International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3775-3782
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0496

Distildoc: Deep Reinforcement Learning for Token-Efficient Multimodal Document Question Answering

Patchipulusu Gayathri Asritha[1], Somala Kanth Mani Sai[2] , Surampalli Bharat Sai[3], Dr. M. Sreelatha[4]
[1,2,3,4]UG Scholar, Department of CSE, R.V.R & J.C. College of Engineering, Chowdavaram, Guntur, Andhra Pradesh, India
Emails: *pgayathriasritha@gmail.com[1], kanthmani.somala@gmail.com[2], bharatsaichowdary@gmail.com[3], lathamoturicse@gmail.com[4]*

Abstract
Large Language Models (LLM) are now being used for document-based question answering across numerous areas such as healthcare, education, and science. The primary problem associated with the use of LLM's is the increase in inference costs based on the number of tokens used for input. This issue tends to become more problematic when dealing with documents that include text, structured tables, and visual content. Although retrieval-augmented generation (RAG) can improve factual grounding, it will result in higher token usage due to large, heterogeneous context. The existing solutions to this problem like LeanContext are geared towards text-only retrieval and utilize tabular Q-learning methods that will not scale well within a high dimensional multimodal environment. To address these issues, we propose a new multimodal question-answering framework called DistilDoc, which is based on deep reinforcement learning techniques designed to be token-efficient. DistilDoc uses an integrated embedding space to consolidate text chunks, each serialized row of a table, and captions of images. So, the context compression process can be modelled as a sequential decision-making problem. DistilDoc uses an off-policy Deep Q-network (DQN) with a state representation of 4608 dimensions to learn optimum compression strategies over 128 possible actions. Experimental evaluation on Arxiv papers, OpenStax textbooks, and WHO reports demonstrates that DistilDoc reduces token usage by 55.86% with a 0.6 drop in sematic similarity. These results highlight the effectiveness of Deep Q-Network on multimodal context compression in improving RAG efficiency.

Keywords: *Multimodal Question Answering, Retrieval-Augmented Generation, Deep Q-Network, Context Compression, Token Efficiency, Large Language Models, Semantic Retrieval, Adaptive Context Selection*

1. Introduction

In recent years, document-based question answering using Large Language Models (LLMs) has gained significant importance in healthcare, legal analysis, and scientific research domains. Retrieval-Augmented Generation (RAG) enhances LLM responses by including relevant domain-specific context (Lewis et al., 2020). Despite its effectiveness, RAG introduces a fundamental problem: the more tokens provided to the LLM, the more tokens must pass through the API, resulting in higher inference costs. Real-world documents contain text, tables, and images. However, existing RAG systems use a fixed retrieval size (static-k) for all queries, even though queries can have very different informational complexities. Simple queries require minimal context, while complex queries need more, leading to inefficient token usage. Lean Context (Arefeen et al., 2024) solves this static-k problem using reinforcement learning and achieves a 37-68% cost reduction. However, it is limited to text-only data and uses tabular Q-learning with discrete, cluster-based states. This makes it inappropriate for high-dimensional representations required by multimodal documents. To address these limitations, we propose DistilDoc, a multimodal question-answering framework based on a Deep Q-Network (DQN). It operates on a continuous 4608-dimensional fused state space and learns independent compression thresholds for text, tables, and images. This enables the system to select context specific to the query and also improves token efficiency across modalities.

2. Literature Review

Retrieval-Augmented Generation (RAG) has become the standard approach for grounding LLM responses

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3775-3782
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0496

