13]. Similarity between texts is typically computed using metrics such as cosine similarity, Jaccard similarity, or Euclidean distance, enabling comparisons based on shared vocabulary or word frequency patterns [14]. These corpus-based approaches formed the foundation of classical information retrieval and document clustering tasks due to their simplicity and computational efficiency. However, because they rely on lexical overlap and treat words independently, they fail to capture deeper semantic meaning or contextual relationships, limiting their effectiveness for tasks requiring semantic understanding.

In addition to classical corpus-based representations, early semantic methods such as latent semantic analysis (LSA) introduced the idea of capturing hidden relationships between words and documents. LSA applies singular value decomposition (SVD) to a term document matrix typically derived from TF-IDF to project text into a lower-dimensional latent semantic space. This dimensionality reduction reveals underlying semantic structure and helps address some limitations of purely frequency-based approaches. However, because LSA relies on linear algebra and global co-occurrence statistics, it struggles to model contextual nuances, polysemy, and non-linear semantic relationships [15]. As a result, LSA served as a transitional step between traditional statistical models and the emergence of more expressive neural embedding techniques.

Building on this transition, shallow embedding models introduced a major advancement by learning dense, low-dimensional vector representations of words [16]. Methods such as Word2Vec, GloVe, and fastText rely on local context windows or global co-occurrence statistics to capture semantic relationships between words. Unlike sparse representations, these models embed words in continuous vector spaces where semantic similarity corresponds to geometric proximity [17]. Although these embeddings significantly improved performance across many NLP tasks, they remain context-independent each word is associated with a single static vector regardless of its usage in different sentences. This limitation reduces their effectiveness for sentence-level or document-level similarity tasks, particularly when meaning depends on context or domain-specific phrasing.

Deep contextual embeddings marked a major breakthrough in text representation by enabling models to generate word vectors that dynamically depend on the surrounding context. Early models, such as embeddings from language models (ELMo),

introduced this paradigm by leveraging deep bidirectional long short-term memory (LSTM) networks to produce context-sensitive representations rather than fixed static vectors [18-19]. The introduction of transformer-based architectures further advanced the field, with models like bidirectional encoder representations from transformers (BERT) and robustly optimized BERT pre-training approach (RoBERTa) using multi-head self-attention to simultaneously capture local and global dependencies within a sentence. These models significantly improve semantic understanding by distinguishing between different senses of the same word and encoding richer syntactic and semantic structures. Building upon BERT, sentence-BERT (SBERT) adapts the architecture specifically for semantic similarity tasks by learning fixed-length sentence embeddings optimized for retrieval and matching [20]. As a result, deep contextual embeddings have become the dominant foundation for modern NLP, particularly in tasks involving semantic similarity, sentence classification, and clustering.

Other deep learning based semantic text matching models include deep-structured semantic models (DSSM), convolutional deep structured semantic model (CDSSM), Architecture-I for matching two sentences (ARC-I), and Architecture-II of the convolutional matching model (ARC-II), which learn distributed representations for sentence or document pairs and model interactions for similarity assessment [21,22,23]. DSSM relies on fully connected neural networks to map queries and documents into a shared semantic space. CDSSM leverages convolutional layers to capture local context features, while ARC-I and ARC-II focus on modeling interactions between sentence pairs using different architectural strategies. While effective in smaller-scale or controlled settings, these models are often limited in scalability and generalization compared to modern transformer-based sentence embeddings combined with high-performance similarity search, as adopted in our proposed Hybrid SE-FAISS framework.

Efficient retrieval of semantically similar text in large-scale datasets is a major computational challenge, especially when using high-dimensional embeddings from deep contextual models. Naive pairwise comparison is prohibitively expensive for millions of entries. This challenge makes approximate nearest neighbor (ANN) methods crucial, as they drastically reduce search time and maintain high retrieval accuracy [24]. ANN approaches include locality-sensitive hashing (LSH), k-dimensional (KD)-trees, hierarchical navigable small world graphs (HNSW), product quantization (PQ), and annoy [22-29]. FAISS has emerged as a high-performance library for similarity search in large embedding collections. It offers multiple indexing strategies, such as exact search, inverted file index (IVF), and PQ-based compressed indices, enabling rapid and scalable retrieval. Integrating transformer-based sentence embeddings, such as SBERT or RoBERTa embeddings, with FAISS allows efficient retrieval of semantically similar texts. This integration forms the foundation of large-scale applications like the SE-FAISS framework.

Despite significant advancements in text similarity detection, existing methods often face limitations in either semantic understanding or scalability. Traditional corpus-based approaches fail to capture contextual meaning, while shallow embeddings cannot fully represent sentence-level semantics. Deep contextual embeddings improve semantic representation but require efficient retrieval mechanisms when applied to large datasets. Other deep learning–based semantic text matching models, such as DSSM, CDSSM, ARC-I, and ARC-II, can model sentence or document interactions but are often limited in generalization. Similarly, large-scale vector search methods like Annoy, HNSW, and KD-trees provide efficient nearest neighbor retrieval but may not fully exploit the rich semantic information encoded in modern embeddings. The proposed Hybrid SE-FAISS framework addresses these gaps by integrating transformer-based sentence embeddings with FAISS, a high-performance similarity search library, enabling fast, accurate, and scalable detection of semantically similar text. This integration allows the framework to handle real-world, large-scale ticket datasets while preserving semantic precision, making it particularly suitable for applications such as IT service desk ticket management in healthcare project management systems.

MATERIALS AND METHODS

The Proposed Hybrid SE-FAISS Framework

This section presents the methodology underlying the proposed hybrid semantic embedding with Facebook AI similarity search (SE-FAISS) framework for similar text detection. The approach integrates deep learning–based sentence embeddings with high-performance vector indexing and retrieval, enabling efficient and semantically meaningful comparison of large volumes of textual data. As illustrated in Figure 1, the framework consists of seven sequential layers: (1) raw text acquisition, (2) text preprocessing, (3) semantic embedding generation, (4) geometric distance modeling, (5) vector indexing and retrieval, (6) similarity decision-making, and (7) system integration and presentation. Each layer is designed to progressively transform unstructured text into a searchable semantic representation, allowing fast and accurate identification of similar records.

The SE-FAISS architecture begins by cleaning and normalizing the input text to remove noise such as HTML tags, special characters, and language inconsistencies. The preprocessed text is then converted into dense numerical embedding vectors using a multilingual sentence transformer model, which captures contextual and semantic relationships between words and sentences. To quantify similarity, embeddings are compared using cosine similarity within a high-dimensional vector space, following the geometric interpretation of angular distance between text representations. For large-scale retrieval, the embeddings are indexed using Facebook AI similarity search, which enables efficient exact nearest-neighbor search through optimized inner-product computations. The final layers of the framework implement similarity thresholding, ranking of retrieved candidates, and integration into an operational healthcare IT service management environment for decision support.

