Diffusion-Aided RAG: Elevating Dense-Retrieval Chatbots via Graph-Based Diffusion Reranking

Sai Teja Dampanaboina[1], Sai Nishchal Gamini[1], Karishma Kunwar[1], Marco Polignano[2], Marco Levantesi[1,3], Giovanni Semeraro[2] and Ernesto William De Luca[1,3]

[1]*Otto-von-Guericke University, Universitätspl. 2, 39106 Magdeburg, Germany*
[2]*University Of Bari Aldo Moro, via E. Orabona 4, 70125, Bari, Italy*
[3]*Leibniz Institute for Educational Media | George Eckert Institute, Brunswick, Germany*

Abstract

This paper presents a comprehensive framework for enhancing dense-retrieval-based chatbots through the integration of graph-based diffusion reranking. Addressing challenges in traditional retrieval-augmented generation (RAG) systems, the proposed methodology incorporates a multi-step pipeline that advances document retrieval and relevance ranking. Initially, candidate passages are retrieved via dense embeddings, followed by the construction of a graph representation that captures inter-passage semantic relationships. Through a graph-based diffusion process, the reranking mechanism refines the selection, amplifying clusters of contextually relevant documents while mitigating noise effects from irrelevant data points. Experimental results demonstrate significant gains in retrieval quality and question-answering accuracy, underscoring the framework's potential for knowledge-intensive real-time applications such as conversational AI. This work reflects a pivotal step towards developing highly accurate, dynamic, and scalable multimodal conversational systems.

Keywords

Retrieval-Augmented Generation, Large Language Models, Chatbots, Knowledge Graph, PageRank

1. Introduction

Advanced chatbots and other modern NLP tools need fast access to up-to-date, specific information. Although Large Language Models (LLMs) can generate fluent responses and handle a wide range of topics, they're stuck with whatever they learned during training, and their knowledge can become outdated or be too general [1]. RAG [2] solves this by enabling the LLM to retrieve information from an external database that can be updated in real time. This strategic decoupling of the LLM's generative function from data management, including storage, indexing, and crucially, retrieval, allows for continuous knowledge updates, thereby enhancing the responsiveness, reliability, and domain fidelity of such systems. Being able to quickly and accurately find the right information from a variety of sources is essential for powering these next-generation NLP systems.

Information Retrieval (IR) has evolved a lot to help us find relevant information more quickly and accurately in huge collections of text [3]. Instead of just matching words on the page, many modern systems use dense representations; basically, numeric embeddings that capture the meaning of queries and documents. This method, called **dense passage retrieval** [4], makes it possible to find passages that are related in meaning even if they don't share the same exact words. Still, pulling back the single best set of passages from an enormous database is tough, and the first batch of results often require additional refinement to make sure they're really on point. That's why it's common to run additional steps like re-ranking to fine-tune and improve the final selection.

We aim to make dense passage retrieval work even better by using a multi-step pipeline. First, we pull an initial batch of candidate passages with a dense retriever. Then we turn those top documents into a graph and run a diffusion process over it. This lets us capture how the passages relate to each other. By using this graph-based diffusion as a re-ranker, we can tweak the initial scores so that the most truly relevant passages end up at the top. The objective is to demonstrate how combining dense retrieval with graph-based diffusion re-ranking can yield superior retrieval performance, providing a more accurate and contextually relevant set of documents essential for applications requiring dynamic knowledge access.

2. Related Work

In recent years, advances in Large Language Models (LLMs) and AI-driven dialog systems have enabled more dynamic, retrieval-augmented conversational platforms. One foundational effort was introduced by Guu *et al.*

CLiC-it 2025: Eleventh Italian Conference on Computational Linguistics, September 24 — 26, 2025, Cagliari, Italy
*All authors contributed equally.

sai.dampanaboina@ovgu.de (S. T. Dampanaboina); sai.gamini@st.ovgu.de (S. N. Gamini); karishma.kunwar@st.ovgu.de (K. Kunwar); marco.polignano@uniba.it (M. Polignano)

in the REALM framework [5], which demonstrated the effectiveness of retrieval-augmented language model pre-training by fine-tuning on open-domain question answering (Q&A). At inference time, **REALM** fetches documents using dense embeddings and conditions the generator on retrieved passages. Building on this idea, Lewis *et al.* formalized the **Retrieval-Augmented Generation (RAG)** architecture [2] , showing that coupling dense retrieval with a pretrained sequence-to-sequence model improves factual grounding and generalization in Q&A. Unlike traditional LLMs that rely solely on parametric memory, RAG leverages a non-parametric index to fetch up-to-date, domain-specific information during generation.

Prior to dense retrieval, sparse vector-space methods—such as TF-IDF or BM25—were the de facto standards for fetching relevant documents [6]. Although BM25 performs well on short, keyword-based queries, it struggles with semantic matching in open-domain contexts[7]. Karpukhin *et al.* [4] showed that a dual-encoder dense retrieval model, trained on relatively few question–passage pairs, could outperform a strong BM25 baseline. Subsequent work by Xiong *et al.* [8] and Qu *et al.* [9] confirmed that dense retrievers better handle paraphrased, abstract, and long-tail queries. These studies also highlighted challenges in dense retrieval—such as selecting hard negatives and mitigating false negatives—and proposed improvements in training objectives and negative sampling strategies.