with external knowledge by augmenting prompts with semantically retrieved document fragments (Lewis et al., 2020). Even though RAG improves factual accuracy by retrieving relevant document fragments, it increases token usage and inference cost. Dense retrieval models such as Sentence-BERT (Reimers & Gurevych, 2019) and SimCSE (Gao et al., 2021) improve retrieval quality but do not reduce token volume. Several approaches attempt to reduce context size before it reaches the LLM. Extractive methods such as Flan-T5 (Chung et al., 2022) generate static summaries of the context, independent of the user's query, possibly leading to over-compression for complex questions or under-compression for simple ones. Semantic Compression (Gilbert et al., 2023) uses an LLM to compress the context, but incurs an additional API cost at the time of compression [1-6]. A common flaw across these methods is the static and query-independent compression, thereby making them inappropriate for heterogeneous document collections. LeanContext (Arefeen et al., 2024) addresses this issue by framing context selection as a reinforcement learning problem and achieves significant cost savings. However, it can only be used with text-only data, depends on discrete K-means clustered states, and produces document-specific Q-tables which cannot be transferred to other domains. Deep Q-Networks (Mnih et al., 2015) extend the concept of Tabular Q-Learning by using Neural Functions on Continuous, High-Dimensional State Spaces thus enabling generalization to unseen states. Multimodal learning models such as CLIP (Radford et al., 2021) and Flamingo (Alayrac et al., 2022) enable cross-modal representation learning, but do not address query-adaptive context compression within RAG framework. Currently, there is no integrated system capable of providing multimodal retrieval, query-adaptive compression and token-efficient inference. DistilDoc aims to address this by combining text-, tabular- and image-based modalities into a single, coherent unit, using a DQN (Deep Q-Network)-based compression policy that operates over a continuous state space. This approach solves the scalability and modality issues present in previous systems.

3. Methodology

This section presents the technical formulation of DistilDoc's adaptive compression mechanism. The three modalities of content (text, tables, and images) are processed into dense vector representations within a shared semantic space. The reinforcement learning components - state representation, action space, network architecture, compression strategy, and reward function are described in the following subsections shown in Figure 1.

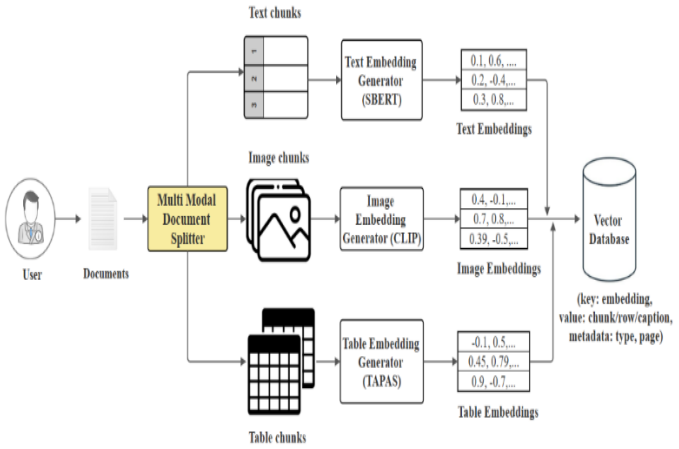

Figure 1 Overview of the DistilDoc multimodal document processing and embedding pipeline.

3.1. Overall Architecture

DistilDoc is a multimodal DistilDoc is a multimodal RAG framework in which a Deep Q-Network agent learns to adaptively compress retrieved context across three document modalities - text, tables, and images - before passing it to the LLM. The system operates in two distinct phases: an offline training phase and an online inference phase. During training, the agent interacts with retrieved content across multiple queries and episodes to learn compression policies that maximize answer quality while minimizing token usage. During inference, the agent determines compression thresholds for each modality in a single forward pass. Unlike conventional RAG systems that use a fixed retrieval size, DistilDoc dynamically selects the most relevant subset of multimodal information for each query, improving cost-effectiveness and scalability.

3.2. Multimodal Document Processing

Each input document is divided into three modalities: textual, tabular, and image data, allowing them to be processed separately but represented in a unified

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3775-3782
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0496

vector space. For the text modality, documents are divided into chunks of approximately 500 words to preserve semantic coherence and enable efficient retrieval. For the tabular modality, rows are converted into structured natural language representations (e.g., "Model is BERT, accuracy is 92%, dataset is SQuAD"), enabling semantic comparison without using separate table-specific embedding model. For the image modality, images are processed using a vision-language model to generate captions describing diagrams, charts, and figures. These captions allow images to be embedded in the same semantic space as text and tables.Processed content across all modalities is converted into dense vector embeddings using a shared pretrained model (e.g., 1536-dimensional embeddings). These embeddings are stored in a vector database along with its associated metadata, which will include the type of modality (text, table or image), page number of the source, and the number of tokens. When user enters a query, the system calculates the embedding for that query and retrieves the top-N most relevant items from each modality based on cosine similarity [7-12]. The retrieved text chunks, table rows, and image captions collectively form the initial context C. This initial context is then refined through the adaptive compression mechanism driven by the Deep Q-Network.

