COMP 652 Final Project Report:

Group E: Jake Cox, Nolan DeBord, Alan Huang, Nina Luong, Arya Rahmanian, Chenjie Yu

I. ABSTRACT

This project investigates extractive question answering (QA), where the goal is to locate an answer within a given context passage. Using the Stanford Question Answering Dataset (SQuAD v1.1) [1], we evaluate a diverse range of models, including both classic and transformer-based neural architectures. These include Long Short-Term Memory (LSTM) models—with and without the dual-flow attention mechanism introduced in [2]—pre-trained transformers such as Bidirectional Encoder Representations from Transformers (BERT) [3], Distilled BERT (DistilBERT) [4], and more generative-oriented models like Generative Pretrained Transformer-2 (GPT-2) [5] and Bidirectional and Auto-Regressive Transformers/Text-to-Text Transfer Transformer (BART/T5) [6]. We explore the performance trade-offs between lightweight and large-scale models in terms of accuracy, efficiency, and architectural complexity. All models are evaluated using F1 score on the SQuAD v1.1 benchmark. This comparative analysis provides insight into the evolving landscape of QA systems and offers practical guidance on selecting and deploying natural language processing (NLP) models in real-world applications.

II. INTRODUCTION

Question answering (QA) is a fundamental problem in natural language processing (NLP), enabling systems to extract relevant answers from unstructured text given a natural-language query. QA tasks span several types such as extractive QA, where the answer is a span selected from the context [1]; abstractive QA, where the system generates a free-form answer [7]; and open-domain QA, where the system retrieves evidence from a large corpus to answer general knowledge questions [8]. These tasks have become increasingly important in modern applications such as search engines, virtual assistants, and automated customer support systems, with examples including DrQA for open-domain QA and retrieval [8], and BERT-based models for query understanding and ranking in web search [3].

This project focuses on extractive QA using the SQuAD v1.1 dataset [1], a widely used benchmark comprising over 100,000 questions based on Wikipedia articles. For each question, the model must select a section of text from the accompanying paragraph that best answers the question.

To explore the relative strengths of different model types, this project evaluates a range of neural architectures, including both classic recurrent models and modern transformer-based approaches.

- Recurrent neural networks (RNNs)**:** Specifically, Long Short-Term Memory (LSTM) models, both with and without bidirectional/dual-flow attention mechanisms [2].
- Transformer-based models: Including BERT [3] and DistilBERT [4], which use self-attention to capture context.
- Autoregressive and generative transformers: Such as GPT-2 [5] and GPT-based APIs that generate answers from context.
- Encoder-decoder models: Including BART and T5 [6], which reformulate QA as a sequence-to-sequence task.

III. PRIOR WORK

Extractive question answering (QA) has seen rapid progress with the development of increasingly powerful neural architectures. One of the earliest neural models evaluated on SQuAD was the Match-LSTM with Answer Pointer [9], which introduced alignment mechanisms between passage and question and helped establish span prediction as the dominant formulation for extractive QA. Later, systems based on recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) networks with attention mechanisms, demonstrated strong performance by modeling contextual relationships between the question and the passage [2]. These approaches laid the foundation for span-based QA by introducing architectures capable of aligning question and context representations effectively.

The introduction of transformer-based models marked a major breakthrough for QA and other NLP tasks. Models such as BERT [3] and DistilBERT [4] achieved state-of-the-art results by using self-attention mechanisms to capture bidirectional context throughout passages, eliminating the need for recurrence. Building on these foundations, more recent models such as LUKE [10], which incorporates entity-aware self-attention, have further advanced the field by integrating structured knowledge about named entities. These innovations have driven substantial improvements in benchmark performance, as reflected in the SQuAD v1.1 leaderboard [11].

Building on these advances, this project implements and compares representative RNN-based and transformer-based models on extractive QA tasks to provide a systematic evaluation of their strengths and limitations.

IV. METHODOLOGY

A. Data Splitting and Evaluation Metric

Each model is evaluated using the standard SQuAD v1.1 train/development split. To further optimize hyperparameters, the original training data was partitioned into a training subset (90%) and a validation subset (10%) using a fixed random seed to ensure reproducibility. The official SQuAD validation set was used as the test set in the final evaluations.