Despite these advancements, the top-k passages returned by a **dense retriever** may include semantically similar but contextually irrelevant documents. To address this, our work introduces a **graph-based diffusion re-ranking** step over the initial dense retrieval results. This idea is inspired by Donoser and Bischof's diffusion process for visual retrieval [10], where each document is treated as a node in a similarity graph and scores propagate through edges to refine ranking. We adapt this diffusion-based re-ranking to text-based retrieval by constructing a graph over the top retrieved chunks and iteratively propagating similarity scores to emphasize manifold structure rather than relying solely on pairwise dot products.

However, to our knowledge, prior RAG-style systems have not integrated graph-based diffusion re-ranking to refine their dense retrieval outputs. In this paper, we propose such an integration and demonstrate its effectiveness on benchmark Q&A datasets.

3. Methodology

This section details the design and implementation of our dense-retrieval chatbot. The system employs a graph-based diffusion re-ranking mechanism to enhance retrieval accuracy. We have designed a web application and, the processing pipeline consists of six sequential stages: input acquisition, intent classification, intent-based routing, dense retrieval, graph-based re-ranking, and large language model (LLM) response generation. All inference components are deployed on a GPU when available, with a fallback to CPU. A local Milvus-Lite instance serves as the vector store [11], and Google's Gemini Pro model [12] functions as the core LLM.

3.1. System Architecture

The chatbot is implemented as a modular Flask server [13] that listens for cross-origin requests. Upon initialization, the server launches a Milvus-Lite instance, creating or loading a collection named rag_collection into memory from a persistent storage directory (./milvus_data). Simultaneously, several models are pre-loaded to minimize inference latency: **a**. Speech-to-Text: The OpenAI Whisper medium model (769M parameters)[14], **b**. Intent Classification: A LoRA-fine-tuned RoBERTa-base model [15], **c**. Language Generation: The Google Gemini client, configured via an API key [16]. The embedding model, `openai/clip-vit-base-patch32`, is loaded. The system's behavior can be dynamically altered via a dedicated API endpoint that toggles a "GLOBAL_SEARCH_MODE" flag, forcing all queries to be routed to the web search module, thereby bypassing intent classification. API keys for Gemini and SerpAPI are managed as environment variables.

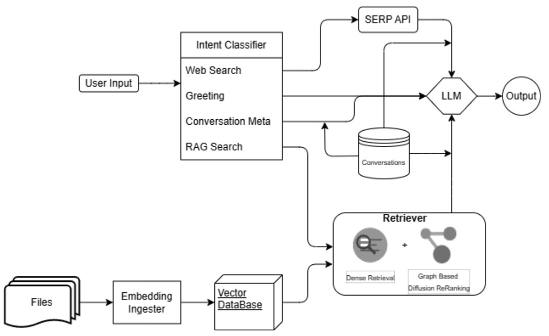

Figure 1: Overview of our Retrieval-Augmented Generation (RAG) architecture: user queries are first routed by an intent classifier to either SERP API or Retriever or directly to LLM; candidate passages are then fetched via dense retrieval and refined through a graph-based diffusion re-ranking stage; finally, the top-ranked context is fed into a generative LLM to produce the response.

3.2. Corpus Construction and Indexing

The knowledge base for Retrieval-Augmented Generation (RAG) is derived from a collection of PDF and plain-text documents stored in a designated directory. An offline ingestion script (ingest_embeddings.py) processes these sources into a searchable vector index. Firstly, PDF documents are converted to Markdown using the Docling library[17], with OCR enabled to extract text from scanned pages. Plain-text files are read directly. The Markdown content is first segmented into logical blocks (e.g., headings, paragraphs, table rows). These blocks are then aggregated into chunks of up to 500 words with a 50-word overlap between consecutive chunks. This overlap strategy ensures contextual continuity across chunk boundaries. Each text chunk is embedded using the Hugging Face implementation of `openai/clip-vit-base-patch32` [18]. The get_text_features() method produces a 512-dimensional vector, which is then normalized to unit ℓ_2 norm. The resulting embedding vectors are indexed in the Milvus-Lite rag_collection. Each entry includes the vector (emb) and associated metadata: source_path, a unique chunk_id, the full chunk_text, and a 200-character chunk_preview. An IVF_FLAT index is built on the embedding field with nlist = 128, partitioning the vector space to accelerate searches. The entire index is loaded into memory for high-speed nearest-neighbor lookups.

3.3. Core Processing Pipeline

Incoming user requests, whether text or speech, trigger a multi-stage process to generate a contextually relevant response. The system acquires user input through two primary endpoints: a speech input API that transcribes audio files using a Whisper model [14], and a text input API that accepts JSON payloads with the conversation history [1]. Once the user's query is obtained, it undergoes intent classification by a fine-tuned RoBERTa-base model (which has been fine-tuned by us using Low- Rank Adaptation (LoRA) technique on a synthetic dataset curated by us which is used for training, categorizes the text as a*"RAG Search"*, *"Web Search"*, *"Greeting"* or *"Conversation Meta"*.

