Neural Document Search and Question-Answering System

Final Report

Team Members:
Eric Xu(yx2876)
Qinhao Chen(qc2354)
Taner Sonmez(tgs2126)
Ishan Bansal(ib2585)

May 10, 2025

Contents

1 Project Overview

The project aims to build a neural document search and question-answering (Q&A) system capable of retrieving relevant information from a large corpus of documents and extracting precise answers to user queries. In other words, when a user types a question, the system first converts the question into a dense vector and retrieves the most relevant passages within a few milliseconds. It then scans those passages, identifies the exact answer span, and responds with a concise answer along with the supporting context. It consists of two key components: a retriever model that converts questions into vectors to find relevant document passages, and a reader model that identifies the exact answer span from these passages. The system uses following datasets: MS MARCO, Natural Questions, and SQuAD. It has been trained on Google Cloud using distributed GPU training and deployed with NVIDIA Triton Inference Server for low-latency response. Our retriever model achieves a Recall@5 of **49.9%** and a Recall@20 of **74.4%**. In comparison, the current state-of-the-art [1] reports **63.3%** and **75.0%** for Recall@5 and Recall@20, respectively, on the same benchmark. On the reader side, our fine-tuned model reaches **78.6%** Exact Match (EM) and **86.5%** F1 on the SQuAD v1.1 validation set—trailing the state-of-the-art performance [2] of **90.6%** EM and **95.7%** F1. **Our code is accessible via this hyperlink, all setup instructions can be found here:** Github.

2 Course Relevance

Cloud Computing

We created two virtual machines (VMs) on Google Cloud Platform (GCP) for our project. Virtual machines offered through Compute Engine, provide scalable and customizable virtualized computing resources in the cloud. GCP VMs support a wide range of configurations, including access to powerful GPUs, flexible storage options, and high-speed networking. For our neural document search and Q&A system, a GCP VM was an ideal choice because it allowed us to efficiently fine-tune large transformer models and run distributed training with GPU acceleration. It also enabled easy deployment using NVIDIA Triton Inference Server, ensuring low-latency responses. The flexibility, scalability, and integration with other GCP services made it a reliable and cost-effective platform for our project. The setup for our machine is shown in figure 4.

Deep Neural Networks

For Retriever and Reader models, we used DitillBERT. It is a lightweight, faster variant of BERT that retains 97% of its language understanding while being 60% smaller and 2× faster, making it ideal for efficient retrieval tasks.

Team Contributions

- **Eric:** Cloud ETL tasks including data cleaning, passage splitting, and Google Cloud Platform setup.
- **Qinhao:** Retriever development, including bi-encoder training, FAISS/Nearest Neighbors indexing, and retrieval evaluation.

- **Taner:** Development of the reader model—performed QA fine-tuning, EM/F1 evaluation, and error analysis, and built the Gradio demo interface.
- **Ishan:** Deployment and performance optimization using Triton Inference Server and GKE.

3 Related Work

Dense retrievers have become a central component in modern open-domain question answering (QA) systems due to their ability to map both queries and documents into a shared embedding space, enabling efficient semantic similarity search[3]. Unlike traditional sparse retrievers such as BM25[6] that rely on exact term matching, dense retrievers use neural encoders like BERT to capture deeper contextual meaning. This shift has led to notable gains in retrieval accuracy and downstream QA performance.

To address the inherent challenge of retrieving and reading relevant information, recent research has adopted a two-component architecture[4]: a retriever that selects a small set of relevant passages from a large corpus, and a reader that extracts or generates precise answers from these passages. This separation allows for modular optimization, scalability, and a better trade-off between efficiency and accuracy. Our project follows this paradigm by using DistilBERT as a retriever for fast semantic search and DistilBERT as reader for answer extraction.

Many researches [7, 5] emphasize the importance of pre-trained language models and explores robust training strategies for dense retrievers, including unsupervised pretraining, end-to-end training with QA models, and retrieval methods without aligned question-passage pairs. These ideas highlight the evolving challenges in designing scalable and accurate retrieval systems, directly influencing our decision to adopt a dual-stage approach with efficient model fine-tuning on Google Cloud. This inspired us to consider lightweight alternatives like DistilBERT and investigate modular training strategies to make our system both performant and resource-efficient.