Model performance is reported using two standard SQuAD evaluation metrics: exact match (EM) and F1 score. The

EM metric measures the percentage of predictions that match any of the ground-truth answers exactly, while the F1 score reflects the harmonic mean of token-level precision and recall. Together, these metrics provide a robust assessment of both strict and partial match quality in extractive QA tasks.

By comparing performance across architectures, this project aims to provide a comprehensive evaluation of model efficiency and effectiveness for extractive QA.

B. Recurrent Neural Networks (RNNs)

Two RNN-based models using LSTMs were developed: a plain BiLSTM and an extended version with bidirectional attention, inspired by the BiDAF [2] architecture's strong performance on SQuAD v1.1.

1) Data Preprocessing and Batching: A preprocessing pipeline was implemented to prepare the SQuAD dataset. Pre-trained GloVe embeddings (100 and 300 dimensions) [12] were loaded, and a vocabulary was constructed from tokenized training data, with special <pad>and <unk>tokens included. Context and question texts were tokenized and mapped to vocabulary indices, with unknown tokens mapped to <unk>. Character-level answer spans provided in SQuAD were converted to token-level start and end indices. Examples in which the answer was truncated due to the maximum context length were excluded to maintain alignment quality. A custom collate function was used to pad sequences and batch inputs efficiently, ensuring consistent input dimensions during training. Finally, the data was wrapped into PyTorch Dataset objects for integration with the training pipeline.

2) Model Architecture: Both models used pre-trained GloVe embeddings to initialize the embedding layer, providing rich semantic information without the need to learn embeddings from scratch. To improve generalization, dropout was applied throughout the network, and tuned learning rates with early stopping were used to improve training stability and reduce overfitting.

- Baseline BiLSTM (Plain LSTM): This model consists of independent bidirectional LSTM encoders for the context and question, producing hidden representations that capture bidirectional dependencies. These representations are passed to span prediction layers to estimate the start and end indices of the answer span.
- BiDAF-style Attention Model: This architecture extends the baseline by introducing a bidirectional attention mechanism that computes context-to-question (C2Q) and question-to-context (Q2C) attentions. As outlined in the original paper [2], BiDAF applies attention between the context and query without using self-attention within sequences, unlike transformers. Compared to the cross-attention in encoder-decoder models [13], BiDAF relies on fixed similarity functions, applies attention only once, and does not involve query-key-value projections. Figure 1 illustrates the architecture, highlighting the flow across embedding, attention, modeling, and prediction layers, and the dual attention paths that enable detailed context-question interaction.

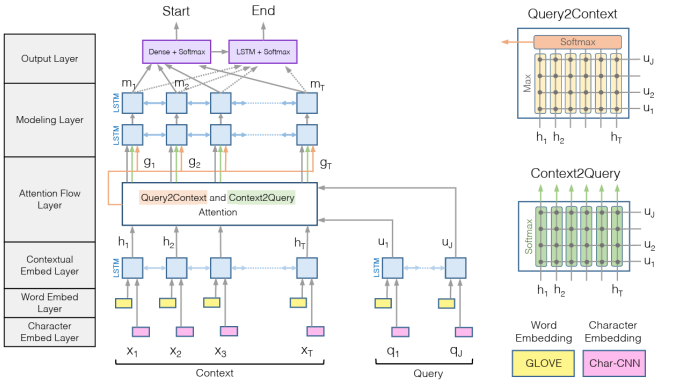
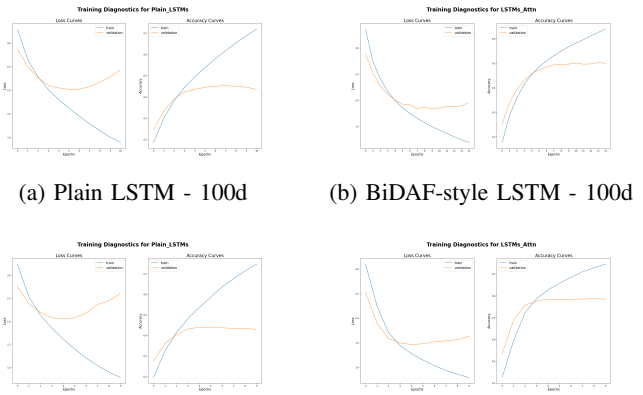


Figure 1: Architecture of the BiDAF (Bidirectional Attention Flow) model, adapted from [2]. The figure shows the embedding, attention, modeling, and output layers, highlighting the context-to-query and query-to-context attention flows that allow fine-grained interaction between the context and question.