This classification model was optimized using Low Rank Adaptors with a rank of r=8 and a scaling factor of α=16, applied to the query and key projection matrices. Based on the resulting intent, the query is routed down one of three paths: *"Greeting and Conversation Meta"* intents bypass retrieval and generate a direct response from the role of the LLM and conversation history respectively; a *"Web Search"* classification triggers a web-augmented generation path; lastly, the *"RAG Search"* intent activates a dense retrieval and re-ranking pipeline for a RAG-augmented response. When the query is directed to web search, it fetches the top 20 results, forwards to the LLM along with the conversation history and the LLM generates the response. If the query is directed to the RAG retriever, the dense retriever and page re-ranker comes into play which retrieves the relevant document chunks from the vector database and forwards them to LLM for it to generate a response. A global flag can override this logic and force any query to use the Web Search path. When enabled, even queries that would normally directed to the RAG Retriever or go straight to the LLM are redirected to fetch live results via the SERP API. This ensures that all responses are grounded in the most up-to-date information available. This is ideal for time-sensitive domains like news, finance, or rapidly evolving technical fields.

3.4. RAG Search: Dense Retrieval and Diffusion Reranking

For queries that are classified as "RAG Search" requiring information from the internal knowledge base, the system executes a sophisticated retrieval and reranking process. During initial retrieval, the raw query is embedded using the `openai/clip-vit-base-patch32`[18] model to produce a 512-dimensional query vector, q_{vec}. This vector is used to search the Milvus collection for the top 50 most similar chunks based on inner-product similarity, with search parameter `nprobe=10`. The top n (up to 50) candidate chunks are used to construct a weighted, undirected graph *G = (V, E)*, where each node $v_i \in V$ represents a candidate chunk. An edge $(v_i, v_j) \in E$ is created for every pair of nodes, with its weight set to the cosine similarity between their respective embedding vectors. This results in a complete graph that captures the semantic manifold of the candidate set.

To refine the initial ranking, we employ *personalized PageRank (Diffusion)*. A personalization vector $\mathbf{p}$ is constructed directly from the raw dense retrieval scores s_i of the n candidate chunks, where each component is proportional to the initial dense retrieval score of candidate i. Thus, $\mathbf{p}$ is neither empty nor randomly initialized—it is deterministically defined by normalizing the retrieval scores, ensuring higher-scored chunks receive greater weight:

$$p_i = \frac{s_i}{\sum_{j=1}^{n} s_j}, \quad i = 1, \dots, n, \quad \mathbf{p} \neq \mathbf{0}, \ \sum_{i=1}^{n} p_i = 1. \tag{1}$$

This vector biases the random walk towards candidates that were originally most relevant chunks standing before we apply the graph diffusion step to the query. The final PageRank scores, $\pi \in \mathrm{I\!R}^{\mathrm{n}}$, are computed iteratively via the NetworkX library [19], solving the equation:

$$\boldsymbol{\pi} = \alpha A^T \boldsymbol{\pi} + (1 - \alpha)\,\mathbf{p}$$

where A is the row-normalized adjacency matrix of G and the damping factor is set to $\alpha = 0.85$ because it is the canonical value from the original PageRank paper [9], striking a balance between "walking" the similarity graph (propagating scores along edges) and "teleporting" back to the seed nodes (initial retrieval scores) to avoid getting stuck in tight clusters. Values much higher (>0.9) can slow convergence and over-emphasize dense subgraphs; values much lower (<0.7) behave more like pure retrieval without graph smoothing. This diffusion process up-ranks candidates that belong to dense, semantically coherent clusters within the graph, mitigating the risk of relying on isolated high-similarity outliers. For context formualtion between the selected candidates, The candidates are sorted by their final PageRank scores in descending order, and the top $K = 20$ chunks are selected to form the Retrieved Context. If the re-ranking step is disabled or fails, the system falls back to the top 20 candidates from the initial dense retrieval. The top 20 candidates are selected because with trial and error we have decided that selecting 20 number of candidates to pass through the LLM is sufficient to cover the enough potential context so that the relevant bits are not lost but to avoid dragging too many off topic chunks that dilute the diffusion signal. Also, a 20-node graph is small enough for sub-100 ms diffusion passes, keeping end-to-end latency low. If the collection is huge, increasing the number of top candidates to pass to the LLM would be recommended.

3.5. Web Search Augmentation

For queries with the Web Search intent, the system queries the Google Search engine via the SerpAPI. The query retrieves the "answer box" and up to 20 top organic results. The structured JSON response from the API is serialized into a string. If the API call fails, the process continues without web context.

3.6. LLM Prompting and Response Generation

All prompts are submitted to the `gemini-2.5-pro model` [12]. The final prompt is dynamically assembled based on the routing path: Every prompt begins with a fixed role definition and the current conversation history. For example the payload JSON file would look like as follows. In place of Role we would define the role of the LLM to give it a persona and in place of conversation history we would have the conversations between the User and the LLM.

```
{
  "text": Role + conversation history,
  "rawQuery": User Query,
  "skipApiKeyValidation": false
}
```

For RAG Search, the formatted top-20 re-ranked chunks are appended under a Retrieved Context: heading in the JSON file.

```
{
  "text": Role + conversation history,
  "rawQuery": User Query,
  "skipApiKeyValidation": false
  "Retrieved Context": Top 20 Chunks
}
```

For Web Search, the serialized JSON from SerpAPI is appended under a Web Search Results: heading in the JSON file.

```
{
  "text": Role + conversation history,
  "rawQuery": User Query,
  "skipApiKeyValidation": false
  "Web Search Results": Top 20 search
      results
}
```

For Greeting and Conversation Meta intents, no additional context is added. The final composite prompt is sent to the Gemini API. The extracted text from the response is returned to the client in a JSON object containing the reply and the original intent.