Together, these works motivated our project design and implementation choices, shaping our focus on building a scalable, dual-component neural QA system trained and deployed efficiently on a cloud platform.

4 Design & Implementation

Retriever Model

The retriever is a bi-encoder based on DistilBERT. It independently encodes the query and passage into dense vector representations and learns to maximize similarity for true query-passage pairs while minimizing it for unrelated ones.

Architecture

- **Input formatting:** Long contexts are first cleaned and split into short passages between 80 and 300 tokens using sentence segmentation. This ensures uniform passage lengths and improves retrieval granularity.

- **Query-passage pairing:** For supervised training, we pair a query Q with a positive passage P^+ (known to be relevant) and treat all other passages in the batch as negatives. Each example thus consists of:

$$(\text{Query}, \text{Passage}_1, \ldots, \text{Passage}_B), \quad \text{with Passage}_i = P^+ \text{ iff } i = \text{label}.$$

- **Encoder (DistilBERT):** Each tower(one half of the bi-encoder architecture) of the bi-encoder is initialized with the same pre-trained DistilBERT. The token embeddings for each input are averaged across the sequence to produce fixed-size representations:

$$\text{embedding}(Q), \quad \text{embedding}(P) \in \mathbb{R}^d.$$

- **Similarity computation:** The model computes similarity via dot product between the query and all candidate passage embeddings:

$$\text{sim}(Q, P_i) = \langle \text{embedding}(Q), \text{embedding}(P_i) \rangle.$$

Training Objective For training the bi-encoder, we use a contrastive learning setup with in-batch negatives. For a given query Q in a batch of size B, we calculate the cross-entropy loss:

$$\mathcal{L} = \text{CE}(\text{sim}(Q, P_1), \ldots, \text{sim}(Q, P_B); \text{label}),$$

where the label is the index of the true positive passage. This cross-entropy loss pushes the output embedding of Q closer to its matching embedding P^+ and away from the rest.

Hyperparameters & Training

- **Datasets:** MS MARCO v2.1, SQuAD v1.1, and HotpotQA (fullwiki)
- **Model:** DistilBERT bi-encoder
- **Batch size:** 64 (with in-batch negatives)
- **Optimizer:** AdamW, learning rate 2×10^{-5}
- **Scheduler:** Polynomial decay with power 0.5
- **Precision:** Mixed-precision (AMP)
- **Epochs:** 5
- **Hardware:** Trained using one NVIDIA T4 GPU

Performance: We evaluated the retriever on the validation split of SQuAD v1.1 using Recall@k, which measures the fraction of times that a gold(ground truth) passage is present among the top-k retrieved candidates. Our retriever achieves 49.9% Recall@5 and 74.4% Recall@20.For comparison, the current state-of-the-art [1] reports 63.3% Recall@5 and 75.0% Recall@20 on

the same testing set, showing that, despite its smaller width, our retriever attains recall values close to the state of the art while requiring less memory and compute at inference time.

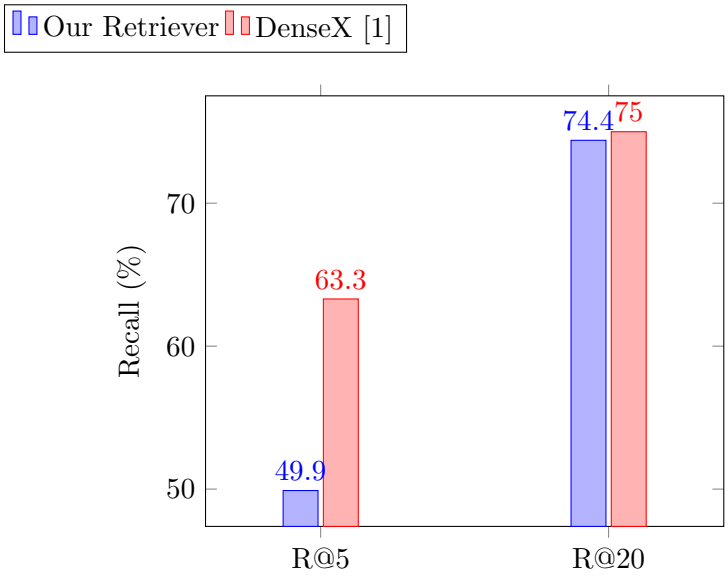

Figure 1: Retriever performance on SQuAD v1.1 validation set.