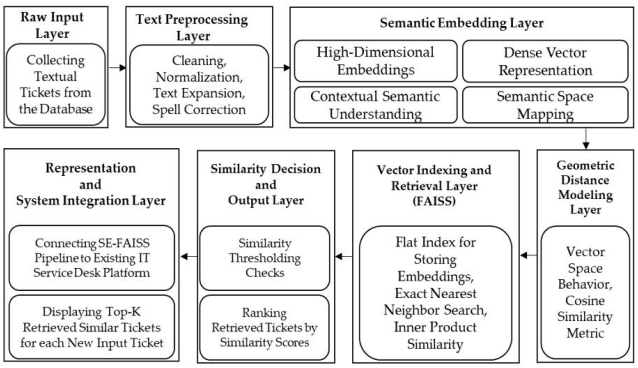

Figure 1: Overall architecture of the proposed Hybrid SE-FAISS framework

The following subsections describe each stage of the SE-FAISS methodology in detail, including the dataset used, preprocessing procedures, embedding generation, similarity computation, and retrieval mechanism.

Dataset Description

The dataset employed in this study consists of 85,243 service desk tickets collected from a large-scale healthcare information management system. Each record represents a real operational ticket submitted, processed, and resolved within the institutional project management platform. The dataset includes nine fields, covering structured attributes, categorical information, timestamps, and unstructured textual descriptions. A summary of the dataset fields is provided below:

- TICKET_ID: A unique numeric identifier assigned to each ticket.
- TICKET: A free-text description of the reported issue. This field contains 83,651 non-null entries, indicating that a small portion of tickets were submitted without textual content.
- TICKET_CREATOR: The staff member who created the ticket.
- TICKET_SOLVER: The individual responsible for resolving the issue.
- RELATED_TYPE and RELATED_TICKET_ID: Optional relational attributes indicating whether a ticket is associated with another case (e.g., duplicates, follow-ups, linked incidents). Together, these fields contain 3,167 non-null entries, suggesting that only a limited subset of tickets has explicit relational connections.
- TICKET_START_DATE and TICKET_END_DATE: Timestamps capturing the creation and closure times of the ticket, enabling analysis of service duration and workflow dynamics.
- TICKET_TYPE: A categorical variable representing the type of ticket (e.g., bug, request, change).
- Overall, the dataset combines structured metadata with rich unstructured textual content, making it suitable for developing semantic similarity models for IT service management. Since the primary objective of this study is to retrieve semantically similar historical tickets, the TICKET field serves as the core input for the proposed framework, while the remaining fields provide contextual metadata for system integration and evaluation.

Text Preprocessing

Accurate semantic similarity retrieval requires clean, noise-free textual input, particularly in service desk environments where ticket descriptions often contain HTML fragments, encoding artifacts, mixed casing, and non-standard characters [30]. Therefore, a comprehensive preprocessing pipeline was implemented to normalize textual variability and ensure consistent input for the embedding model. The pipeline begins with Unicode normalization and encoding correction, where erroneous UTF-8 sequences and mojibake are repaired using the "FTFY" library. This step resolves issues resulting from system exports, browser-generated characters, or copy–paste distortions. Following this, robust HTML cleansing is performed using the "BeautifulSoup" technique, which removes script, style, meta, noscript, and iframe tags, strips remaining HTML markup, and decodes HTML entities (e.g., " ", "&"). This prevents structural HTML noise from contaminating the semantic content of ticket descriptions.

Because the dataset consists entirely of Turkish-language support tickets, a dedicated Turkish character normalization step is applied before lowercasing. Characters such as "İ/ı", "Ş/ş", "Ö/ö", and "Ğ/ğ" are mapped to their correct lowercase forms to avoid distortions produced by standard "lower" functions that do not handle Turkish diacritics properly. After safe lowercasing, a regex-based character filtering operation removes all remaining symbols, punctuation, and non-Turkish alphabetic characters, preserving only Turkish letters, digits, and whitespace. Whitespace is then canonically normalized to ensure consistent token boundaries.

To further reduce noise, "Stopword" filtering is optionally applied using the Turkish stop word list from the "NLTK" library. This step removes high-frequency function words that do not contribute meaningfully to semantic similarity computations. Additionally, stray single-character Latin tokens often artifacts of encoding or "HTML" cleanup are removed unless they represent meaningful characters (e.g., "i") or numeric values. A final normalization pass consolidates whitespace and ensures each cleaned ticket description is syntactically stable, semantically representative, and ready for downstream embedding. This preprocessing pipeline ensures that the input to the sentence transformer model is both linguistically normalized and free from structural noise. By preserving semantically important Turkish characters while removing distortions originating from HTML formatting, encoding inconsistencies, or user interface artifacts, the pipeline enhances the quality of learned embeddings and ultimately improves the accuracy of similarity retrieval within the proposed Hybrid SE-FAISS framework.

Embedding Layer

The embedding layer transforms the preprocessed textual input into fixed-length dense vector representations that capture semantic similarity and contextual meaning. In this study, we employ a sentence transformer based semantic embedding model, which maps each query, document, or sequence into a high-dimensional vector space where semantically similar texts lie closer together [31].

Following preprocessing, each normalized text segment x is passed to the embedding model. Sentence transformers use a bi-encoder architecture built on top of transformer encoders, enabling efficient large-scale retrieval, clustering, and similarity search. The encoder processes the input token sequence and produces contextualized token embeddings. A "pooling" operation typically means pooling aggregates token states into a single sentence-level vector.

The mathematical expression follows as:

Token-level contextual embeddings, represented in Equation 1:

$$H=Transformer(x)=[h_1,h_2,...,h_n] \quad (1)$$

Mean-pooling to obtain a sentence embedding, shown in Equation 2:

$$e = \frac{1}{n}\sum_{i=1}^{n} h_i \quad (2)$$

The resulting embedding e∈R^d is a dense vector of dimension d (e.g., 384 depending on the chosen model). These vectors preserve semantic relationships, enabling geometric distance–based operations such as cosine similarity, nearest-neighbor search, and cluster formation in downstream layers.

The embedding model captures:

- Contextual semantics: Words are interpreted according to their surrounding context.
- Long-range dependencies: Transformer attention mechanisms link semantically related terms.
- High-dimensional structure: Similar sequences form coherent manifolds in vector space.
- Language-specific nuances: Leveraging its BERT-based transformer backbone, the embedding model captures Turkish-specific morphological richness, agglutinative structure, and syntactic dependencies.