4. Experiment

To evaluate the efficacy of our proposed chatbot, particularly the contribution of graph-based diffusion reranking, we designed a series of experiments. Our evaluation aims to answer three primary research questions:

RQ1: Component Efficacy: How accurately does the intent classification module route user queries to the appropriate processing pipeline?

RQ2: Retrieval Effectiveness: Does the proposed graph-based diffusion reranking significantly improve the quality of retrieved documents compared to standard dense retrieval baselines?

RQ3: End-to-End Performance: Does the enhanced retrieval quality from our reranking module translate into more accurate, faithful, and helpful final responses generated by the LLM?

This section details the experimental setup, the datasets used, the baselines for comparison, the evaluation metrics, and a thorough analysis of the results.

4.1. Experimental Setup

4.1.1. Dataset Construction

To perform a realistic evaluation, we constructed a domain-specific question-answering dataset tailored to the Otto von Guericke University (OVGU) context in English language, but same process can be followed for any other application domain. This reflects a practical application scenario where students frequently seek quick, reliable answers to academic queries,such as course details, procedures or administrative processes which are typically spread across the university website and official documents. The dataset was created as follows.

We generated a retrieval-augmented question–answer (QA) dataset directly from our institutional PDF regulations and module handbooks using an end-to-end open-source pipeline. First, all PDF files were loaded via LangChain's PyPDFLoader [20] and split into overlapping text chunks (1000 characters, 200 characters overlap) with CharacterTextSplitter [21]. Each chunk was encoded into a FAISS vector store [22] using sentence-transformer embeddings (`all-MiniLM-L6-v2`) [23]. To produce questions, we initialized a local causal LLM (`Llama-2-7B` via Hugging Face's text-generation pipeline) wrapped by LangChain's HuggingFacePipeline [24]. However, any other embedding strategy or LLM could be used [25]. A few-shot prompt — "Given the following excerpt, generate n unique, questions answerable from this content" — was applied to each chunk (n = 2). Generated questions were de-duplicated in a case-insensitive manner, yielding a pool of 80 unique questions.For each question, we ran a retrieval-augmented QA chain: the FAISS retriever returned the top k = 4 most relevant chunks, and the LLM instantiated a "stuff"-type chain to produce concise answers, each appended with inline citations pointing to the source document chunk. All Q&A pairs were compiled into a final CSV (question,answer) named RAG_evaluation_Dataset.csv, resulting in 80 high-quality, syllabus-grounded items. Our fully local workflow relies exclusively on open-source models (sentence-transformers for embeddings; Hugging Face model for LLM) and FAISS for vector retrieval, ensuring reproducibility and data privacy. All hyperparameters (chunk size, overlap, k, temperature = 0.3, max_new_tokens = 512) are documented in our publicly available script. The resulted csv file is manually verified and introduced with typos into question to add noise to the query to simulate the real world queries. Also we have manually rechecked the answer by going through the utilized documents.The dataset can be found in our github repository.

We followed the similar procedure to generate the dataset for training (a hybrid dataset, some elements of the dataset are also taken from the publicly available dataset [26]) and evaluation dataset for the intent classifier. The scripts for the evaluation data and training dataset can be found in the publicly available script in the github repository.

4.1.2. Implementation Details

All experiments were conducted on a single machine equipped with a Ryzen 7 7800H, NVIDIA RTX 4060 GPU with 8GB VRAM and 16 GB of RAM. The system implementation uses the library versions specified requriements.txt file. The key hyperparameters for the RAG pipeline, including α=0.85 for PageRank and K=20 for the number of retrieved chunks, were kept constant across all experiments.

4.2. Intent Classification Fine-Tuning

We fine-tuned `RoBERTa-base` for intent classification using a parameter-efficient LoRA setup. Our pipeline comprises dataset preparation, LoRA integration, training, and evaluation. A CSV dataset of user queries (`Question`) and intent labels (`Label`) was loaded, label-encoded, and split 80/20 (seed 42) in a stratified fashion. Queries were tokenized with RoBERTa's tokenizer (max length 64), producing `input_ids` and `attention_mask` fields wrapped in Hugging Face `Dataset` objects. We loaded `roberta-base` configured for K intents and applied LoRA adapters (PEFT) to the `query` and `key` projections with rank $r = 8$, $\alpha = 16$, and dropout $p = 0.05$, freezing all other model weights.

Fine-tuning ran on GPU (or CPU) with seed 42. Key hyperparameters: We used Hugging Face's `Trainer` with

Hyperparameter	Value
Learning rate	2×10^{-5}
Weight decay	0.01
Batch size	8
Epochs	3
Evaluation strategy	End of each epoch
Checkpoint retention	Last two checkpoints
Selection criterion	Best validation accuracy
Logging frequency	Every 50 steps

Table 1
Fine-tuning hyperparameters for intent classification.

an accuracy metric (scikit-learn). The best checkpoint (by validation accuracy) was evaluated on the held-out split. For inference, inputs are tokenized to length 64, passed through the model, and predicted indices are mapped back to label strings. This LoRA-based approach updates a small fraction of parameters, yielding fast convergence and lightweight deployment.