3.3. State Representation

To represent the current state of the agent, the embedding of the query is subtracted from the mean embedding of each modality to determine the similarity of the content from each modality to what is requested in the query. The DQN uses these representations to decide how aggressively to compress each modality. Given a query embedding $v_q \in Rd$ and the retrieved context C from Vector database, the mean embedding for each modality (v_text, v_table, and v_image) are calculated. The state is defined as:

s = [v_text - v_q ; v_table - v_q ; v_image - v_q]

This results in a state vector $s \in R3d$. (e.g., 4608 dimensions for d=1536).

Unlike LeanContext, which discretizes states using clustering, DistilDoc directly uses the continuous representation, preserving semantic detail and enabling better generalization shown in Figure 2.

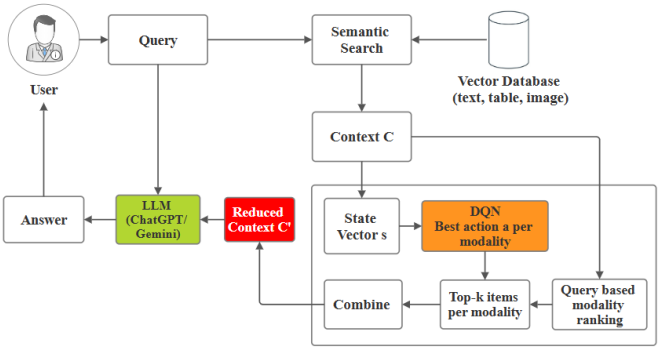

Figure 2 Overview of the system operating using the DQN to compress multimodal context into one combined context.

3.4.Action Space

In DistilDoc, the action space consists of several independent, modality-specific, discrete compression strategies. Each action is represented as a three-tuple: a = (a_text, a_table, a_image) where each element of the tuple indicates the fraction of data that will remain for the respective modality. The thresholds are defined as:

- Text: {0.05,0.10,0. 15,0.40} (8 values)
- Table: {0.10,0.20,0.30,0.40} (4 values)
- Image: {0.10,0.20,0.30,0.40} (4 values)

This gives a total action space of $|A| = 8 \times 4 \times 4 = 128$ actions. Text has finer granularity due to its larger volume, while tables and images use fewer thresholds due to their discrete nature.

3.5.Deep Q-Network Architecture

A Deep Q-Network, (DQN), underlies the functionality of the DistilDoc application. It operates by accepting a 4608-dimensional (3d) State Vector, for each of the 128 possible actions, indicating the expected reward for applying that compression strategy against the current query. The DQN's network structure is as follows:

- **Input Layer:** 4608 neurons (state vectors)
- **Hidden Layer 1:** 512 neurons (ReLU)
- **Hidden Layer 2**: 256 neurons (ReLU)
- **Output Layer:** 128 neurons (Linear)

The network approximates the action-value function Q(s, a; θ). The optimal action is selected as:

$$a^* = \arg\max_a Q(s, a; \theta)$$

Rather than storing one Q-value per (state, action)

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3775-3782
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0496

cell in a lookup table, the DQN generalises across the continuous state space using learned weights and therefore providing a sensible Q-value for a (state, action) pair that was not experienced during training.

3.6.Context Compression Strategy

When selecting an action, represented as a combination of text, table and image, a = (a_text, a_table, a_image), each of the three different modalities are compressed independently of each other. For modality m, the number of retained items is:

$$k\_m = \lceil a\_m * N\_m \rceil$$

Where: N_m = the total number of items retrieved for modality m. Items are ranked using cosine similarity with the query embedding, and only the top k_m items are retained. The resulting compressed context C′ is passed to the LLM.

3.7. Reward Function

The reward function combines two objectives: maintaining the quality of the answer when it is compressed and penalising excessive use of tokens. The reward function is formulated as follows:

$$R = \alpha * Sim(A\_red, A\_full) + \beta * ROUGE\text{-}L(A\_red, A\_full) - \gamma * \tau$$

where:

- A_red is the answer generated using the compressed context C'
- A_full is the reference answer formed using the full context C
- Sim(·,·) is the cosine similarity measured between the embeddings of the two answers
- ROUGE-L measures the lexical overlap between two answers based on the longest common subsequence
- τ = t/T, where t is the total number of tokens in the compressed context C' and T is the total number of tokens in the full context C.

The weights α, β and γ determine the relative importance of the different objectives within the reward function. Setting α = 0.7, β = 0.2, and γ = 0.1 prioritises semantic correctness while penalizing excessive token usage.