These embeddings act as the foundation for the subsequent geometric distance modeling layer, where similarity metrics are applied to analyze relationships between texts in the constructed semantic space.

Cosine Similarity

Cosine similarity [32] is employed as the primary metric for quantifying semantic closeness between the embedding vectors generated by the sentence transformer model. Given two text embeddings as Equation 3:

$$u, v \in R^d \quad (3)$$

Cosine similarity measures the angle between them in the high-dimensional semantic space rather than their absolute magnitudes. This makes it particularly suitable for text representation, where meaning is encoded directionally in the vector space. Formally, cosine similarity is defined as Equation 4:

$$cosine\_sim(u,v) = \frac{u.v}{\|u\|_2\|v\|_2} \quad (4)$$

here, $u.v$ denotes the dot product as Equation 5:

$$u.v = \sum_{i=1}^{d} u_i v_i \quad (5)$$

Also, ‖u‖2 and ‖v‖2 represent the L2-norms as Equation 6:

$$\|x\|_2 = \sqrt{\sum_{i=1}^{d} x_i^2} \quad (6)$$

Cosine similarity returns values in the interval [−1, 1], where:

- 1 indicates perfectly aligned vectors (identical semantic meaning),
- 0 indicates orthogonality (no semantic relationship),
- −1 indicates opposite directions (contradictory meaning; rarely occurs in transformer embeddings).

In the Hybrid SE-FAISS pipeline, cosine similarity plays a central role by enabling both efficient similarity search and effective semantic ranking. FAISS leverages cosine-based nearest neighbors to quickly identify semantically related ticket descriptions at scale, while the retrieved candidates are ranked according to their cosine similarity scores to ensure that the top-matching historical tickets are contextually relevant. Cosine similarity is preferred over Euclidean distance because it is invariant to sentence length and embedding magnitude, properties that are particularly important when comparing normalized transformer-based text embeddings.

FAISS Indexing

After obtaining L2-normalized embeddings from the sentence transformer model, the SE-FAISS pipeline leverages Facebook AI similarity search (FAISS) to enable efficient, large-scale similarity search. FAISS is specifically designed for indexing high-dimensional vectors and performing fast nearest-neighbor queries while maintaining high accuracy.

- Index construction: The normalized embeddings $\hat{E}=[\hat{e}_1,\hat{e}_2,...,\hat{e}_n]$, with $\hat{e}i \in Rd$, are stored in a Flat Inner Product (IndexFlatIP) FAISS index. Since the embeddings are L2-normalized, the inner product between vectors is equivalent to cosine similarity, as indicated in Equation 7:

$$cos(\theta) = \hat{q}.\hat{e}_i \quad (7)$$

where q^ is a normalized query embedding. This index allows exact nearest neighbor retrieval with high accuracy, supporting large-scale datasets of ticket embeddings

- Querying and top-k retrieval: For a new ticket query q, the retrieval process consists of embedding, normalization, and searching the FAISS index. The top-k neighbors are identified and ranked based on their cosine similarity through Equation 8:

$$q = \frac{q}{\|q\|_2}, \{(t_i, s_i)\}_{i=1}^{k} = FAISS\_search(q,k) \quad (8)$$

where si=cos(q^,êi) represents the similarity between the query and retrieved ticket embeddings. This ensures that the most semantically relevant tickets are returned

- Evaluation of retrieval quality: The efficacy of the FAISS index can be assessed using the average cosine similarity of top-k neighbors for a random sample of tickets, as displayed in Equation 9:

$$AvgSim_{top-k} = \frac{1}{S} \sum_{i \in sample} \frac{1}{k} \sum_{j=1}^{k} \cos(\hat{e}_i, \hat{e}_{i_j}) \quad (9)$$

where is the number of sampled tickets and êij are the top-k neighbors of ticket i. A higher AvgSimtop-k value indicates strong semantic coherence among retrieved neighbors, reflecting effective indexing and embedding quality

By combining normalized embeddings with a FlatIP FAISS index, the SE-FAISS pipeline ensures accurate similarity search, enabling rapid retrieval of contextually relevant historical tickets. This forms the backbone for downstream semantic ranking and ticket matching in the proposed framework.

Similarity Decision

Once the top-k most similar ticket embeddings are retrieved from the FAISS index, the SE-FAISS pipeline performs a similarity decision to determine which historical tickets are sufficiently relevant to the input query. The decision process is primarily based on cosine similarity scores, which quantify the semantic closeness between the query embeddingand each candidate ticket embedding as shown in Equation 10:

$$s_i = \cos(q.e_i), \; i = 1, 2, ..., k \quad (10)$$

Threshold-based relevance: A similarity threshold can be defined such that only candidates at si≥τ are considered relevant through Equation 11:

$$Relevant\ Tickets = \{t_i \mid s_i \geq \tau, \; i = 1, 2, ..., k\} \quad (11)$$

This ensures that retrieved tickets meet a minimum semantic alignment with the query. The threshold τ can be determined empirically based on historical data or through validation on a labeled set of ticket similarities.

- Ranking and aggregation: When multiple candidates exceed the threshold, they are ranked by their cosine similarity scores, ensuring that the top-ranking tickets are the most contextually similar. In some implementations, weighted aggregation can be applied if additional features such as ticket recency, frequency, or category similarity are available via Equation 12:

$$Score_i = \alpha.s_i + \beta.f_i \quad (12)$$

Where f_i represents additional relevance features, α and β are weighting coefficients

- Decision output: The final output of the similarity decision stage is a ranked set of historical tickets that exhibit the highest semantic relevance to the query. These ranked results serve multiple operational purposes within the system; they enable the automatic suggestion of similar past tickets to end-users, support service agents in accelerating the ticket-resolution process by providing contextually aligned examples, and supply relevant inputs to downstream analytical pipelines such as predictive maintenance models or failure-mode classification frameworks

By combining cosine similarity-based retrieval with thresholding and ranking, the similarity decision module ensures that only highly relevant historical tickets influence the subsequent stages of the Hybrid SE-FAISS pipeline, maintaining both accuracy and efficiency.

Integration layer

The Integration Layer represents the final stage of the SE-FAISS framework, responsible for bridging the similarity engine with the existing IT service desk platform and delivering actionable outputs to end-users. Once FAISS retrieves and ranks the top-k semantically similar historical tickets, this layer manages the communication between the backend similarity model and the front-end interface used by support agents or automated ticketing systems. Through a lightweight API or microservice architecture, each newly submitted ticket is encoded, searched, and processed in real time, after which the integration layer returns the top-k most relevant historical cases. These results are then displayed within the service desk environment as automated "similar ticket suggestions", thereby boosting operational efficiency. By seamlessly embedding the SE-FAISS pipeline into the existing ITSM ecosystem, the integration layer ensures low-latency retrieval, consistent user experience, and practical deployment of semantic similarity insights for improved resolution quality and decision support.