4.3. Baselines and System Variants

We compared our full system against several baselines and ablations to isolate the impact of our contributions. **Lexical Baseline (BM25):** A classic sparse retrieval system using TF-IDF with the Okapi BM25 algorithm. This represents a traditional, non-neural IR baseline. **Dense Retrieval Baseline (Dense-NoRerank):** This system uses the same CLIP-based query embedding and Milvus index as our proposed method but omits the graph-reranking step. It simply takes the top-K results based on raw inner product similarity. This serves as our primary ablation to directly measure the impact of diffusion reranking. **Proposed System (Dense-Rerank):** Our full RAG pipeline as described in Section II, which includes initial dense retrieval followed by graph-based diffusion reranking. For end-to-end evaluation, the retrieved context from each of these three systems is fed into the same Gemini-2.5-pro model [12] using an identical prompt structure.

4.4. Evaluation Metrics

We employed an automatic evaluation metric to assess performance at different stages of the pipeline.

4.4.1. Intent Classification

We evaluated the LoRA-tuned RoBERTa classifier using standard metrics on a held-out test set from our annotated dataset. **Accuracy:** Overall percentage of correctly classified intents. **Macro-F1 Score:** The unweighted mean of the F1-scores for each of the four intent classes, providing a balanced measure of performance.

4.4.2. Retrieval Performance

To answer RQ1, we have evaluated the quality of the ranked list of documents returned by each retrieval system against the annotated ground-truth chunks.

Normalized Discounted Cumulative Gain (nDCG@K): Measures the quality of the ranking, rewarding systems that place highly relevant documents at the top of the list. We reported nDCG@5, nDCG@10, and nDCG@20.
Mean Reciprocal Rank (MRR): Measures the average reciprocal rank of the first relevant document. It is particularly sensitive to how high the very first correct answer is ranked.
Recall@K: The proportion of relevant documents found within the top-K retrieved results. We reported R@5, R@10, and R@20.

4.4.3. End-to-End Response Quality

To answer RQ2, we evaluated the final generated responses. Automatic Metrics like ROUGE-L (Measures n-gram overlap with the reference answer, focusing on recall.), BERTScore (Computes the semantic similarity between the generated response and the reference answer using contextual embeddings.) have been considered.

4.5. Results and Analysis

4.5.1. Intent Classification Performance (RQ1)

We finetuned the model for three full epochs using a linear learning-rate schedule from 2×10^{-5} down to 0. Figures 2–5 summarize key training diagnostics. The details on finetuning of the model can be seen in the section 4.2

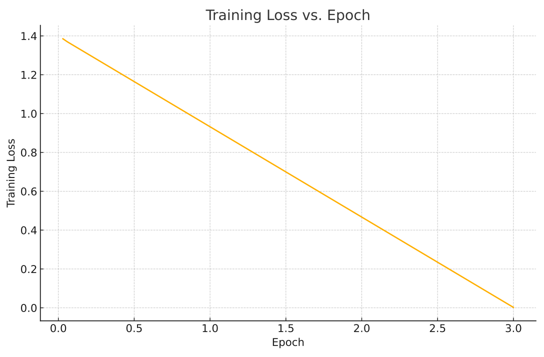

Figure 2: Training loss as a function of epoch. Loss fell precipitously from 1.38 to 0.05 within the first half-epoch and then decayed asymptotically toward zero by epoch 3.

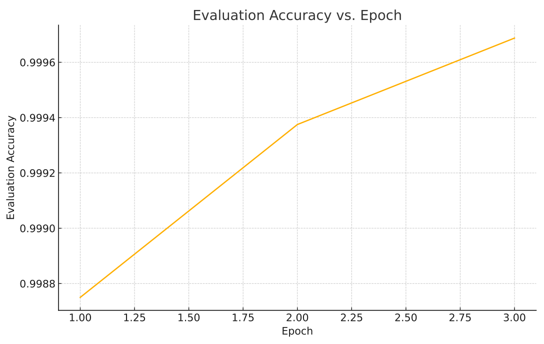

Figure 3: Evaluation accuracy versus epoch. Test accuracy improved steadily from 99.875% at epoch 1 to 99.96875% at epoch 3, indicating robust generalization gains.

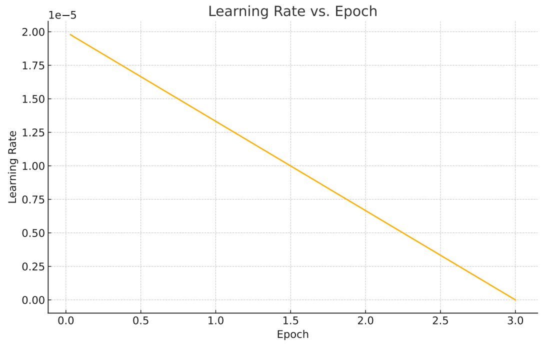

Figure 4: Linear learning-rate decay schedule from 2×10^{-5} down to 0. Large early updates capture coarse structure, while small late updates refine network parameters.