Reader Model

Our reader processes the question along with the top-K retrieved passages (here $K = 5$) in one batch, then selects the single best answer span across all contexts. In other words, it takes a question and each retrieved passage(coming from retriever) as input and produces token-level start and end scores for the answer span. Then, it selects the answer span with the highest start score + end score and converts that token range back into the final answer text.

Architecture

- **Input formatting:** We form K pairs

$$\big[\texttt{[CLS]},\, Q,\, \texttt{[SEP]},\, P_p,\, \texttt{[SEP]}\big], \quad p = 1, \ldots, 5,$$

  where each P_p is one of the top-5 passages.

- **Encoder (DistilBERT):** Each token $t_{p,i}$ in passage p is mapped to a vector

$$h_{p,i} \in \mathbb{R}^d, \quad i = 0, \ldots, N_p - 1,$$

  capturing its contextual meaning.

- **Start/End logits:** Two lightweight linear heads compute unnormalized scores:

$$\text{start\_logits}_p[i] = W_{\text{start}}^\top h_{p,i}, \quad \text{end\_logits}_p[i] = W_{\text{end}}^\top h_{p,i}.$$

  A higher logit means the model "prefers" that token as a start (or end).

- **From logits to probabilities:** We apply a softmax over each set of logits to obtain probability distribution:

$$p_{\text{start}}(p,i) = \frac{\exp(\text{start\_logits}_p[i])}{\sum_k \exp(\text{start\_logits}_p[k])}, \quad p_{\text{end}}(p,j) = \frac{\exp(\text{end\_logits}_p[j])}{\sum_k \exp(\text{end\_logits}_p[k])}.$$

  These are true probabilities (sum to 1), describing the model's confidence.

- **Span scoring & selection:** For each passage p and each valid span (i, j) with $i \leq j$, we compute

$$\text{score}(p,i,j) = \text{start\_logits}_p[i] + \text{end\_logits}_p[j] \;\approx\; \log\big(p_{\text{start}}(p,i) \times p_{\text{end}}(p,j)\big).$$

  We choose the triple (p^*, i^*, j^*) maximizing this score, then detokenize $t_{p^*,i^*} \ldots t_{p^*,j^*}$ as our answer.

Training Objective: We supervise with ground truth indices (i^*, j^*) in each passage. The loss for one example is

$$\mathcal{L} = -\log p_{\text{start}}(p,i^*) - \log p_{\text{end}}(p,j^*) = \underbrace{\text{CE}(\text{start\_logits}_p, i^*)}_{\text{penalizes low } p_{\text{start}}(i^*)} + \underbrace{\text{CE}(\text{end\_logits}_p, j^*)}_{\text{penalizes low } p_{\text{end}}(j^*)}.$$

Minimizing this negative log-likelihood pushes the model to assign high probability to the correct start and end tokens.

Hyperparameters & Training

- **Dataset:** SQuAD v1.1
- **Optimizer:** AdamW, learning rate 3×10^{-5}
- **Batch size:** 32 (mixed-precision AMP)
- **Epochs:** 2 (approximately 2 hr per epoch on T4 GPU)
- **Warmup:** Linear schedule, 10% warmup steps
- **Hardware:** Trained using one NVIDIA T4 GPU

Performance: Our fine-tuned reader achieves 78.6 % EM and 86.5 % F1 on the SQuAD v1.1 validation set, approaching state-of-the-art accuracy [2] while using a smaller model footprint and enabling faster deployment.

