NLP Milestone 3 Report

- 52-3899 Karim Mohamed Gamaleldin T-01
- 52-1008 Aly Raafat AbdelFattah T-01
- 52-4509 Omar Ahmed Saad T-02

May 2025

1) Introduction:

Large language models have gotten really good at pulling answers straight from text, but they can still lose track of facts or forget what was said earlier in a conversation. Retrieval-Augmented Generation (RAG) helps by first finding the most relevant passages in a database and then letting the model answer using only that text, so responses stay accurate and on topic.

In this milestone, we use the SQuAD v1.1 dataset to compare two ways of improving QA: fine-tuning DistilBERT and crafting prompts for Phi-4 (both zero-shot and with a chain-of-thought style). We build a RAG pipeline around each model, turning it into a simple chatbot that keeps a short memory of past turns to make follow-up questions clearer. We test each setup on 100/500 sample questions, measure exact-match and overlap scores, and then discuss what works best and where to go next.

2) Dataset:

The Stanford Question Answering Dataset (SQuAD) v1.1 is a widely-adopted extractive QA benchmark built from crowd-sourced question-answer pairs on Wikipedia articles. In total it comprises:

- ≈100 000 QA pairs drawn from 536 Wikipedia articles
- Train split: 87 599 examples
- Dev split: 10 570 examples

Each example consists of:

- Context: a paragraph of contiguous text (typically 100–200 words)
- Question: a free-form natural-language query about that paragraph
- Answer: a contiguous text span within the context, with character-level offsets

Why SQuAD?

1. Diversity of topics: covers a broad range of subjects from history to science
2. Clean annotations: each answer is directly extractable, simplifying evaluation
3. Benchmark status: extensive prior work for comparison

Sampling for This Milestone

To stay within our compute budget, we randomly sample 100/500 examples from the original dev split. All experiments are evaluated on this same 100/500-example subset to ensure a fair, computationally tractable comparison.

3) Models & Enhancement Approaches:

3.1 Experiment 1: DistilBert with Fine-Tuning

For our first experiment, we use DistilBERT **(distilbert-base-uncased)**, a distilled, smaller variant of BERT that maintains ~97% of BERT’s language understanding capabilities at roughly half the size and twice the inference speed. Fine-tuning this model on SQuAD allows us to adapt it to the extractive QA task while keeping resource usage manageable.

Preprocessing Overview

- Question Cleaning: Strip leading/trailing whitespace from each question to ensure consistent tokenization.
- Joint Tokenization: Encode each question and its context paragraph together, capping at 384 tokens.

- Sliding-Window Handling: For contexts longer than 384 tokens, apply overlapping windows with a 128-token stride, so that every part of a long passage is seen by the model at least once.
- Span Alignment: Use the tokenizer's character-to-token offset mappings to convert each answer's original character start/end into precise token start/end positions. Windows that do not fully contain the answer are marked as "no answer" spans.
- Padding & Masks: Pad all inputs to the fixed length of 384 tokens and generate attention masks to distinguish real content from padding.

Training Configuration

Training was managed via Hugging Face's Trainer API with the following settings:

- Batch Size: 32 examples per device for both training and evaluation
- Learning Rate: 3×10^{-5}
- Epochs: 2 full passes over the training data
- Weight Decay: 0.01 for regularization
- Mixed Precision: bfloat16 enabled for faster computation on compatible hardware
- Platform: Google Colab (T4 GPU)

3.2 Experiment 2: Phi-4 with Zero-Shot Prompting

For our second experiment, we leverage **Phi-4**, a large open-domain LLM known for strong zero-shot capabilities on downstream tasks without any fine-tuning.

Prompt Engineering Approach

Rather than updating model weights, we rely purely on carefully crafted instructions to steer Phi-4 toward extractive QA behavior. The prompt explicitly:

- Positions the model as an “expert extractive question-answering system.”
- Restricts use to the provided context (no external knowledge).
- Demands exact reproduction of the answer span from the context.
- Handles unanswerable questions by returning “Unsure about answer.”
- Incorporates an optional “memory” field for additional background (never to be used to invent or expand beyond the context).

Prompt:

```
"""
You are an expert extractive question-answering system.
Use only the provided context to answer the question.
Always output the answer using the exact wording and phrasing as it appears in the
context.
If the answer is not contained in the context, reply “Unsure about answer.”

<context>{context}</context>

# Memory is provided for additional background—refer to it if needed, but do not
use it to invent or expand answers outside the context.
<memory>{memory}</memory>



<question>{input}</question>

'Answer:'
"""
```

→ Here {context} is replaced with the paragraph to search, {memory} with any extra notes or left blank, and {input} with the user’s question.