The model computes a similarity matrix between all context-query word pairs to guide attention. C2Q attention identifies the most relevant question words for each context token, while Q2C attention highlights the context tokens most aligned with the overall question. These outputs are fused with the context representation via concatenation and element-wise multiplication, then processed through stacked BiLSTM layers with residual connections and layer normalization. Notably, this implementation omits the character embedding layer from the original design and adds layer normalization and dropout for improved stability.

3) Training: We conducted training experiments for both the baseline BiLSTM model and the BiDAF-style attention model using the SQuAD v1.1 dataset and GloVe embeddings (100d and 300d). To optimize hyperparameters, we employed Optuna, a modern hyperparameter optimization framework that combines Bayesian search and pruning strategies [14]. Optuna was used to tune key settings, including learning rate, dropout rate, batch size, weight decay, and hidden size, ensuring robust performance across configurations. The CrossEntropyLoss function was used for training. Figure 2 presents the training and validation loss and accuracy curves for the two models using GloVe embeddings of 100 and 300 dimensions. The BiDAF-style models (Figures 2b and 2d) consistently show faster convergence, lower validation loss, and higher validation accuracy compared to the plain BiLSTM models (Figures 2a and 2c), highlighting the benefits of the bidirectional attention mechanism in improving generalization and reducing overfitting. For example, with 300-dimensional embeddings, the BiDAF model achieved a substantially lower validation loss (1.63 vs. 2.61) and a higher exact match score (51.3% vs. 37.7%) compared to the plain BiLSTM.

Within the BiDAF models, the 300-dimensional version outperforms the 100-dimensional one, demonstrating smoother convergence, lower loss, and higher validation accuracy due



(a) Plain LSTM - 100d (b) BiDAF-style LSTM - 100d

(c) Plain LSTM - 300d (d) BiDAF-style LSTM - 300d

Figure 2: Training and validation loss and accuracy curves for the BiLSTM and BiDAF-style models with GloVe 100d and 300d embeddings.

to the richer semantic information of larger embeddings. Although the BiDAF-300d setup required slightly longer training time (approximately 6,015 seconds) compared to the plain LSTM (4,712 seconds), this trade-off resulted in meaningful accuracy improvements.

The detailed training results, including optimized hyperparameters and validation metrics, are summarized in Table I. Together, these patterns highlight that both architectural improvements (via dual-flow attention) and embedding richness contribute significantly to overall model performance on extractive QA tasks.

Table I: Summary of training results for plain LSTM and BiDAF-style LSTM models with GloVe 100d and 300d embeddings, reporting validation loss, validation exact match (EM), validation F1 score, and training time.

Model	Val Loss	Val EM (%)	Val F1 (%)	Training Time (s)
Plain LSTM 100d	2.44	37.64	51.25	5284.56
BiDAF LSTM 100d	1.96	43.91	57.92	4123.37
Plain LSTM 300d	2.61	37.74	50.50	4712.45
BiDAF LSTM 300d	1.63	51.26	66.48	6014.87

4) Evaluation on Test Set: The final models were evaluated on the SQuAD v1.1 validation set to assess generalization performance. As shown in Table II, the BiDAF-style LSTM with 300d embeddings achieved the best overall performance, with the highest EM and F1 scores. With consistently stronger accuracy across both the validation and test sets, it shows the clear benefits of combining bidirectional attention mechanisms with richer embeddings in extractive QA tasks. In contrast, the plain LSTM, particularly with 100d embeddings, showed lower scores, underscoring the limitations of models without attention.

Given its consistently strong results, the BiDAF-300d model serves as the representative RNN baseline for comparison against transformer-based models in the next analysis stage.

Table II: Summary of test set evaluation results for plain LSTM and BiDAF-style LSTM models with GloVe 100d and 300d embeddings, reporting exact match (EM) and F1 scores.