4. Training Procedure

The DQN is trained on a set of documents and queries before deployment. The agent learns compression strategies by maximizing answer quality while minimizing token usage. The complete training process is presented in Algorithm 1.

4.1.Algorithm 1: DistilDoc DQN Training

Input:

D : Documents
Q : Training queries
A : Action space
E : Episodes per query
γ : Discount factor

Output:

θ : Trained DQN parameters

- Initialize replay buffer B
- Initialize DQN parameters θ
- Initialize target network $\theta^- \leftarrow \theta$
- Extract text, table, and image modalities from D
- Embed all elements and store in vector database
- for each query $q \in Q$ do
- for episode = 1 to E do
- C ← Retrieve multimodal context for q
- v_q ← Embed(q)
- Compute v_text, v_table, v_image from C
- s ←
- [v_text - v_q ; v_table - v_q ; v_image - v_q]
- Select action a using ε-greedy policy:
- a = random action with prob ε
- else a = argmax_a Q(s, a; θ)
- C′ ← Compress C using action a
- A_red ← LLM(q, C′)
- A_full ← LLM(q, C)
- τ ← t / T
- r ← α·Sim(A_red, A_full) + β·ROUGE(A_red, A_full) − γ·τ
- Store (s, a, r) in replay buffer B
- Sample mini-batch from B
- $y \leftarrow r + \gamma \max_{a'} Q(s', a'; \theta^-)$
- Compute loss:
- $L = (y - Q(s, a; \theta))^2$
- Update θ by minimizing loss L
- Periodically update $\theta^- \leftarrow \theta$
- end for
- end for
- return θ

4.2.Data Preparation

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3775-3782
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0496

Input documents are pre-processed into text chunks, serialised table rows, and image captions as previously explained in Section 3.1. All these components are embedded and stored in the vector database prior to training. Training queries are created manually or automatically using document-specific QA pairs. For each query, the reference answer A_full is obtained from ground truth or generated once using the full context and cached to avoid repeated LLM calls.

4.3.Training Episodes

For each training query, the agent participates in E training episodes. During each episode, the agent retrieves context, constructs the state, selects an action using ε-greedy policy, compresses the context, and receives a reward. ε is initially high to encourage exploration and gradually decays to favor optimal actions [13-21]. However, as the training process continues, the ε decays, causing the agent to select the "best" compression action for each query more frequently. This reinforces the agent's understanding of the most effective compression action for each query, reducing the time spent experimenting.

4.4.Experience Replay

All transitions of the form (s, a, r) tuple are stored in a replay buffer with fixed capacity, B. During each gradient step to update the DQN parameters, a random mini-batch is selected from the replay buffer. Using samples from the replay buffer to update the DQN parameters is essential for learning, as high correlations between transition updates can cause instability in the learning process of an agent (Mnih et al., 2015)

4.5.Target Network and Optmization

A target network is used to provide stable Q-value estimates and is updated periodically. The DQN is trained using the Adam optimiser with gradient clipping to ensure stable learning. The goal of training the DQN is to minimise the amount of difference between the target and predicted values, which in turn allows the DQN to learn effective ways to compress each query over time. To optimise the DQN's parameters, the Adam optimiser is used and the gradients are clipped to mitigate instability caused by noisy reward signals from the DQN output generated by LLMs. As a consequence, the result of this will be the creation of a stable and efficient learning process for generating the compression policy.

4.6.Inference Pipeline

Once the DQN has been trained, the inference system will load the parameters θ into memory. During inference, the trained DQN selects the optimal compression action for each query using a single forward pass, followed by one LLM call to generate the final response. This two-step process allows for minimal latency and token costs while producing an optimal output, regardless of the document's complexity.

4.7.Algorithm 2: DistilDoc Inference

Input:

- D : Documents
- Q : Training queries
- A : Action space
- E : Episodes per query
- γ : Discount factor

Output:

θ : Trained DQN parameters

- Load trained DQN parameters θ
- C ← Retrieve multimodal context for query q
- v_q ← Embed(q)
- Compute v_text, v_table, v_image from C
- s ← [v_text − v_q ; v_table − v_q ; v_image − v_q]
- a* = argmax_a Q(s, a; θ)
- C′ ← Compress C using action a*
- A ← LLM(q, C′)
- return A

The pipeline requires one embedding call, one DQN forward pass, and one LLM call per query. This reduces latency and token cost by sending only compressed context C′ to the LLM.