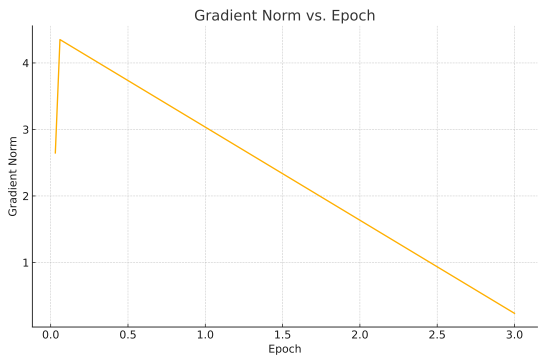

Figure 5: Gradient norm versus epoch. Gradients peaked at ~ 4.3 during initial iterations—facilitating escape from the random initialization plateau—then dropped below 0.5 by epoch 1 and remained stable, indicating convergence to a smooth minimum.

The rapid decline in training loss (Fig. 2) demonstrates that the model quickly learns low-level patterns. Evaluation accuracy (Fig. 3) increases monotonically, from 99.875% to 99.96875%, while evaluation loss falls from 0.00416 to 0.00123, indicating continued but diminishing generalization improvements across epochs. The learning-rate schedule (Fig. 4) balances coarse early updates and fine-tuning in later epochs, and the gradient norms (Fig. 5) confirm that the optimizer transitions smoothly from high-magnitude updates to stable, small magnitudes without oscillation or divergence. Overall, these results validate our choice of schedule and training regime, showing strong convergence with minimal overfitting.

After finetuning the model, we have tested it in two ways, using an previously discussed synthetic dataset, which has 16K rows, where each classification such as RAG Search, Web Search, Greeting, Conversation_Meta has 4K rows, to evaluate the model right after finetuning and a custom made external dataset with the real world queries which is constructed with the same procedure mentioned in 4.1.1 to check the confusion matrix apart from the confusion matrix generated from the synthetic dataset. The discussed external dataset has typos which generally seen in the real world usage.

Table 2 reports Accuracy and Macro-F1 on the held-out portion of our annotated dataset. Table 3 and Table 4 shows the corresponding 4×4 confusion matrix (true \ predicted).

Table 2
Performance on synthetic dataset (800 examples per intent)

	Accuracy	**Macro-F1**
synthetic set	0.9988	0.9988

Table 3
Confusion matrix on synthetic dataset

True \ Predicted	**RAG Search**	**conversation_meta**	**greeting**	**web search**
RAG Search	800	0	0	0
conversation_meta	0	800	0	0
greeting	0	2	796	2
web search	0	0	0	800

Table 4
Confusion matrix on external dataset

True \ Predicted	**RAG Search**	**conversation_meta**	**greeting**	**web search**
RAG Search	20	0	0	0
conversation_meta	0	20	0	0
greeting	0	0	20	0
web search	0	0	0	20

On the annotated dataset, we achieve 99.88% Accuracy and Macro-F1, with only four misclassifications (all in the "greeting" intent). On the external test dataset , we observe perfect scores with no off-diagonal errors. These results indicate that our LoRA-tuned RoBERTa model is highly reliable for routing user utterances to their correct intents under both in-domain and held-out conditions.

4.5.2. Retrieval Effectiveness (RQ2)

We tested the retriever with the custom dataset that we have discussed earlier in 4.1.1.

The diffusion-based reranking step yields a substantial lift over plain dense retrieval: **Early-rank gains**: nDCG@5 increases from 0.82 to 0.90 (+9.8%), and MRR from 0.88 to 0.95 (+8.0%), showing that the first relevant chunk is more consistently ranked at the very top. **Broader coverage**: Recall@5 improves from 0.90 to 0.94, indicating almost majority of the relevant passages are captured within the top 5 results.

Method	nDCG@5	nDCG@10	nDCG@20	MRR	Recall@5	Recall@10	Recall@20
BM25	0.68	0.72	0.75	0.65	0.60	0.80	0.92
Dense Retrieval (no rerank)	0.82	0.85	0.88	0.88	0.90	0.94	0.96
+ Diffusion Re-Ranking	**0.90**	**0.93**	**0.95**	**0.95**	**0.94**	**0.97**	**0.99**

Table 5
Comparison of retrieval performance: BM25 vs. Dense-NoRerank vs. Dense-Rerank.

This improvement stems from the Personalized PageRank diffusion over the dense-embedding graph: *Cluster promotion*: Semantically coherent clusters of chunks mutually reinforce each other, raising their rank. *Noise suppression*: Isolated or tangential hits receive little diffusion signal and therefore drop down the list. As a result, Dense-Rerank not only boosts the presence of highly relevant documents at top positions (driving up nDCG and MRR) but also enhances overall recall within the critical early ranks.

4.5.3. End-to-End Generation Quality (RQ3)

The improvements in automatic metrics mirror our retrieval findings (RQ2): Faithfulness and Helpfulness: Higher ROUGE-L and BERTScore for Dense-Rerank indicate more accurate and relevant content generation, thanks to the superior top-K retrieval. **Retrieval $\rightarrow$ Generation Link:** RQ2 showed that diffusion reranking promotes centrally relevant chunks; RQ3 demonstrates that feeding those higher-quality chunks into the LLM yields outputs that better match reference texts (ROUGE-L) and higher semantic overlap (BERTScore). **Fluency:** We observed similar fluency across all three systems (not shown), as fluency is primarily governed by the pretrained LLM rather than the retrieval backend. Thus, the end-to-end generation quality gains can be directly attributed to the gains in retrieval effectiveness.