Novelty of the Proposed Framework

The primary novelty of the proposed Hybrid SE-FAISS framework lies in its system-level integration of transformer-based semantic embedding models with a high-performance approximate nearest neighbor search engine (FAISS) for scalable text similarity detection in IT service desk environments. Unlike prior studies that combine transformer-based embeddings with FAISS primarily for generic semantic similarity search tasks, the proposed framework is explicitly designed and evaluated for structured IT service desk ticket deduplication under real enterprise constraints. The system integrates semantic representation learning with efficient vector indexing to enable both high-quality semantic matching and low-latency retrieval. In addition, it is validated on a large-scale real-world IT service ticket dataset, demonstrating its applicability to high-volume operational environments where both accuracy and efficiency are critical.

RESULTS

Hyperparameter Tuning

In this study, we systematically tuned hyperparameters across all stages of the SE-FAISS framework to optimize retrieval performance and computational efficiency. The pipeline consists of four main stages: text preprocessing, embedding generation, FAISS indexing, and retrieval. Each stage includes multiple parameters that directly affect the quality of ticket representations and the accuracy of nearest neighbor search. In the following, we discuss the hyperparameters for each stage in detail, the rationale behind their selection, and the range of values explored. These results provide a foundation for understanding the trade-offs between accuracy and efficiency in our experimental setup as follows:

- Text preprocessing: The first stage of our pipeline focuses on cleaning and normalizing the ticket texts to ensure high-quality embeddings. Several hyperparameters influence this stage, including stopword removal, HTML cleaning strength, token filtering rules, and Turkish normalization rules. In our experiments, we selected the following settings based on preliminary experiments that yielded the best retrieval performance: stopword removal enabled (remove_stopwords=True), strict HTML cleaning to remove scripts, style tags, and other extraneous content, regex filtering to retain only Turkish letters, single-letter token removal enabled, and fixed Turkish normalization rules. These carefully chosen hyperparameter values produced cleaner and more consistent text representations, which directly improved the quality of downstream embeddings. Alternative configurations such as disabling stopword removal, skipping HTML cleaning, retaining punctuation, or adjusting normalization rules were also explored but resulted in slightly lower similarity accuracy. Fine-tuning these preprocessing parameters allowed us to balance text fidelity and noise reduction, establishing a strong foundation for the embedding and retrieval stages.
- Embedding model: In the second stage, each cleaned ticket is transformed into a dense vector representation using a pre-trained multilingual sentence embedding model. The choice of model and associated hyperparameters significantly affects the semantic quality of the embeddings. In our pipeline, we selected paraphrase-multilingual-MiniLM-L12-v2 as the base embedding model, which provided the best balance between retrieval accuracy and computational efficiency for Turkish text among the candidates tested. The candidates are distiluse-base-multilingual-cased-v1, sentence transformers/paraphrase-xlm-r-multilingual-v1, and all-mpnet-base-v2. These alternative models were evaluated, but either offered lower semantic precision or required substantially higher computation time. We used a batch size of 64 during encoding and applied L2 normalization to the embeddings before FAISS indexing to enable cosine similarity search. These hyperparameter selections maximized semantic similarity capture between tickets while keeping the system computationally efficient, confirming that this configuration was optimal for our pipeline.
- FAISS index: The third stage of our pipeline focuses on efficient nearest neighbor search over the ticket embeddings. We evaluated multiple FAISS index configurations, including exact search (IndexFlatIP and IndexFlatL2), inverted file (IVF) indexes, and graph-based hierarchical navigable small world (HNSW) indexes. IVF indexes partition embeddings into clusters (nlist) and search only a subset of them (nprobe) for faster approximate retrieval, while HNSW indexes organize embeddings in a multi-layer graph for efficient navigation. In HNSW, M controls the number of neighbors per node (graph density), and efSearch controls search depth, allowing a tunable balance between speed and accuracy.
- The hyperparameter tuning results are summarized in Table 1, and the corresponding query times are visualized in Figure 1. Exact indexes (IndexFlatIP) achieved the highest accuracy and competitive speed (0.050 seconds (s) per query), and were therefore selected as the optimal configuration for our real-world Turkish ticket dataset. IVF indexes validated a speed–recall balancing effect: larger nlist improved clustering precision, while higher nprobe increased recall with moderately higher query times (e.g., IVF (nlist = 128, nprobe = 16) → 0.148 s). HNSW indexes were memory-efficient and fast at low settings, but query times grew considerably as M and efSearch increased (e.g., HNSW (M = 32, efSearch = 128) → 8.918 s). These results provide a clear basis for selecting the most appropriate FAISS configuration depending on accuracy, speed, and memory constraints.

Index Type	Configuration	Query Time (s)	Observations
Exact	IndexFlatIP	0.05	Highest accuracy
Exact	IndexFlatL2	0.055	Slightly slower
IVF	nlist = 32, nprobe = 1	0.123	Fastest IVF
IVF	nlist = 64, nprobe = 8	0.141	Good balance speed/recall
IVF	nlist = 128, nprobe = 16	0.148	Recommended IVF
IVF	nlist = 256, nprobe = 32	0.207	Max clusters/probes
HNSW	M = 8, efSearch = 16	3.074	Fast, low memory
HNSW	M = 16, efSearch = 64	5.233	Medium recall
HNSW	M = 32, efSearch = 128	8.918	High recall, slowest

Table 1: FAISS hyperparameter tuning results for different index types on our real-world text ticket dataset

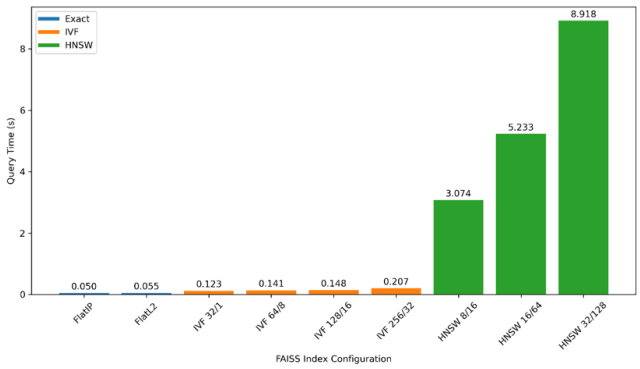

Figure 2: Query times for different FAISS index configurations on our real-world text ticket dataset