5. Experimental Setup

5.1.Dataset

Experiments were conducted on three document corpora representing multimodal content. Arxiv contains research papers with tables, figures, and mathematical content. OpenStax provides a balanced mix of text, tables, and diagrams across Biology, Statistics, and Chemistry. WHO documents include statistical tables, country-level data, and descriptive public health text.

5.2.Baseline Models

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3775-3782
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0496

5.2.1. DistilDoc is compared to four benchmarks:

Full-context RAG (Context - Original) submits every retrieved chunk to the LLM without compression. Semantic Compression uses similarity-based filtering to remove less relevant content; however, it does not adapt dynamically to different queries. SBERT-based compression ranks content using sentence embeddings, achieving high token reduction but lower contextual understanding. LeanContext uses tabular Q-learning for adaptive text compression but is limited to text-only data.

- Evaluation Metrics
- We assess each technique according to the following four criteria:
- ROUGE-L: This metric quantifies how much of the original wording appears in the generated content. Simply, measures the lexical overlap.
- Semantic Similarity: This captures the meaning of two different responses and provides a score indicating whether two responses share the same meaning.
- Token Reduction (%): This metric measures the reduction in context size after compression.
- Response latency: Response latency measures how long it takes for a response to be generated after a user has submitted a query.

6. Implementation Details

All content is embedded using Cohere embed-v4.0 (1536 dimensions). The DQN is implemented in PyTorch with two hidden layers (512 and 256 neurons), trained using Adam (lr = 1e-4) and gradient clipping (1.0). ε-greedy exploration decays from 1.0 to 0.05. A replay buffer of size 500 and batch size 16 are used, with target network updates every 5 steps. Each query is trained for 5 episodes. The LLM used for answer generation is Groq's llama-3.1-8b-instant. All queries (training or evaluation) have the following prompt template used to secure the response: "Answer the following question based only on the provided context. If the answer is not present in the context, respond with 'Not found.' Context: {C′} Question: {Q}." Both the compressed and full-context answers are generated from the same prompt. All experiments were run on Google Colab with GPU acceleration enabled.

7. Results & Analysis

7.1. Overall Performance

Table 1 illustrates how different techniques perform in terms of token usage, semantic similarity, response latency, and cost savings. DistilDoc achieves the lowest total token usage (108 tokens) while maintaining semantic similarity of 0.47, which is comparable to Semantic Compression and close to the original full-context score of 0.53. Additionally, DistilDoc has the fastest average response latency of any of the methods evaluated (1.6 seconds), and provides the greatest cost savings (53.46%). This demonstrates an effective balance between efficiency and answer quality. The key to this efficiency is due to different compression strategy for each modality. Instead of applying fixed threshold, DistilDoc dynamically adjusts compression for text, tables and images based on their relevance to the query.

7.2.Quality versus Efficiency Trade-off

Full-context RAG gives the best semantic similarity (0.53) so has the best quality; however, it has the greatest token usage and latency (2.8s). Whereas, SBERT gives the best savings but shows a significant reduction in quality from a semantic similarity (0.41). This indicates that SBERT has reduced answer quality shown in Figure 3.

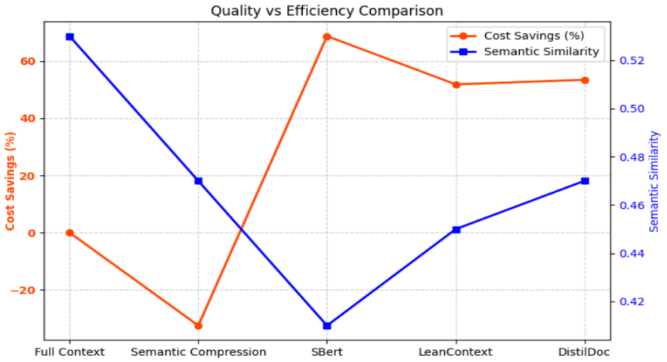

Figure 3 Trade-Off Between Cost Savings And Answer Quality (Semantic Similarity) Across Different Methods

Table 1 Comparison of DistilDoc and baseline methods on 200 random samples of Arxiv papers.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3775-3782
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0496