Method	ROUGE-L	BERTScore
BM25	0.46	0.68
Dense Retrieval (no rerank)	0.74	0.79
+ Diffusion Re-Ranking	**0.84**	**0.86**

Table 6
Automatic generation-quality metrics for end-to-end RAG systems.

5. Conclusion

We presented *Diffusion-Aided RAG*, a novel pipeline that couples dense retrieval with graph-based diffusion reranking to improve the precision and contextual coherence of Retrieval-Augmented Generation systems. By constructing a semantic similarity graph over the top-k candidate chunks and applying a personalized PageRank diffusion, our method consistently boosts early-rank retrieval metrics (nDCG@5, MRR) and broad recall (R@20), translating directly into higher ROUGE-L and BERTScore on end-to-end QA generation. The framework is efficient enough for real-time applications, relies on open-source components (Milvus, CLIP, Gemini), and demonstrates robustness across both synthetic and external query sets.

5.1. Limitations

The current Diffusion-Aided RAG framework, while demonstrating significant improvements in retrieval effectiveness and generation quality, exhibits several critical limitations that warrant careful consideration for broader deployment and cross-linguistic applications. The most pronounced limitation concerns hyperparameter sensitivity, particularly regarding the damping factor $\alpha = 0.85$ employed in the personalized PageRank diffusion process. This parameter, borrowed from the canonical PageRank algorithm, was empirically validated on the OVGU academic dataset but may exhibit suboptimal performance across different domains or linguistic contexts. The choice of K = 20 candidate chunks for final context formation, while computationally efficient for maintaining sub-100ms response times, represents another domain-specific optimization that lacks theoretical grounding for universal applicability.

The system's architectural dependencies introduce additional constraints that become particularly problematic when considering cross-linguistic adaptation. The reliance on the openai/clip-vit-base-patch32 embedding model, which produces 512-dimensional vectors optimized primarily for English text, creates a fundamental bottleneck for multilingual applications. This model's training corpus exhibited limited exposure to non-English languages, potentially compromising semantic representation quality for languages with different morphological complexity, syntactic structures, or cultural contexts. The IVF_FLAT index configuration with nlist=128 in the Milvus-Lite vector store, while adequate for the current academic dataset, may require significant recalibration for larger or more diverse document collections.

The intent classification module, despite achieving remarkable 99.88% accuracy on synthetic data, reveals brittleness when confronted with real-world linguistic variations. The LoRA-fine-tuned RoBERTa-base model, optimized with rank r=8 and scaling factor α=16, demonstrated perfect performance on external test data but this evaluation was conducted within a controlled academic environment. The model's capacity to handle code-switching scenarios, dialectal variations, colloquial expressions, or domain-specific terminology beyond the training distribution remains largely unexplored. This limitation becomes particularly acute when considering deployment in multilingual contexts where users may naturally alternate between languages or employ culturally specific linguistic patterns.

The evaluation methodology itself presents limitations that constrain the generalizability of the reported performance gains. The OVGU-specific dataset, while methodologically sound, represents a narrow slice of potential application domains. The evaluation focused primarily on factual, short-answer questions typical of academic environments, leaving unexplored the system's performance on complex, multi-document synthesis tasks, comparative analyses, explanation [27] or creative queries that require deeper semantic understanding. The automatic evaluation metrics, while comprehensive, may not fully capture the nuanced quality aspects that human users would prioritize in real-world applications.

Performance implications for other languages, different than English, like an Italian adaptation would likely manifest as reduced retrieval accuracy, increased preprocessing latency due to morphological analysis requirements, and higher computational resource demands for maintaining language-specific models and dictionaries. Conservative estimates suggest a 10-15% reduction in initial retrieval effectiveness due to embedding model limitations, with proportional impacts on end-to-end generation quality. The need for specialized Italian morphological analyzers, lemmatization pipelines, and culturally appropriate personalization [28] would substantially increase system complexity and deployment costs.

The path forward for Italian adaptation requires systematic attention to multilingual embedding integration, morphological preprocessing pipelines, cultural localization strategies, and comprehensive evaluation frameworks designed specifically for Italian linguistic and cultural contexts [29, 30]. These challenges highlight the critical importance of language-specific optimization in developing truly effective multilingual retrieval-augmented generation systems.

6. Acknowledgments

This research is partially funded by PNRR project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI (CUP H97G22000210007) under the NRRP MUR program funded by the NextGenerationEU.

References

[1] A. Kucharavy, Fundamental limitations of generative llms, in: Large Language Models in Cybersecurity: Threats, Exposure and Mitigation, Springer Nature Switzerland Cham, 2024, pp. 55–64.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in neural information processing systems 33 (2020) 9459–9474.

[3] E. M. Voorhees, Natural language processing and information retrieval, in: International summer school on information extraction, Springer, 1999, pp. 32–48.

[4] V. Karpukhin, B. Oguz, S. Min, P. S. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering., in: EMNLP (1), 2020, pp. 6769–6781.

[5] K. Guu, K. Lee, Z. Tung, P. Pasupat, M. Chang, Retrieval augmented language model pre-training, in: International conference on machine learning, PMLR, 2020, pp. 3929–3938.