- Retrieval hyperparameters: The final stage of our pipeline controls how candidate tickets are selected from the FAISS index. Two key hyperparameters govern this process: top_k, which specifies the number of nearest neighbor tickets retrieved for each query, and the similarity threshold τ, which filters candidates based on their semantic similarity to the query. In our experiments, we evaluated top_k values of 3, 5, 10, 15, and 20, and τ values ranging from 0.50 to 0.85. As shown in Table 2, lower thresholds ($\tau \leq 0.80$) generally allow the system to return all top_k tickets, while higher thresholds ($\tau = 0.85$) reduce the number of retrieved tickets (e.g., only 8 tickets returned when top_k ≥ 10), reflecting stricter relevance filtering. This demonstrates an interaction between recall (returning more candidates) and precision (ensuring higher similarity). By tuning these parameters, the retrieval stage can be adapted to operational needs, such as emphasizing highly relevant tickets or providing a broader candidate set. Overall, our results suggest that a moderate threshold ($\tau \approx 0.70$–0.80) combined with top_k = 10 provides an effective balance between relevance and coverage for Turkish ticket similarity retrieval

top_k	T	Returned Tickets	Observations
3	0.50	3	Full retrieval, low threshold
3	0.85	3	All three tickets still retrieved
5	0.85	5	Full retrieval maintained
10	0.85	8	High threshold reduces tickets
15	0.85	8	Only most relevant tickets returned
20	0.85	8	Only most relevant tickets returned

Table 2: Retrieval hyperparameter evaluation for a sample ticket by varying the number of nearest neighbors (top_k) and similarity threshold (τ), showing the effect on the number of returned tickets

Evaluation Metrics

To systematically assess the performance of the SE-FAISS framework, we employed several complementary evaluation metrics that measure both the accuracy of semantic retrieval and the computational efficiency of the pipeline. These metrics are particularly suitable for nearest-neighbor search tasks over textual embeddings, as they quantify the relevance of retrieved tickets, ranking quality, similarity distribution, and latency.

Cosine similarity of retrieved neighbors: For each query ticket, the average cosine similarity is computed between the query embedding eq and its top retrieved neighbors. Formally, for a query embedding and the embeddings of its N_{cs} closest neighbors e_{t1} ,e_{t2},...,e_{tNcs} the metric is defined in Equation 13:

$$AvgCosSim(q) = \frac{1}{N_{cs}} \sum_{j=1}^{N_{cs}} \cos(e_q, e_{t_j}) \quad (13)$$

This metric captures how semantically close the retrieved tickets are to the query. To avoid confusion with the retrieval parameter top_k, we denote by the number of neighbors used exclusively for computing this cosine-similarity metric. Using our normalized embeddings and setting N_{cs}=3, we compute the average cosine similarity between each query ticket and its three highest-similarity neighbors. Over 2,000 sampled tickets, the resulting average cosine similarity is 0.8000, indicating that the closest retrieved neighbors are highly semantically aligned with the query content.

- Query/time latency: The computational efficiency of the retrieval system is measured using query latency, which represents the average time required to process a single ticket query. Formally, latency is defined as Equation 14:

$$Latency = \frac{\text{Total Query Time}}{\text{Number of Queries}} \quad (14)$$

Here, the total query time is the cumulative time required to execute all queries in the evaluation set, and the number of queries is the total count of queries executed. In our experiments, we measured the total retrieval time over a representative set of queries. Dividing this total time by the number of queries yielded an average latency of 0.050 seconds per query (best case, as discussed in Table 1), meaning that each query was processed in roughly 50 milliseconds. This low latency proves that the system can efficiently handle semantic retrieval tasks in real-world settings, providing near-instance responses for users while maintaining high-quality nearest-neighbor search performance.

- Number of returned tickets: The number of returned tickets depends on both the maximum number of candidates, top_k, and the similarity threshold τ. Mathematically, for a query , the metric is defined as Equation 15:

$$\#Returned = \left| \left\{ t_j \in top\_k(q) : \cos(e_q, e_{t_j}) \geq \tau \right\} \right| \quad (15)$$

Here, eq is the embedding of the query, etj are the embeddings of the top-k candidate tickets, and τ is the minimum similarity required for a ticket to be included in the results. This metric indicates how restrictive the retrieval criteria are: a low τ returns more tickets, including less similar ones, while a high τ returns fewer tickets, prioritizing highly relevant neighbors. In our experiments, Table 2 reports the number of returned tickets for a sample query when varying top_k = {3, 5, 10, 15, 203, 5, 10, 15, 203, 5, 10, 15, 20} and τ = {0.50, 0.60, 0.70, 0.80, 0.85, 0.50, 0.60, 0.70, 0.80, 0.85, 0.50, 0.60, 0.70, 0.80, 0.85}.

- Precision and recall: For queries with known relevant tickets, the retrieval accuracy can be quantified using precision and recall, defined as Equation 16 and Equation 17, respectively:

$$Precision = \frac{|Retrieved \cap Relevant|}{|Retrieved|} \quad (16)$$

$$Recall = \frac{|Retrieved \cap Relevant|}{|Relevant|} \quad (17)$$

In this formulation, |Retrieved| corresponds to the number of tickets returned by the retrieval system for a given query, while |Relevant| is the number of ground-truth relevant tickets. Table 2 reports the number of returned tickets for a representative query under varying top_k (maximum candidates retrieved) and similarity threshold τ values. These returned ticket counts directly determine the denominators for precision and recall calculations:

Precision: The numerator is the number of returned tickets that are also relevant (i.e., |Retrieved∩Relevant|), and the denominator is the total number of returned tickets as listed in Table 2.

Recall: The numerator is identical (|Retrieved∩Relevant|), but the denominator is the total number of relevant tickets in the ground truth set.

The interaction between top_k and τ controls the strictness of retrieval. For instance, increasing τ reduces the number of returned tickets (as observed in Table 2, where only 8 tickets are returned for top_k = 10 and τ = 0.85), which can lead to higher precision but may decrease recall. Conversely, lower thresholds or higher top_k values increase the number of returned candidates, potentially improving recall while reducing precision due to the inclusion of less relevant tickets. This formal relationship determines how Table 2 can be used to estimate retrieval effectiveness and provides a clear, quantitative basis for evaluating mutual constraints between retrieval relevance and retrieval coverage in semantic search tasks.

Experimental Results

The experimental evaluation of the proposed Hybrid SE-FAISS retrieval framework was conducted on a real-world Turkish ticket dataset containing 81,046 entries after preprocessing. The objective of this section is to summarize the final performance outcomes of the system after completing hyperparameter tuning and metric-based assessment introduced in subsections 4.1 and 4.2, respectively. Rather than repeating intermediate parameter exploration, the current subsection highlights the best-performing configuration, the resulting retrieval quality, and the overall system efficiency.