Text Reduction Method	Avg. Total Tokens	Avg. Prompt Tokens	Avg. Completion Tokens	Semantic Similarity	Response Latency	Cost Savings (%)
Context (Original)	243	211	32	0.53	2.8	0.00
Semantic Compression	322	113	27	0.47	3.1	-32.51
SBert	157	128	26	0.41	2.1	68.72
LeanContext	117	92	25	0.45	1.8	51.85
DistilDoc	**108**	**100**	**21**	**0.47**	**1.6**	**53.46**

LeanContext yields improvements in this trade-off, while using adaptive compression, between moderate token reductions, with a reasonable amount of semantic similarity (0.45). DistilDoc continues to improve on this balanced trade-off yielding low token usage (108 tokens), fastest response time (1.6s), and a competitive semantic similarity (0.47). Looking at the quality-efficiency graph, DistilDoc is the most closely positioned to the optimal region where both efficiency and quality are relatively high. Although the semantic similarity is slightly less than full-context RAG, it should not be considered a significant decrease in overall quality; it represents the elimination of redundant and less-important information allowing the model to focus on providing the best relevant information shown in Table 1.

Conclusion

This paper presented DistilDoc, a deep reinforcement learning framework that addresses the static-k problem in multimodal document question answering. By training a Deep Q-Network to select independent compression thresholds across text, table, and image modalities, the system moves beyond the fixed retrieval sizes and text-only limitations of prior approaches. The results show that significant token savings can be achieved without meaningful loss in answer quality, and that extending compression to structured and visual content captures information that text-only systems miss entirely. Queries answered by a figure or a results table which LeanContext cannot serve are handled naturally within the same framework, demonstrating that modality-aware compression is both practical and effective. One direction we intend to pursue is scaling DistilDoc to longer documents and larger query sets, where the DQN policy must generalise across a wider range of query types and document structures. This would more rigorously test whether the learned compression strategy holds up under conditions closer to real-world deployment, where document collections are rarely as clean or domain-consistent as curated evaluation sets.

References

[1]. Arefeen, M. A., Debnath, B., & Chakradhar, S. (2024). LeanContext: Cost-efficient domain-specific question answering using LLMs. Natural Language Processing Journal.
[2]. Alayrac, J.-B., Donahue, J., Luc, P., et al. (2022). Flamingo: A visual language model for few-shot learning. Advances in Neural Information Processing Systems.
[3]. Chase, H. (2026). LangChain. Retrieved April 1, 2026, from https://github.com/hwchase17/langchain
[4]. ChromaDB. (2026). Vector database. Retrieved April 1, 2026, from https://www.trychroma.com/
[5]. Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al., 2022. Scaling instruction-finetuned language models.
[6]. Cohere. (2026). Cohere embeddings API. Retrieved April 1, 2026, from https://cohere.com/
[7]. Cornell University. (2026). arXiv e-Print archive. Retrieved April 1, 2026, from https://arxiv.org/

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 3775-3782
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0496

[8]. Gilbert, H., Sandborn, M., Schmidt, D. C., et al. (2023). Semantic compression with large language models. arXiv preprint.
[9]. Gao, T., Yao, X., & Chen, D. (2021). SimCSE: Simple contrastive learning of sentence embeddings. Proceedings of EMNLP.
[10]. Groq. (2026). Groq LLM API. Retrieved April 1, 2026, from https://groq.com/
[11]. Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems.
[12]. Li, Y. (2023). Unlocking context constraints of LLMs: Enhancing context efficiency using self-information-based filtering. arXiv preprint.
[13]. Miller, D. (2019). Leveraging BERT for extractive text summarization on lectures. arXiv preprint.
[14]. Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518, 529–533.
[15]. OpenStax. (2026). OpenStax textbooks. Retrieved April 1, 2026, from https://openstax.org/
[16]. Pinecone. (2026). Vector database. Retrieved April 1, 2026, from https://www.pinecone.io/learn/vector-database/
[17]. Radford, A., Kim, J. W., Hallacy, C., et al. (2021). Learning transferable visual models from natural language supervision. International Conference on Machine Learning.
[18]. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. arXiv preprint.
[19]. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT Press.
[20]. Touvron, H., Martin, L., Stone, K., et al. (2023). LLaMA 2: Open foundation and fine-tuned chat models. arXiv preprint.
[21]. World Health Organization. (2026). WHO annual reports. Retrieved April 1, 2026, from https://www.who.int/