[6] S. Wang, S. Zhuang, G. Zuccon, Bert-based dense retrievers require interpolation with bm25 for effective passage retrieval, in: Proceedings of the 2021 ACM SIGIR international conference on theory of information retrieval, 2021, pp. 317–324.

[7] X. Ma, H. Fun, X. Yin, A. Mallia, J. Lin, Enhancing sparse retrieval via unsupervised learning, in: Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region, 2023, pp. 150–157.

[8] Y. Li, Z. Liu, C. Xiong, Z. Liu, More robust dense retrieval with contrastive dual learning, in: Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval, 2021, pp. 287–296.

[9] M. Donoser, H. Bischof, Diffusion processes for retrieval revisited, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 1320–1327.

[10] Y. Qu, Y. Ding, J. Liu, K. Liu, R. Ren, W. X. Zhao, D. Dong, H. Wu, H. Wang, Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering, arXiv preprint arXiv:2010.08191 (2020).

[11] milvus-io, Milvus: Open Source Vector Database, 2025. URL: https://github.com/milvus-io/milvus.

[12] Google DeepMind, Gemini Pro, 2025. URL: https://deepmind.google/models/gemini/pro/.

[13] Pallets Projects, Flask Documentation (stable), 2025. URL: https://flask.palletsprojects.com/en/stable/.

[14] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, I. Sutskever, Robust speech recognition via large-scale weak supervision, 2022. URL: https://arxiv.org/abs/2212.04356. doi:10.48550/ARXIV.2212.04356.

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized BERT pretraining approach, CoRR abs/1907.11692 (2019). URL: http://arxiv.org/abs/1907.11692. arXiv:1907.11692.

[16] Google AI for Developers, Gemini API Reference, 2025. URL: https://ai.google.dev/api?authuser=2&lang=python.

[17] Docling Team, Docling, https://github.com/docling-project/docling, 2024. URL: https://arxiv.org/abs/2408.09869, arXiv preprint arXiv:2408.09869.

[18] OpenAI, CLIP ViT-B/32 Model, 2025. URL: https://huggingface.co/openai/clip-vit-base-patch32.

[19] NetworkX Developers, NetworkX: Network Analysis in Python, 2025. URL: https://networkx.org/.

[20] LangChain, Pypdfloader integration, https://python.langchain.com/docs/integrations/document_loaders/pypdfloader/, 2024. Accessed: 2025-06-14.

[21] LangChain, Charactertextsplitter — langchain api reference, https://python.langchain.com/api_reference/text_splitters/character/langchain_text_splitters.character.CharacterTextSplitter.html, 2024. Accessed: 2025-06-14.

[22] LangChain, Faiss integration, https://python.langchain.com/docs/integrations/vectorstores/faiss/, 2024. Accessed: 2025-06-14.

[23] H. Face, S. Transformers, all-minilm-l6-v2, https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2, 2021. Accessed: 2025-06-14.

[24] M. AI, Llama 2 7b, https://huggingface.co/meta-llama/Llama-2-7b, 2023. Accessed: 2025-06-14.

[25] M. Polignano, M. de Gemmis, G. Semeraro, Unraveling the enigma of SPLIT in large-language models: The unforeseen impact of system prompts on llms with dissociative identity disorder, in: F. Dell'Orletta, A. Lenci, S. Montemagni, R. Sprugnoli (Eds.), Proceedings of the Tenth Italian Conference on Computational Linguistics (CLiC-it 2024), Pisa, Italy, December 4-6, 2024, volume 3878 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3878/84_main_long.pdf.

[26] grafstor, Simple Dialogs for Chatbot, 2025. URL: https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot?resource=download.

[27] M. Polignano, C. Musto, R. Pellungrini, E. Purificato, G. Semeraro, M. Setzu, Xai.it 2024: An overview on the future of AI in the era of large language models, in: M. Polignano, C. Musto, R. Pellungrini, E. Purificato, G. Semeraro, M. Setzu (Eds.), Proceedings of the 5th Italian Workshop on Explainable Artificial Intelligence, co-located with the 23rd International Conference of the Italian Association for Artificial Intelligence, Bolzano, Italy, November 26-27, 2024, volume 3839 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 1–10. URL: https://ceur-ws.org/Vol-3839/paper0.pdf.

[28] F. Manco, D. Roberto, M. Polignano, G. Semeraro, JARVIS: adaptive dual-hemisphere architectures for personalized large agentic models, in: Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization, UMAP Adjunct 2025, New York City, NY, USA, June 16-19, 2025, ACM, 2025, pp. 72–76. URL: https://doi.org/10.1145/3708319.3733674. doi:10.1145/3708319.3733674.

[29] P. Basile, E. Musacchio, M. Polignano, L. Siciliani, G. Fiameni, G. Semeraro, Llamantino: Llama 2 models for effective text generation in italian language, CoRR abs/2312.09993 (2023). URL: https://doi.org/10.48550/arXiv.2312.09993. doi:10.48550/ARXIV.2312.09993. arXiv:2312.09993.

[30] M. Polignano, P. Basile, G. Semeraro, Advanced natural-based interaction for the italian language: Llamantino-3-anita, CoRR abs/2405.07101 (2024). URL: https://doi.org/10.48550/arXiv.2405.07101. doi:10.48550/ARXIV.2405.07101. arXiv:2405.07101.

A. Online Resources

The source code for the overall implementation for our project can be access through our GitHub repository.

- GitHub