Model	GloVe Dim	Test EM (%)	Test F1 (%)
Plain LSTM	100d	44.40	55.22
BiDAF LSTM	100d	50.14	61.47
Plain LSTM	300d	43.19	54.26
BiDAF LSTM	300d	58.45	70.26

C. BERT & DistilBERT

Two variations of Google's BERT (Bidirectional Encoder Representations from Transformers) [15] model—BERT and distilBERT—were evaluated on the SQuAD v1.1 dataset. BERT was among the first deeply bidirectional models pre-trained in an unsupervised manner on a large corpus of plain text. Although bidirectional architectures and transformers had previously existed in NLP, BERT's key innovation was integrating deeply bidirectional representations **with Transformers**. In contrast, previous word embeddings, such as GloVe, are unable to capture context and prone to misinterpretation of words with multiple meanings. BERT addresses these shortcomings by utilizing extensive unsupervised pre-training, facilitating effective transfer learning.

As illustrated in 3, each input token (including the special [CLS] and [SEP] markers) is first embedded and then processed by a stack of transformer encoder layers. At the top of this stack, each token's hidden state becomes a **contextualized embedding**, meaning it reflects not the token itself and the information from every other token in the sequence. The [CLS] token's final embedding is commonly used for downstream classification tasks, while all other token embeddings carry bidirectional context for prediction. This pipeline allows BERT to dynamically adjust each token's representation based on its surrounding words—something static embeddings like GloVe cannot achieve.

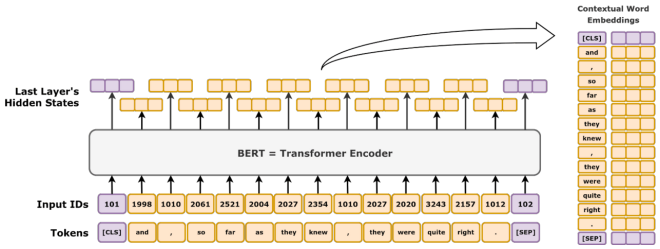
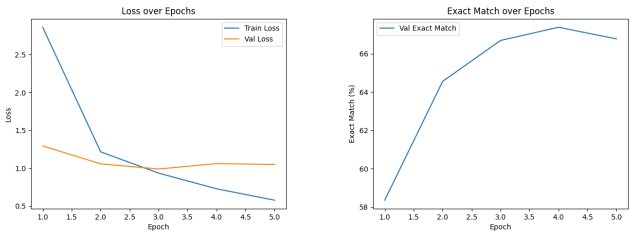
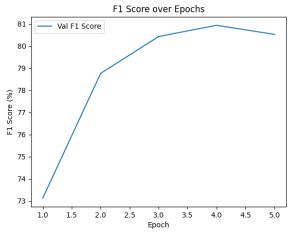


Figure 3: High-level schematic diagram of BERT

1) Data Pre-Processing: The work done to pre-process SQuAD data involved tokenizing each question-context pair with **Hugging Face's AutoTokenizer** [16]. The context was truncated to accommodate for BERT's maximum context window, 512 tokens, along with a 128 token stride when necessary. When the tokenizer splits contexts into multiple segments



(a) BERT Loss Plot (b) BERT Exact Match



(c) BERT F1

Figure 4: Training loss & Validation Loss, F1, and EM while training.

to maintain the 512 token context window, additional logic was added to check whether or not the answer was not fully contained in the tokenized window. If so, the answer would be marked as "no answer" to prevent incorrect training and to allow the model to accurately infer based off of input context. The tokenized train, validation, and test sets were batched and wrapped in PyTorch DataLoaders.

2) BERT:

a) Training:

- **Model:** BERT-base (110M parameters)
- **Epochs:** 5
- **Optimizer:** AdamW with learning rate 10^{-5}
- **Scheduler:** Linear schedule with warm-up (10% of total training steps)
- **Regularization Methods:** Dropout, early stopping based on validation loss
- **Hardware:** 1 NVIDIA RTX 4090 GPU
- **Training Duration:** Approximately 15 minutes per epoch (total $\sim$1 hour)

The best-performing checkpoint (epoch 3) was selected for final evaluation, supported by observing the divergence between training and validation loss in Figure 4a, along with the plateauing of F1 and EM scores in Figures 4b and 4c.

b) BERT-base Results: The fine-tuned BERT-base model achieved an Exact Match (EM) score of **78.97%** and an F1 score of **86.77%** on the evaluation dataset.