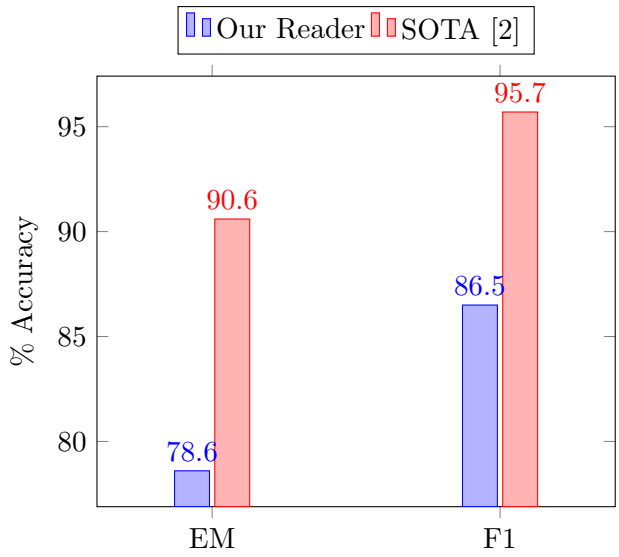

Figure 2: Comparison of Exact Match (EM) and F1 on SQuAD v1.1.

Process description: Combining retriever and reader models, appendix figure 3 illustrates our end-to-end workflow:

1. **Query:** The user asks a natural-language question.
2. **Retriever:** A DistilBERT bi-encoder embeds the question and the entire corpus, computing cosine similarity to return the top-5 most relevant passages $\{C_1, \ldots, C_5\}$.
3. **Reader:** Each pair (Q, C_i) is fed to a DistilBERT reader with span-prediction heads that output token-level start and end logits. These are converted to probabilities $p_{\text{start}}(i)$ and $p_{\text{end}}(j)$, from which the reader proposes the best span in that passage and an associated score.
4. **Selection:** Across all K passages we compare the reader scores and keep the span with the highest joint score (equivalently the largest sum of start+end logits).
5. **Answer:** The text corresponding to the winning span is detokenized and returned to the user as the system's answer.

5 Live Demo

A lightweight demo 5 is provided in Google Drive, the hyperlink(please use Columbia Email): Demo.

The folder contains all necessary files to run demo:

- `app.py` – the end-to-end Q&A interface
- `requirements.txt` – Python dependencies
- `model2/` and `final_qa_model/` – retriever and reader checkpoints
- `cleaned_query_passage_pairs.parquet`, `passage_embeds.npy` – corpus and embeddings

Local run (CPU or GPU).

```
# 1. Create a virtual environment
python -m venv venv && source venv/bin/activate

# 2. Install dependencies
pip install -r requirements.txt

# 3. Launch Gradio App
python app.py
```

This opens app in your local machine with specific url. Type a question (e.g. "What causes rainbows to form?") and press SUBMIT; the app shows the extracted answer and the top-5 supporting passages.

6 Evaluation

The empirical evaluation of the developed neural document search and question-answering system focused on assessing the individual efficacy of its core components: the dense passage retriever and the extractive answer reader. Standard benchmarks and metrics were employed to quantify their performance. The retriever component, a bi-encoder architecture based on DistilBERT, was evaluated on the SQuAD v1.1 validation set. Performance was measured using Recall@k, which indicates the proportion of queries for which a gold-standard relevant passage is ranked within the top-k retrieved documents. Our retriever demonstrated a Recall@5 of 49.9% and a Recall@20 of 74.4% using the lightweight DistilBERT architecture. The state-of-the-art DenseX [1] system reports Recall@5 and Recall@20 of 63.3% and 75.0%, respectively, on the same benchmark. The reader component, also leveraging DistilBERT with a span-prediction head, is tasked with extracting the precise answer span from the top-5 passages provided by the retriever. Its performance was benchmarked on the SQuAD v1.1 validation set using the standard metrics of Exact Match (EM) and F1-score. The fine-tuned reader achieved an EM score of 78.6% and an F1-score of 86.5% while again benefiting from the smaller footprint of DistilBERT. The state-of-the-art for reader is ANNA [2] which reported 90.6% EM and 95.7% F1. The appendix presents example queries together with the system's predicted answers and their confidence scores 6.

7 Discussion

A key advantage of the proposed system is that it attains near–state-of-the-art accuracy with a substantially smaller model, while most existing approaches depend on far larger transformer backbones. The modular retriever-reader design further facilitates independent optimization and offers flexibility for future upgrades to individual components. The utilization of Google Cloud Platform for model training and the planned deployment via NVIDIA Triton Inference Server reflect sound MLOps practices.