3.3 Experiment 2: Phi-4 with Chain of Thoughts Prompting

Prompt Engineering Approach

Chain-of-thought prompting explicitly asks the model to “think through” its answer step by step before giving a final extractive response. This can help surface relevant evidence in the context and reduce spurious matches or hallucinations.

Prompt

```
"""
        You are an expert extractive question-answering system.
        When given a context and a question, you will:
        1. Think through the relevant part of the context step by step.
        2. Show your reasoning clearly (chain-of-thought).
        3. Finally, output **only** the answer using the exact wording as it appears
in the context.
        If the answer is not contained in the context, your final answer must be
'Unsure about answer.'

        <context>{context}</context>

        # Memory is provided for additional background—refer to it if needed, but do
not use it to invent or expand answers outside the context.
        <memory>{memory}</memory>

        <question>{input}</question>

        Begin by reasoning step by step, then conclude with 'Answer: <your
extractive answer>'.
        """
```

4) Models Evaluation Approach:

To assess each model’s raw QA capability (without any multi-turn or history management), we supplied only the true context paragraph and the question, then collected the outputs for exact-match and token-overlap metrics on our 500-example subset. The results are:

System	EM (%)	F1 (%)	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
DistilBERT	0.2	7.83	0.0719	0.0307	0.0696	0.0125
DistilBERT + Fine Tuning	74.6	83.40	0.7596	0.5055	0.7588	0.4592
Phi-4 zero-shot prompting	17.8	47.87	0.4338	0.3075	0.4324	0.1081
Phi-4 chain-of-thought prompting	59.0	76.18	0.7104	0.5166	0.7098	0.3184

Comments:

- Fine-tuned DistilBERT achieves the highest raw QA scores (EM 74.6 %, F1 83.4 %), confirming that task-specific gradient updates are crucial for extractive QA.
- DistilBERT without fine-tuning (i.e., vanilla model) yields near-zero EM/F1, underscoring that language-model pretraining alone isn't sufficient for pinpointing answer spans.
- Phi-4 zero-shot delivers modest out-of-the-box performance, showing that large LLMs can approximate QA but benefit from explicit guidance.
- Chain-of-thought prompting on Phi-4 dramatically narrows the gap to the fine-tuned baseline, illustrating how intermediate reasoning steps help the model focus on the correct context.
- When EM is much lower than F1, like in the zero-shot case, it means the model often gets some of the right words but doesn't reproduce the exact answer verbatim.

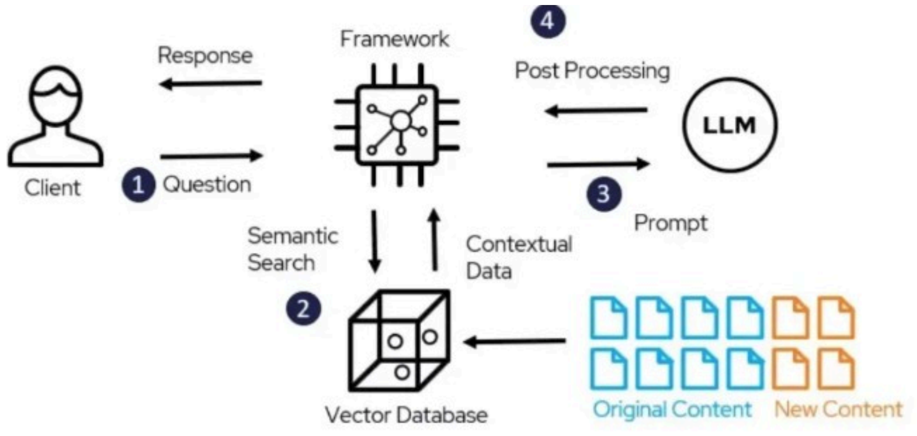
5) Retrieval-Augmented Generation (RAG) System:

5.1 Index Construction

- Embedding model: **intfloat/multilingual-e5-large-instruct** to encode the SQuAD validation set context into a dense vector.
- Vector store: **ChromaDB**, holding all dev-split paragraphs.
- Retrieval parameter: K=3 nearest neighbors per query.