Below are some examples from the test set:

- **Example #1:**

  Question: **What are environmentalists concerned about having released from the Amazon region?**
  Context: Environmentalists are concerned about loss of biodiversity that will result from destruction of the forest, and also about the release of the carbon contained within the vegetation, which could accelerate climate change.
  Predicted Answer: "carbon"
  Ground Truth: ["carbon contained within the vegetation", "carbon contained within the vegetation", "carbon"]

  Analysis: Although the model provided a concise answer correctly identifying the element (carbon), it lacked the full descriptive detail.

- **Example #2:**

  Question: **What delimits the delta of the Rhine in the East?**
  Context: The mouth of the Rhine into Lake Constance forms an inland delta. The delta is delimited in the West by the Alter Rhein ("Old Rhine") and in the East by a modern canalized section.
  Predicted Answer: "a modern canalized section"
  Ground Truth: ["modern canalized section", "modern canalized", "modern canalized section", "canalized section"]

  Analysis: The prediction exactly matched the ground truth.

- **Example #3:**

  Question: **Where was the disease spreading between 1348 and 1350?**
  Context: From Italy, the disease spread northwest across Europe, striking France, Spain, Portugal and England by June 1348, then turned and spread east through Germany and Scandinavia from 1348 to 1350.
  Predicted Answer: "northwestern Russia"
  Ground Truth: ["Germany and Scandinavia", "Germany and Scandinavia", "Germany and Scandinavia"]

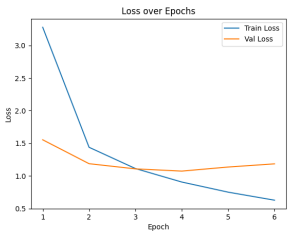
  Analysis: This prediction represents a clear error, as the model incorrectly identified a region not mentioned within the relevant context. It is possible that the model may have become confused by information located outside its immediate context window or that an error occurred during preprocessing, such as misaligned tokenization or truncation of relevant text.

Overall, while BERT performs effectively in capturing relevant keywords and phrases in most scenarios, certain predicted answers highlight room for improvement in contextual comprehension.

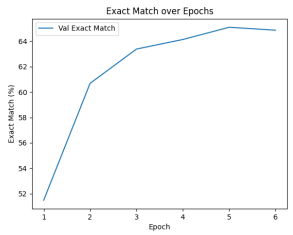
3) distilBERT:

a) Training:

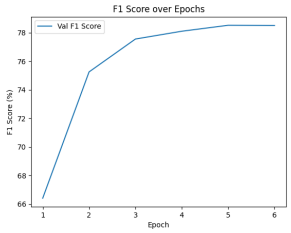
- **Model:** distilBERT-base (67M parameters)
- **Epochs:** 6
- **Optimizer:** AdamW with learning rate 10^{-5}
- **Scheduler:** Linear schedule with warm-up (10% of total training steps)
- **Regularization Methods:** Dropout, early stopping based on validation loss



(a) distilBERT Loss Plot



(b) distilBERT Exact Match



(c) distilBERT F1

Figure 5: Training loss & Validation Loss, F1, and EM while training distilBERT

- **Hardware:** 1 NVIDIA RTX 4090
- **Training Duration:** Approximately 9.25 minutes per epoch (total ∼46.2 minutes)

The best-performing checkpoint (epoch 4) was selected for final evaluation, supported by observing the divergence between training and validation loss in Figure 5a, along with the plateauing of F1 and EM scores in Figures 5a and 5c. The training was similar as to BERT where the best epoch was 3-4 and trained 25% faster on the same hardware. This is in line in what was expected since distilBERT is 60% the size of BERT.

b) distilBERT Results: The fine-tuned distilBERT-base model achieved an Exact Match (EM) score of **75.7%** and an F1 score of **84.4%** on the evaluation dataset.

- **Example #1:**

  Question: **What is the clarity of liquid oxygen?**
  Context: Oxygen condenses at 90.20 K (-182.95 °C, -297.31 °F), and freezes at 54.36 K (-218.79 °C, -361.82 °F). Both liquid and solid O_2 are clear substances with a light sky-blue color caused by absorption in the red (in contrast with the blue color of the sky, which is due to Rayleigh scattering of blue light). High-purity liquid O_2 is usually obtained by the fractional distillation of liquefied air. ...
  Predicted Answer: "90.20 K"
  Ground Truth: ["clear", "clear substances with a light sky-blue color", "clear", "clear", "clear"]

  Analysis: The model has mistaken the numeric condensation temperature for the requested qualitative property. Instead of identifying the adjective "clear", it returned "90.20 K," which is irrelevant to the question of clarity. This possibly suggests the model has over-prioritized numeric tokens in proximity to its keyword "liquid oxygen" rather than the descriptive context.