Notably, the DistilBERT-QA reader achieved approximately 80% EM when provided with the correct passage, even with a modest fine-tuning budget, underscoring the efficacy of transfer learning from pre-trained models. Furthermore, the performance of the Gradio demo on a Mac M1, leveraging cached embeddings and FP16 MPS support, was observed to be nearly as responsive as on a T4 GPU, highlighting the potential for efficient local execution of certain system components.

However, the evaluation also illuminated clear limitations, particularly when the system was challenged with "hard" queries requiring multi-hop reasoning, numerical computation (counting), or robust coreference resolution 7. For instance, a query such as, "Who was the second person to walk on the Moon, and what city was he born in?" resulted in an incorrect answer ("Roald Amundsen"). This failure stemmed from two primary issues: the necessity for multi-hop reasoning (identifying Buzz Aldrin, then his birthplace) and a "retrieval gap" where the indexed corpus lacked specific information about Apollo astronauts, leading the retriever to semantically similar but incorrect entities. Similarly, the query "When did she win her first Olympic gold?" failed due to unresolved coreference for "she," causing the retriever to fetch generic Olympic information. Questions requiring counting, like "How many countries share a land border with Germany?", exposed the reader's inability to perform numerical aggregation, as it is designed for span extraction. A lexical puzzle, "Which chemical element has an atomic number equal to the number of letters in its English name?", was unanswerable as it requires a combinatorial search beyond the extractive capabilities of the reader.

These "retrieval gaps" occur when the indexed corpus does not contain the specific domain knowledge (e.g., detailed astronaut biographies, lists of Nobel laureates by birth year). The reader's limitations become apparent with queries that demand more than direct span extraction, such as those involving multi-step reasoning, arithmetic, or complex logical deductions. Addressing these failures is critical for progressing towards a more robust, industry-grade QA system.

7.1 NVIDIA Triton Deployment

NVIDIA Triton Inference Server offers substantial advantages for production environments. Triton is engineered for high-performance, low-latency inference on NVIDIA GPUs, supporting concurrent execution of multiple models or model instances, thereby maximizing GPU utilization. Its framework-agnostic nature allows seamless serving of models exported to ONNX or TorchScript. Features like dynamic batching further optimize throughput.

Deployment necessitates NVIDIA GPUs with appropriate drivers, Docker, the NVIDIA Container Toolkit, the retriever and reader models exported to a Triton-compatible format (e.g., ONNX), and Triton client libraries for application interaction.

Initially, the trained PyTorch retriever and reader models are exported to an optimized format, such as ONNX, suitable for inference. Following export, a Triton model repository must be established with a specific directory structure: a root directory containing subdirectories for each model (e.g., retriever, reader in Github), each with versioned subdirectories (e.g., 1/ in Github) housing the actual model files (e.g., model.onnx). Crucially, each model directory must contain a config.pbtxt file. This protobuf text file defines metadata such as the model's

name, the backend (e.g., onnxruntime), maximum batch size, and detailed specifications for all input and output tensors, including their names, data types, and dimensions, which must precisely match those of the exported model. Pre-processing steps like text tokenization and post-processing such as converting output logits to answer spans are typically managed by the client application or can be integrated into Triton using its Python backend or ensemble features for a more self-contained pipeline. The Triton Inference Server is then launched using Docker, with the model repository mounted and GPU access enabled. Finally, a client application is developed using Triton client libraries to send inference requests (with appropriately processed inputs) to the server and handle the responses for the two-stage retrieval and reading process.

8 Conclusion & Future Work

8.1 Conclusion

This project has culminated in the development of a dual-component neural document search and question-answering system founded on DistilBERT. The system achieves Recall@20 of 74.4% for its retriever and an F1-score of 86.5% for its reader on the SQuAD v1.1 benchmark. By combining compact transformer architectures with distributed training on Google Cloud Platform and an executable deployment path via NVIDIA Triton Inference Server, the system demonstrates a practical, scalable approach to neural information retrieval and question answering. Its modular design—validated through a working Gradio demo—shows that the pipeline can efficiently retrieve relevant passages and return precise answers in real time.