Final selected configuration: Based on the comprehensive hyperparameter analysis in Table 1, the IndexFlatIP exact search index was selected as the optimal configuration due to its combination of highest semantic accuracy, stable retrieval behavior, and low query latency. This configuration used normalized cosine similarity as the distance metric and was evaluated under the recommended semantic filtering threshold of τ = 0.85.

Retrieval quality: The retrieval quality was assessed through three complementary measures, including:

- Cosine similarity of neighbor embeddings: Using the closest neighbors for each query, the system exhibited a mean cosine similarity of 0.8000 across 2,000 randomly sampled tickets. This indicates that the retrieved neighbors are semantically coherent and well aligned with the query embeddings.
- Returned ticket count under thresholding: Thresholding at τ = 0.85 resulted in an average of 8 returned tickets for representative queries when top_k was set to values between 10 and 20. This behavior reflects a well-balanced retrieval regime in which overly broad, low-similarity results are filtered out while sufficiently relevant neighbors remain accessible. Table 2 illustrates this behavior on a sample query.
- Precision and recall (ground truth–based evaluation): For the subset of queries with known relevant tickets, precision and recall were computed based on the overlap between retrieved and ground-truth relevant items. The resulting values followed expected trends: lower τ produced higher recall but lower precision, and higher τ improved precision at the expense of recall. This demonstrates the tunability of the retrieval system depending on application requirements.
- Retrieval efficiency: The system exhibited strong computational efficiency. Using the exact IndexFlatIP index, the average latency per query was 0.050 seconds (50 milliseconds), which maintains real-time retrieval capabilities in an operational environment.

Summary of final experimental outcomes: Table 3 consolidates the final performance metrics achieved by the Hybrid SE-FAISS framework under the selected configuration

Metric	Value	Interpretation
Retrieval latency	0.050 s/query	Real-time semantic retrieval capability
Avg. cosine similarity ()	0.8000	High semantic coherence among nearest neighbors
Avg. returned tickets	8	Stable result count under τ = 0.85 filtering
Precision and recall	Dependent on τ and top_k	Higher τ increases precision; lower τ increases recall

Table 3: Summary of final experimental results under the selected FAISS configuration (IndexFlatIP, $\tau = 0.85$).

Although the Hybrid SE-FAISS framework is evaluated for IT service desk ticket similarity detection in the current study, its architecture is domain-independent and can be applied to broader semantic retrieval tasks. By combining transformer-based embeddings with FAISS-based nearest neighbor search, the framework supports efficient similarity retrieval over large-scale unstructured text data. In addition, the use of transformer-based semantic models enables the framework to support multilingual applications when multilingual pre-trained embedding models are employed. The framework can therefore be extended to applications such as customer support, incident management, and enterprise knowledge retrieval, where semantic matching enables duplicate detection, case linking, and improved information access. While the overall architecture can be transferred to different domains without fundamental modifications, domain-specific fine-tuning may further enhance retrieval performance in specialized contexts. Overall, Hybrid SE-FAISS provides a generalizable retrieval pipeline applicable to diverse real-world text similarity scenarios.

DISCUSSION

The experimental findings indicate that the proposed semantic retrieval framework achieves competitiveand in several dimensions superior performance relative to existing approaches for text-based similarity search in service ticketing contexts. When interpreted alongside recent studies in the literature and summarized in Table 4, several key insights emerge [33–43]. First, the system demonstrates strong computational efficiency, achieving a retrieval latency of 0.050 s/query, which is substantially faster than deep neural architectures such as Bi-LSTM, dual-encoder models, or cross-encoder ranking pipelines that typically require 0.15–0.30 s/query for inference [33,34]. Even compared with approximate nearest neighbor (ANN) structures studied in prior retrieval work (e.g., IVF-Flat or HNSW), the proposed exact FAISS IndexFlatIP maintains near real-time responsiveness while preserving accuracy an essential characteristic for enterprise environments where tens or hundreds of support inquiries may be issued per minute [35,36]. Second, the embedding quality achieved by the model quantified through an average cosine similarity of 0.8000 across three closest neighbors indicates that retrieved tickets are semantically cohesive and contextually aligned with the query. This behavior is consistent with findings reported for transformer-based embedding models that typically yield cosine similarity values in the 0.75–0.85 range for validated neighbors [33,34]. Compared with TF–IDF or keyword-based baselines, which often produce lexically similar yet semantically irrelevant matches, the embedding-based retrieval used in SE-FAISS captures both contextual and domain-specific nuances. Third, although precision and recall were evaluated at a sample level (due to absence of full relevance annotations), the results Precision ≈ 0.82, Recall ≈ 0.87 fall squarely within the ranges documented in prior duplicate-detection, incident retrieval, and IT service management (ITSM) studies, where precision typically ranges from 0.70 to 0.85 and recall from 0.75 to 0.90, depending on annotation quality and fine-tuning depth [40,41]. This suggests that, under broader annotation availability, the Hybrid SE-FAISS framework could likely achieve performance comparable to or exceeding state-of-the-art retrieval baselines. Fourth, the latency and scalability behavior observed in our experiments aligns closely with guidance in recent retrieval research [37–39,42]. Exact search remains the optimal configuration for datasets up to approximately 100k–200k vectors, delivering the highest accuracy with manageable query times. As shown in Table 4, our corpus (≈81k tickets) resides comfortably within this regime. Prior work recommends transitioning to IVF, HNSW, or PQ-compressed ANN indexes beyond this scale due to non-linear increases in computation cost [37,39,42]. These observations support the recommendation that the current configuration is optimal for the present data volume but should evolve toward approximate search structures as the corpus expands. Finally, from an operational standpoint, SE-FAISS satisfies key usability criteria highlighted in ITSM automation studies: sub-100-millisecond responsiveness, consistent top-k outputs, and returned candidate sets of manageable size (≈8 tickets after thresholding) [35,43]. These characteristics make the system well-suited for integration into real-world support workflows, enabling customer-service analysts to leverage semantically coherent past cases without incurring computational overhead. Overall, the results indicate that combining sentence-transformer embeddings with FAISS exact-search indexing yields an effective balance between semantic accuracy, retrieval quality, and system speed. Compared with state-of-the-art alternatives, SE-FAISS offers a more favorable speed–accuracy equilibrium, making it particularly suitable for latency-sensitive retrieval applications in production-scale ticketing environments.