- **Example #9828:**

  Question: **Western Imperialism divided the globe according to which theory?**
  Context: "... European expansion caused the world to be divided by how developed and developing nation are portrayed through the world systems theory. The two main regions are the core and the periphery. The core consists of high areas of income and profit; the periphery is on the opposing side of the spectrum consisting of areas of low income and profit...
  Predicted Answer: "world systems theory"
  Ground Truth: ["the world systems theory", "world systems theory", "world systems theory.", "world systems theory"]

  Analysis: The model correctly identified the core concept "world systems theory" as the answer.

While both models performed well on the test set, BERT-base achieved higher Exact Match and F1 scores, reflecting stronger overall performance. Although BERT required approximately one hour of training, distilBERT, due to its smaller size, trained about 25% faster but sometimes produced less precise answers. Overall, BERT offered better accuracy, while distilBERT provided an efficient balance between speed and performance.

D. Text-to-Text Transfer Transformer (T5)

T5 is a series of large language models developed by Google AI introduced in 2019. Like the original Transformer model, T5 models are encoder-decoder Transformers, where the encoder processes the input text, and the decoder generates the output text. Unlike models designed for specific tasks, T5's text-to-text framework allows it to be applied to a variety of tasks without needing significant modifications. This unified approach simplifies the model architecture and training process. The original T5 models are pre-trained on the Colossal Clean Crawled Corpus (C4), containing text and code scraped from the internet. This pre-training process enables the models to learn general language understanding and generation abilities. T5 models can then be fine-tuned on specific downstream tasks, adapting their knowledge to perform well in various applications.

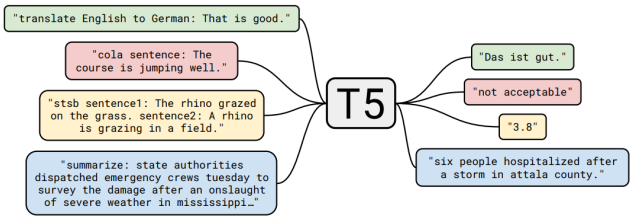


Figure 6: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "Text-to-Text Transfer Transformer".[17]

1) Data Pre-Processing:

- Inputs formatted as "question: [question] context: [context]".
- Answers are randomly selected from available options for training.
- Tokenization with a max sequence length of 512 (input) and 128 (answer).

2) Model Architecture:

- This model used a pre-trained T5-Small(60 million parameters) model and fine tuned with SQuAD dataset. It is used for extractive question answering tasks.
- Libraries: datasets, transformers, evaluate.
- Hyper-parameters: Learning rate 3e-4, batch size 16, epoch size 10, weight decay 0.01, FP16 mixed precision.

3) Training:

- Hardware: 2 NVIDIA T4 GPU
- Training Time: About 6.5 hours.
- Training loss decreased steadily from 0.054600 (epoch 1) to 0.005900 (epoch 10) in Table III.
- Validation loss fluctuated slightly, ending at 0.016470 in Table III.
- EM peaked at 70.76 (epoch 8) in Table III.
- F1 peaked at 82.80 (epoch 8) in Table III.

The T5-small model achieved reasonable performance on SQuAD (63.86 EM / 77.79 F1) after 10 epochs, demonstrating the feasibility of using smaller transformers for QA tasks. Training loss(Figure 7) decreased smoothly, while validation loss(Figure 7) plateaued, suggesting mild overfitting. The gap between validation and test scores (e.g., EM drops from 70.76 to 63.86) also indicates overfitting. It could be caused by limited training epochs, small model size or insufficient regularization. While overfitting issues were noted, the results provide a baseline for future optimizations. Use larger models (T5-base/large) or add dropout/L2 regularization could further improve robustness and accuracy.

Table III: Summary of training and validation set loss and evaluation results for exact match (EM) and F1 scores.