8.2 Future Work

To further advance this system, enhancements to retriever accuracy could be pursued by investigating more advanced dense retrieval architectures or employing sophisticated training paradigms. A critical step would be to enrich the indexed corpus with more comprehensive knowledge sources, such as the full Wikipedia or domain-specific corpora relevant to anticipated query types, to directly address the "retrieval gaps" observed. Concurrently, expanding the training datasets for both retriever and reader models to encompass larger corpora, such as the entirety of Wikipedia, could significantly improve their generalization capabilities and the factual correctness of the answers. Hybrid retrieval approaches, combining the strengths of dense methods with traditional sparse methods like BM25[6], could also improve recall, especially for keyword-centric queries.

For the reader and overall pipeline, addressing the identified limitations is paramount. This includes developing multi-hop reasoning capabilities, potentially by implementing a pipeline that can chain sequential lookups or integrate with a small graph database for queries requiring transitive information retrieval. For questions involving numerical operations like counting or date arithmetic, dedicated post-processing scripts or specialized numeric reasoning heads could be integrated. Interactive query rewriting or robust coreference resolution modules should be explored to handle ambiguous anaphora (e.g., "she," "it") before the query is sent to the retriever.

Implementing a confidence threshold mechanism for the final system output is a key objective. This would involve calibrating the model's output scores (e.g., from the reader component) to represent true confidence levels. If the system's confidence in a generated answer falls below a predetermined threshold, it would refrain from providing a response, thereby reducing the likelihood of presenting incorrect or misleading information to the user.

Finally, enriching the system through a user feedback mechanism could enable continuous learning and model refinement. Evaluating and adapting the system for diverse and specialized domains beyond the general-purpose datasets currently used would also be a valuable extension to broaden its applicability.

References

[1] Tianyu Chen, Hongyin Wang, Shuohang Wang, Wenhao Yu, Kun Ma, Xiaochang Zhao, Hongyu Zhang, and Dong Yu. Dense x retrieval: What retrieval granularity should we use? *arXiv preprint arXiv:2312.06648*, 2023.

[2] Changwook Jun, Hansol Jang, Myoseop Sim, Hyun Kim, Jooyoung Choi, Kyungkoo Min, and Kyunghoon Bae. ANNA: Enhanced language representation for question answering. *ACL 2022 Workshop RepL4NLP Submission*, 2022.

[3] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

[4] Sascha Metzger. A deep dive into reader-retriever models + example code. https://medium.com/@saschametzger/a-deep-dive-into-reader-retriever-models-example-code-bca5c7550941, 2023. Accessed: 2025-05-07.

[5] David Obermann. In-task knowledge transfer: Applying adapterfusion to dense passage retrieval/eingereicht von david obermann. 2024.

[6] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[7] Devendra Singh Sachan. Designing accurate retrieval systems using language models. 2024.

Appendix

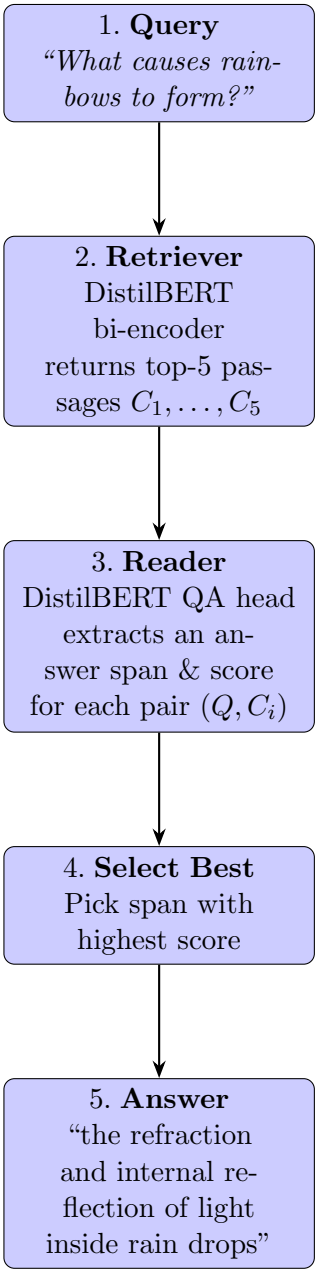

Figure 3: End-to-end pipeline

Machine configuration

Machine type	n1-standard-8 (8 vCPUs, 30 GB Memory)
CPU platform	Unknown CPU Platform
Minimum CPU platform	None
Architecture	–
vCPUs to core ratio	–
Custom visible cores	–
All-core turbo-only mode	–
Display device	Disabled Enable to use screen capturing and recording tools
GPUs	1 x NVIDIA T4
Resource policies	