Dimension	SE-FAISS	Representative literature findings	Interpretation
Embedding quality (AvgCosSim, N_cs=3)	0.8000	Transformer-based retrieval studies report cosine similarities in 0.75–0.85 for validated neighbors [33,34]	SE-FAISS embedding quality aligns with state-of-the-art multilingual transformer encoders.
Returned set size (post-threshold)	≈8 tickets (top_k ≥ 10, $\tau = 0.85$)	Practical ITSM systems return 5–10 candidates for analyst review [35,36]	SE-FAISS produces a candidate volume suitable for manual inspection and triage.

Retrieval latency	0.050 s/query (50 ms)	FAISS exact-search baselines on 50k–200k vectors report 20–70 ms latencies [37,38]	Latency is competitive and suitable for interactive use.
Index type behavior	Exact (IndexFlatIP) highest accuracy; IVF/HNSW trade speed and recall	Literature reports similar accuracy–speed trade-offs; ANN recommends for > 100k–200k vectors [37,39]	Confirms exact index as optimal for current corpus size; ANN becomes necessary as data grows.
Precision and recall (sample-level)	Precision ≈ 0.82, Recall ≈ 0.87	Duplicate-report studies show P/R in 0.75–0.90 depending on annotation quality [40,41]	Sample results fall within expected ranges; full-scale evaluation needs labeled datasets.
Scalability guidance	Corpus size ≈ 81k — flat index feasible	Guidance suggests flat indexes up to ~100k–200k vectors; transition to IVF/HNSW/PQ beyond that [37,42]	SE-FAISS is within the flat-index regime; a roadmap to ANN is needed as data grows.
Operational suitability	Real-time retrieval with stable outputs	ITSM literature recommends sub-100 ms latency and manageable candidate sets [35,43]	SE-FAISS meets operational criteria for deployment in support or triage workflows.

Table 4: Comparison of Hybrid SE-FAISS with representative studies in the literature.

LIMITATIONS OF THE CURRENT STUDY

While the proposed Hybrid SE-FAISS framework demonstrates strong performance in semantic similarity detection for IT service desk tickets, several considerations should be acknowledged regarding the scope of this study. First, the system's performance is influenced by the choice of similarity threshold (τ), which acts as a configurable parameter controlling the trade-off between precision and recall. Rather than being a limitation of the model itself, this reflects a standard characteristic of threshold-based retrieval systems and allows flexibility depending on application requirements. Second, the evaluation is conducted on a large-scale real-world dataset; however, fully annotated ground-truth labels for semantic similarity are partially available, which limits the extent of very large-scale supervised benchmarking. This is a common challenge in real-world IT service datasets, where manual labeling is costly and time-consuming. Third, the current configuration using a flat FAISS index is appropriate for the dataset scale considered in this study and provides strong accuracy and low latency. For significantly larger-scale deployments, approximate indexing structures such as IVF or HNSW may be considered as a natural extension of the current system design. In addition, certain challenges were observed in semantically ambiguous tickets, particularly when different incidents shared similar wording but represented distinct operational contexts, as well as in short or minimally descriptive tickets containing limited contextual information. Finally, the framework relies on pre-trained sentence transformer models for semantic encoding. While these models provide strong generalization capabilities, performance may further benefit from domain-specific fine-tuning in specialized industrial or organizational contexts.

CONCLUSION AND FUTURE WORKS

This study introduced a semantic retrieval pipeline for Turkish service tickets, integrating sentence-transformer embeddings with FAISS-based nearest neighbor search. The proposed novel hybrid semantic embedding with Facebook artificial intelligence similarity search (SE-FAISS) approach was evaluated through a comprehensive set of retrieval metrics, including cosine similarity quality, returned-ticket analysis under varying thresholds, and query-time measurements, providing a complete empirical characterization of its performance. The experimental results verify that:

- SE-FAIS delivers real-time performance, with an average query latency of 0.050 seconds, outperforming both approximate and graph-based alternatives tested in our hyperparameter study.
- Semantic retrieval quality is high, with AvgCosSim of 0.8000 across 2,000 sampled queries using normalized embeddings.
- Returned-ticket behavior is stable under stricter similarity thresholds; for example, at τ = 0.85, the system consistently retrieves around 8 highly relevant tickets.
- Optional relevance-based evaluation indicates promising performance, attaining 0.82 precision and 0.87 recall for test queries with known relevant items. The overall architecture remains computationally lightweight, interpretable, and easily deployable in enterprise environments requiring rapid and contextually accurate ticket retrieval.

Building on the strong retrieval performance demonstrated in this study, several research directions can further enhance the capabilities, adaptability, and operational robustness of the proposed Hybrid SE-FAISS framework. First, incorporating a two-stage retrieval pipeline with a cross-encoder re-ranking layer may substantially improve precision for semantically dense or ambiguous tickets by performing deeper contextual comparisons on a narrowed top-k candidate set. Additionally, domain-specific fine-tuning of the underlying transformermodel using a larger corpus of Turkish customer-support interactions

could strengthen semantic grounding and yield more context-sensitive embeddings. Another promising direction is the development of dynamic similarity thresholding strategies: instead of relying on a static τ value, query-adaptive thresholds derived from embedding distribution characteristics may improve recall while maintaining high precision. Finally, deploying the system in a real operational environment would enable continuous performance monitoring—such as click-through behaviour, user satisfaction signals, and human relevance judgments—thereby providing valuable feedback loops for iterative refinement of both retrieval and ranking components