Epoch	Training Loss	Validation Loss	Exact Match	F1
1	0.054600	0.014168	69.748858	81.836120
2	0.016100	0.014550	69.600457	81.941127
3	0.013600	0.014440	69.257991	81.681099
4	0.011600	0.014095	70.433790	82.277726
5	0.010100	0.014538	70.627854	82.556124
6	0.008900	0.015419	70.582192	82.720153
7	0.007800	0.015426	70.764840	82.679631
8	0.007100	0.015901	70.764840	82.797846
9	0.006400	0.016163	70.650685	82.724431
10	0.005900	0.016470	70.285388	82.515690

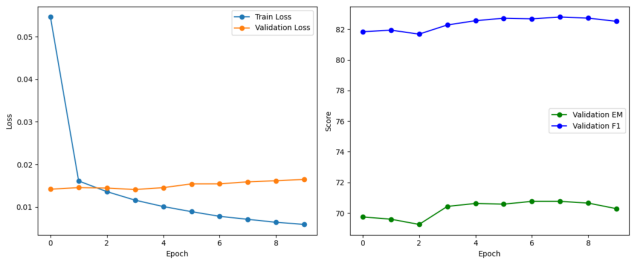


Figure 7: T5-Small Loss and EM/F1 over Training

E. Evaluation Using Generative Language Models

In addition to the primary transformer-based models discussed earlier, we also conducted exploratory experiments using three generative language models: `google/flan-t5-base`, `gpt2`, and the recent open-source model `deepseek-ai/deepseek-llm-7b-chat`. These models are not inherently designed for span-based extractive QA, and thus require a different evaluation strategy for compatibility with the SQuAD v1.1 dataset.

To evaluate these generative models on extractive QA, we adopted a two-stage approach. First, we formatted each input as a prompt combining the question and the corresponding context, and generated a free-form answer using the language model. Then, we compared the generated answer with the ground-truth answer using both rule-based and embedding-based methods.

Initially, we used a *yes/no prompt* approach, where the model itself was asked to decide whether the generated answer matched the ground truth. The prompt format was:

```
Do the following two answers have the
same meaning?
Answer A: [generated answer]
Answer B: [ground truth answer]
Respond with 'Yes' or 'No'.
```

However, we found that relying solely on this yes/no prompting approach often produced inconsistent or inaccurate

decisions, likely due to generative ambiguity and prompt sensitivity. For example, in response to the question:

"What denomination is the Diocesan School for Girls in Auckland affiliated with?"

GPT-2 generated:

Answer A: "It is now the first time a member of the US House of Representatives has been elected to the U.S. Senate, and President Trump has made it a priority to get rid of the controversial filibuster..."

while the ground-truth answer was:

Answer B: "Anglican"

This illustrates that GPT-2, when not fine-tuned, can generate entirely irrelevant content, even when provided with a properly structured prompt.

To improve robustness, we incorporated semantic similarity scoring using a sentence transformer. Specifically, we computed the cosine similarity between the embeddings of the generated and ground-truth answers. If the similarity score exceeded a threshold of 0.6, we treated the prediction as correct. This hybrid method improved the reliability of evaluation for generative models that do not produce exact spans but can still yield semantically valid responses.

For example, in the case of DeepSeek, given a question about language usage:

"Which languages was the Phags-pa script used to write?"

the model generated:

Answer A: "The Phags-pa script was used to write the Mongolian, Tibetan, and Chinese languages."

compared to the ground-truth:

Answer B: "Mongolian, Tibetan, and Chinese"

Although Answer A is not an exact match, it is semantically equivalent. The sentence similarity score exceeded the 0.6 threshold, allowing the prediction to be treated as correct. This example demonstrates how embedding-based evaluation can successfully recognize correct answers that deviate slightly in form but not in meaning.

F. Evaluation Results of Generative Models

To quantify the effectiveness of generative models for extractive QA, we computed final F1 scores using our hybrid evaluation strategy that combines yes/no prompting with sentence embedding similarity. Table IV summarizes the performance of each model.

Table IV: F1 Scores of Generative Models on SQuAD v1.1

Model	Instruction-Tuned	QA-Aware	F1 Score
GPT-2	No	No	13.20
FLAN-T5-Base	Yes	Yes	83.85
DeepSeek LLM-7B	Yes (Chat-tuned)	Yes (Likely)	89.70

As expected, GPT-2 achieved the lowest F1 score (13.20), due to its lack of instruction tuning and domain-specific training for extractive question answering. FLAN-T5-Base, a lightweight instruction-tuned model, significantly outperformed GPT-2 with an F1 score of 83.85. The best performance was observed with DeepSeek LLM-7B, which achieved an F1 score of 89.70. This result highlights the benefit of large-scale instruction-tuned models in zero-shot QA settings, particularly when enhanced with hybrid evaluation strategies that account for semantic similarity.