Figure 4: VM Setup

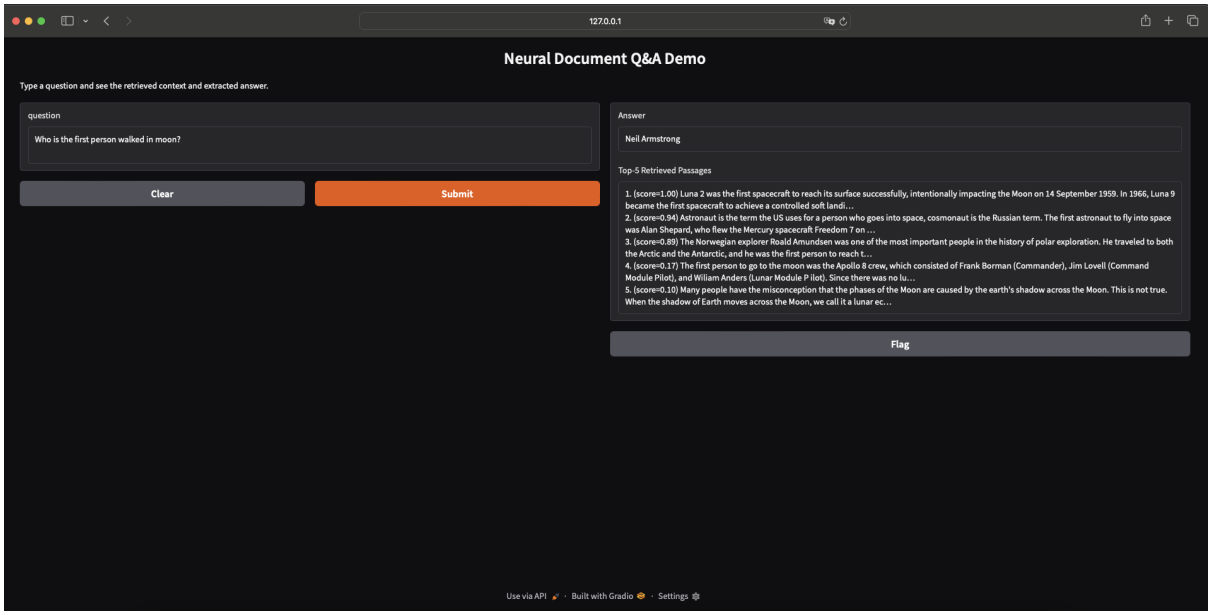

Figure 5: Demo

```
Query: What is the capital of France?
Answer: Paris (score=0.993)
From passage: France spans 643,801 square kilometres (248,573 sq mi) and has a total population of 66.7 million. It is a unitary semi-pres
idential republic with the capital in Paris, the country's largest city and main cultural and commercial centre. During the Iron Age, what
is now metropolitan France was inhabited by the Gauls, a Celtic people. The area was annexed in 51 BC by Rome, which held Gaul until 486,
when the Germanic Franks conquered the region and formed the Kingdom of France.


Query: What is the chemical formula for water?
Answer: H 2 O (score=0.974)
From passage: Answer: Yes, water is a compound. A compound forms whenever two or more atoms form chemical bonds with each other. The chemi
cal formula for water is H 2 O, which means each molecule of water consists of one oxygen atom chemically bonded to two hydrogen atoms.Thu
s, water is a compound.It's also a molecule, which is any chemical species formed by two or more atoms chemically bonded to each other. Th
e terms molecule and compound mean the same thing and can be used interchangeably.he chemical formula for water is H 2 O, which means each
molecule of water consists of one oxygen atom chemically bonded to two hydrogen atoms. Thus, water is a compound. It's also a molecule, wh
ich is any chemical species formed by two or more atoms chemically bonded to each other.


Query: How far is the Moon from Earth?
Answer: 384, 000 km (score=0.948)
From passage: As we all know, earth and moon form a part of our solar system. The surface area of the moon is 37.8 million square km and t
he surface area of the earth is 510 million square km. The moon is situated 384, 000 km away from the earth. The earth is situated 149, 66
8, 992 km (93, 000, 000 miles) from the sun. The earth s distance from the sun is conducive to life. Also, the moon does not have water, b
ut the earth has water.


Query: When did the first human walk on the Moon?
Answer: July 20, 1969 (score=0.901)
From passage: The first person to go to the moon was the Apollo 8 crew, which consisted of Frank Borman (Commander), Jim Lovell (Command M
odule Pilot), and Wiliam Anders (Lunar Module P ilot). Since there was no lunar module, Anders was tasked with taking photographs of the m
oon. It was he who shot the famous Earthrise photo from lunar orbit on Christmas Eve, 1968. The first person to land on the moon was Neil
Armstrong and Buzz Aldrin of Apollo 11 on July 20, 1969. The first person to actually walk on the moon was Armstrong, who set foot on the
surface about 6 hours after the landing.
```