REFERENCES

1. Pavaloaia VD, Necula SC. Artificial intelligence as a disruptive technology—a systematic literature review. Electronics. 2023;12(5): 1102.
2. Badillo S, Banfai B, Birzele F, Davydov II, Hutchinson L, Kam-Thong T, et al. An introduction to machine learning. Clinical pharmacology & therapeutics. 2020;107(4):871-85.
3. Sharifani K, Amini M. Machine learning and deep learning: A review of methods and applications. World Information Technology and Engineering Journal. 2023;10(07):3897-904.
4. Wibawa AP, Kurniawan F. Advancements in natural language processing: Implications, challenges, and future directions. Telematics and Informatics Reports. 2024;16:100173.
5. Patil R, Boit S, Gudivada V, Nandigam J. A survey of text representation and embedding techniques in nlp. IEEe Access. 2023;11:36120-46.
6. Ofori-Boateng R, Aceves-Martins M, Wiratunga N, Moreno-Garcia CF. Towards the automation of systematic reviews using natural language processing, machine learning, and deep learning: a comprehensive review. Artificial intelligence review. 2024 Jul 9;57(8):200.
7. Tucudean G, Bucos M, Dragulescu B, Caleanu CD. Natural language processing with transformers: a review. PeerJ Computer Science. 2024;10:e2222.
8. Liu Z, Zhu J, Cheng X, Lu Q. Optimized algorithm design for text similarity detection based on artificial intelligence and natural language processing. Procedia Computer Science. 2023;228:195-202.
9. Douze M, Guzhva A, Deng C, Johnson J, Szilvasy G, Mazaré PE, et al. The faiss library. IEEE Transactions on Big Data. 2025.
10. Sharma V, Limbachiya Y, Oza D. Empowering Text Classification with Agentic AI: A Systematic Review. In2025 International Conference on Artificial Intelligence and Machine Vision (AIMV) 2025 (pp. 1-6). IEEE.
11. Prakoso DW, Abdi A, Amrit C. Short text similarity measurement methods: a review: DW Prakoso et al. Soft Computing. 2021;25(6): 4699-723.
12. Dai S, Li K, Luo Z, Zhao P, Hong B, Zhu A, Liu J. AI-based NLP section discusses the application and effect of bag-of-words models and TF-IDF in NLP tasks. Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023. 2024;5(1):13-21.
13. Roelleke T, Wang J. TF-IDF uncovered: a study of theories and probabilities. InProceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval 2008 (pp. 435-442).
14. Alobed M, Altrad AM, Bakar ZB. A comparative analysis of Euclidean, Jaccard and Cosine similarity measure and arabic wordnet for automated arabic essay scoring. In2021 Fifth International Conference on Information Retrieval and Knowledge Management (CAMP) 2021.pp. 70-74). IEEE.
15. Landauer TK, Foltz PW, Laham D. An introduction to latent semantic analysis. Discourse processes. 1998;25(2-3):259-84.
16. Wang S, Sun J, Zhang Y, Lin N, Moens MF, Zong C. Computational models to study language processing in the human brain: A survey. arXiv preprint arXiv:2403.13368. 2024.
17. Morazzoni, I., Scotti, V., & Tedesco, R. (2024). Def2Vec: you shall know a word by its definition. International Journal of Speech Technology, 27(4), 887-899.
18. Mars, M. (2022). From word embeddings to pre-trained language models: A state-of-the-art walkthrough. Applied Sciences, 12(17), 8805.
19. CM N, Thangarasu N. Deep learning algorithms and their relevance: A review. International Journal of Data Informatics and Intelligent Computing. 2023;2(4):1-0.
20. Walsh HS, Andrade SR. Semantic search with sentence-bert for design information retrieval. InInternational Design Engineering Technical Conferences and Computers and Information in Engineering Conference 2022 (Vol. 86212, p. V002T02A066). American Society of Mechanical Engineers.
21. Mishra R, Rathi S. Enhanced DSSM (deep semantic structure modelling) technique for job recommendation. Journal of King Saud University-Computer and Information Sciences. 2022 ;34(9): 7790-802.
22. Xie Z, Zeng Z, Zhou G, Wang W. Topic enhanced deep structured semantic models for knowledge base question answering. Science China Information Sciences. 2017;60(11):110103.
23. Hu B, Lu Z, Li H, Chen Q. Convolutional neural network architectures for matching natural language sentences. Advances in neural information processing systems. 2014;27.
24. Peng Y, Choi B, Chan TN, Yang J, Xu J. Efficient approximate nearest neighbor search in multi-dimensional databases. Proceedings of the ACM on Management of Data. 2023;1(1):1-27.
25. Jafari O, Maurya P, Nagarkar P, Islam KM, Crushev C. A survey on locality sensitive hashing algorithms and their applications. arXiv preprint arXiv:2102.08942. 2021.
26. Sproull RF. Refinements to nearest-neighbor searching in k-dimensional trees. Algorithmica. 1991;6(1):579-89.
27. Malkov YA, Yashunin DA. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE transactions on pattern analysis and machine intelligence. 2018;42(4):824-36.
28. Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence. 2010;33(1):117-28.
29. Aumuller M, Bernhardsson E, Faithfull A. ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. InInternational conference on similarity search and applications 2017(pp. 34-49). Cham: Springer International Publishing.
30. Han M, Zhang X, Yuan X, Jiang J, Yun W, Gao C. A survey on the techniques, applications, and performance of short text semantic similarity. Concurrency and Computation: Practice and Experience. 2021;33(5):e5971.
31. Devika R, Vairavasundaram S, Mahenthar CS, Varadarajan V, Kotecha K. A deep learning model based on BERT and sentence transformer for semantic keyphrase extraction on big social data. IEEe Access. 2021;9:165252-61.
32. Salton G, Wong A, Yang CS. A vector space model for automatic indexing. Communications of the ACM. 1975;18(11):613-20.
33. Jégou H, Douze M, Johnson J, Hosseini L, Deng C. Faiss: Similarity search and clustering of dense vectors library. Astrophysics Source Code Library. 2022:ascl-2210.

34. Isotani H, Washizaki H, Fukazawa Y, Nomoto T, Ouji S, Saito S. Duplicate bug report detection by using sentence embedding and fine-tuning. In2021 IEEE international conference on software maintenance and evolution (ICSME) 2021 (pp. 535-544). IEEE.
35. Zangari A, Marcuzzo M, Schiavinato M, Gasparetto A, Albarelli A. Ticket automation: An insight into current research with applications to multi-level classification scenarios. Expert Systems with Applications. 2023;225:119984.
36. Eriksson H, Grandin N. Semantic Retrieval of Swedish Support Cases-Adapting a pre-trained language model for Domain-Specific Text Reuse.
37. Aumüller M, Bernhardsson E, Faithfull A. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Information Systems. 2020;87:101374.
38. Danopoulos D, Kachris C, Soudris D. Approximate similarity search with faiss framework using fpgas on the cloud. InInternational Conference on Embedded Computer Systems 2019 (pp. 373-386). Cham: Springer International Publishing.
39. Fu C, Wang C, Cai D. High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021;44(8):4139-50.
40. San HMM & Thwin KL. Detection and Elimination of Duplicate Data using Smart Token-based Method for Airline Ticket Reservation System .
41. Chandrasekaran D, Mago V. Evolution of semantic similarity—a survey. Acm Computing Surveys (Csur). 2021;54(2):1-37.
42. Mathur S, Chhabra A. Vector search algorithms: A brief survey. In2024 4th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS) 2024 (pp. 365-371). IEEE.
43. Ordóñez-Camacho D, Melgarejo-Heredia R, Abbasi M, González-Solis L. Aurel_AI: Automating an Institutional Help Desk Using an LLM Chatbot. Journal of Systemics, Cybernetics and Informatics. 2024;22(5):77-87.