5.2 RAG Pipeline Design

- Question embedding: Compute a vector for the input question using the same e5-large-instruct encoder.
- Context retrieval: Query ChromaDB for the top-3 most similar context embeddings.
- Answer extraction: For each of our four models, feed the question plus each retrieved context into the QA module:
  - DistilBERT (fine-tuned)
  - Phi-4 zero-shot prompt
  - Phi-4 CoT prompt
- Answer output: Take the output from the LLM, do any needed post processing and give it back to the user



6) RAG Evaluation:

- Below are the retrieval-augmented QA scores, computed on a randomly sampled 100-example subset of the SQuAD v1.1 validation split:
- Metrics:
  - Exact Match (EM)
  - F1
  - ROUGE-1
  - ROUGE-2
  - ROUGE-L
  - BLEU

System	EM (%)	F1 (%)	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
DistilBERT (no fine-tuning) w/ RAG	0.0	1.74	0.0141	0.0036	0.0141	0.0054
DistilBERT (fine-tuned) w/ RAG	70.0	73.28	0.6875	0.4450	0.6873	0.5423
Phi-4 zero-shot prompting w/ RAG	9.0	35.40	0.3288	0.2296	0.3286	0.0697
Phi-4 chain-of-thought w/ RAG	50.0	65.75	0.6317	0.4457	0.6298	0.2573

→A similar ranking between the different models is seen. DistilBERT fine tuned and Phi 4 with chain of thoughts are the best 2 models. The decrease in performance in all models could be because of:

1. Retrieving the wrong context from the vector database
2. The text become to long so it is harder for the model to extract the correct answer

In Conclusion: A model can’t answer correctly if it doesn’t see the right text.

7) Chatbot System Design:

In this section, we describe how each QA model is wrapped into a multi-turn conversational interface, with a focus on how dialogue history (**memory**) is managed, injected into prompts, and used to refine user queries.

7.1 Memory Management

❖ Fixed-length memory: We maintain the last N turns (default N = 10) in a Python deque, ensuring older context is automatically dropped once capacity is reached.

❖ The memory is passed as a separate {memory} variable into our retrieval prompt template, so the LLM can "remember" the flow of the conversation when deciding how to retrieve and answer.

❖ After the model produces its response, both the user input and the model's reply are appended back to the memory buffer.

7.2 Query Refinement via Memory

❖ In multi-turn chat, follow-up questions often rely implicitly on the previous context.

❖ Mechanism: If memory is non-empty, we first rephrase the user's latest message by calling the secondary LLM (refinement_llm) on a template that includes:

- ➢ the raw user query
- ➢ the full memory string of past turns.

Prompt:

```
"""
You are a Query Refinement Highly Intelligent Agent.
Your job is to rewrite a user's question, incorporating relevant details from the
conversation memory, into a concise, keyword-rich search query optimized for
retrieving SQuAD passages from a vector database.
```

```
If the user's query is not related to any information in the conversation memory,
do NOT modify it and just return the original query unchanged.
Respond with **only** the rewritten query or original query—no explanations,
please.

User query:
{user_query}

Conversation memory:
{memory_context}
"""
```

❖ Outcome of this step is: A refined query is more explicit before retrieval, improving the relevance of returned contexts.

8) Conclusion:

In this milestone, we saw that fine-tuning DistilBERT on SQuAD gives the highest accuracy when the exact context is provided (EM 74.6 %, F1 83.4 %) and remains strong under RAG (EM 70 %, F1 73.3 %). By contrast, the out-of-the-box DistilBERT model fails almost entirely, even with retrieval, which shows how crucial task-specific training is for precise answer-span extraction. Phi-4 zero-shot prompting sits in the middle, delivering modest performance (EM 17.8 %, F1 47.9 % without RAG; EM 9 %, F1 35.4 % with RAG), while chain-of-thought prompting unlocks much of its potential (EM 60 %, F1 76.2 % without RAG; EM 50 %, F1 65.8 % with RAG). This demonstrates that asking the model to “think aloud” can nearly match fine-tuning, provided the retriever surfaces the right paragraphs.

Our results highlight two main trade-offs:

- **Fine-tuning** requires labeled data and compute, but delivers the most reliable accuracy.
- **Prompt engineering** (especially CoT) is fast and flexible, yet remains inferior when using Phi-4 with it

Looking ahead, there are several promising directions:

- **Stronger LLMs**, for example ChatGPT or Gemini using their API can give us better results than the 14 billion parameter Phi-4.
- **Stronger retrieval**, for example by fine-tuning the embedding model or adding a lightweight reranker to boost recall of relevant contexts;
- **Hybrid approaches**, combining a bit of fine-tuning with chain-of-thought prompts to balance performance and resource demands;

By improving retrieval and exploring mixed training-and-prompting strategies, we can further narrow the gap between heavy fine-tuning and prompt engineered workflows, paving the way for efficient, accurate, and natural multi-turn QA systems.

References

1. Chase, H. **LangChain: A Framework for Developing LLM Applications**. GitHub repository. Available: Link
2. Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). **SQuAD: 100,000+ Questions for Machine Comprehension of Text**. Available: Link
3. **Chroma DB**: An embeddable vector database for AI applications. Available: Link.
4. Poulopoulos, D. **bert-qa-finetuning**. GitHub repository. Available: Link
5. **RAG Architecture**. Deepchecks Glossary. Available: Link