Figure 6: Full Pipeline

```
Query: Who was the second person to walk on the Moon, and what city was he born in?
Answer: Roald Amundsen (score=0.598)
From passage: The Norwegian explorer Roald Amundsen was one of the most important people in the history of polar exploration. He traveled
to both the Arctic and the Antarctic, and he was the first person to reach the South Pole. (1872–1928). One of the most important figures
in the history of polar exploration was Roald Amundsen. He was the first person to reach the South Pole, the first to sail through the Nor
thwest Passage, and the first to fly over the North Pole.


Query: Which Nobel Prize in Literature laureate was born the same year the RMS Titanic sank?
Answer: Jose Echegaray (score=0.849)
From passage: The news that she had been awarded the Nobel Prize for Literature came when the Chilean poet was serving as a consul in the
city of Petropolis, Brazil. In 1945, Gabriela Mistral became the first Latin American to ever win the Nobel Prize for Literature.owever, h
received the Nobel Prize for his work as a playwright, having written 67 plays such as The Great Galeotti (1881). The choice to award the
Literature Nobel Prize to Jose Echegaray was widely criticized by other Spanish writers such as Leopoldo Alas (Clarin) and Emilia Pardo Ba
zan.


Query: When did she win her first Olympic gold?
Answer: 1896 (score=0.908)
From passage: The modern Olympics were first held in 1896 in Athens, Greece. They began after decades of increased interest in reviving th
e ancient games. Various Olympic-style games had been held starting in the 1600s, although they were small and mostly involved participant
s from the regions where they were held.enewed Interest. Interest in reviving the ancient games began to increase after the Greece's war o
f independence from the Ottoman Empire, which lasted from 1821 to 1832. In 1856, Evangelos Zappas, a wealthy Greek businessman, made an of
fer to Greece's King Otto to sponsor modern Olympic Games.


Query: How many countries share a land border with Germany?
Answer: Germany (score=0.791)
From passage: Europe is divided from Asia on the east by the water divide of the Ural Mountains, the Ural River, the Caspian Sea and the C
aucasus Mountains. The Black Sea to the southeast, the Atlantic Ocean in the west and the Arctic Ocean in the north. Europe is the second
smallest continent by surface area. Western Russia is a part of the European continent. Europe consists of the countries of Finland, Swede
n, Norway, Scotland, England, Iceland, Ireland, Portugal, Spain, France, Switzerland, Italy, Germany, Greece, Turkey, Poland, Denmark amon
g others. Between the 16th and 20th centuries, European nations controlled at various times the Americas, most of Africa, Oceania and larg
e portions of Asia.


Query: Which chemical element has an atomic number equal to the number of letters in its English name?
Answer: Z (score=0.811)
From passage: A chemical element or element is a chemical substance consisting of atoms having the same number of protons in their atomic
nuclei (i.e. the same atomic number, Z).There are 118 elements that have been identified, of which the first 94 occur naturally on Earth w
ith the remaining 24 being synthetic elements.sotopes are atoms of the same element (that is, with the same number of protons in their ato
mic nucleus), but having different numbers of neutrons. Most (66 of 94) naturally occurring elements have more than one stable isotope.
```

Figure 7: Issues Encountered: Hard questions that were answered wrongly