V. COMPARATIVE ANALYSIS

A. Quantitative

1) Embedding Size Matters: As briefly discussed in the RNNs section, the table II shows that embedding dimensionality has a clear impact on performance across the RNN-based models. Specifically, models using 300-dimensional GloVe embeddings consistently outperformed their 100-dimensional counterparts in both exact match (EM) and F1 scores. For instance, the BiDAF-style LSTM with 300d embeddings achieved an F1 score of 70.26%, compared to 54.26% with 100d embeddings. A similar pattern is observed in EM scores across both models and on both the validation and test sets. These results suggest that the richer semantic information provided by the higher-dimensional embeddings enables the model to capture more nuanced word relationships. Although this improvement comes at the cost of increased model size and training time, the trade-off is generally expected in practice and is often justified by the resulting performance gains. In general, these findings underscore that embedding quality is a critical design factor in extractive QA systems.

2) RNN-based vs Transformer-based Models: T5-Small (Transformer) outperformed the BiDAF-style LSTM (RNN) in accuracy, achieving 63.86 EM and 77.79 F1 compared to the RNN's 58.45 EM and 70.26 F1. However, T5 required significantly more computational resources, with 6.5 hours of training on dual GPUs versus the RNN's 1.67 hours on a single GPU. The transformer model also exhibited notable overfitting (e.g., a 7% EM drop from validation to test), while the RNN maintained stable validation performance.

3) Among Transformer-based Models: BERT-base(110M) outperforms T5-small(60M) in both EM and F1 scores, achieving 78.97 EM and 86.77 F1 compared to T5-small's 63.86 EM and 77.79 F1. However, T5's encoder-decoder architecture offers greater flexibility for multi-task NLP applications. The performance gap may narrow with larger T5 variants (e.g., T5-base or T5-large), but the results highlight BERT's efficiency and accuracy advantages for extractive QA on SQuAD v1.1.

B. Qualitative

1) Comparison of Advantages: The plain LSTM offers simplicity, fast training, and low computational requirements, making it a practical choice for smaller-scale applications. The BiDAF-style LSTM builds on this by introducing attention mechanisms that significantly improve the model's ability to align context and question, leading to better accuracy and more stable convergence across both validation and test sets.

T5's encoder-decoder transformer architecture leveraged pre-training on large corpora for superior generalization and flexibility across NLP tasks. T5's pre-training enables robust performance across diverse QA scenarios. T5's architecture scales better with larger models (e.g., T5-base/large), whereas

RNNs face diminishing returns due to recurrent computation bottlenecks.

2) Comparison of Shortcomings: As the basis for many QA models, the RNN models face important limitations. The plain BiLSTM struggles to capture fine-grained context-question interactions, resulting in lower exact match and F1 scores. While the BiDAF-style LSTM improves performance, it comes with increased computational cost and training time. Importantly, neither RNN architecture benefits from large-scale pretraining, which limits their ability to generalize compared to transformer-based models.

While T5 excelled in large-scale applications, its computational demands and overfitting risks made it less ideal for resource-constrained environments.

VI. FUTURE IMPROVEMENTS

One potential improvement for RNN-based models is to integrate character-level embeddings, as used in the original BiDAF model [2], to capture sub-word information and handle out-of-vocabulary words. Further, incorporating deep contextualized word embeddings such as Embeddings from Language Models (ELMo) [18], which dynamically model syntax, semantics, and polysemy, could strengthen context-question alignment.

The Transformer-based networks have pushed the reasoning-skills to human-level abilities. It can even excel the human capabilities on a few tasks of General Language Understanding Evaluation (GLUE). Transformer-based networks have changed the face of NLP tasks. They can go far beyond the results obtained with RNNs, and they can do it faster. They have helped solve many problems at the same time by providing a direct and efficient way to combine several downstream tasks. Nevertheless, much work remains before having a system with a human-level comprehension of the underlying meaning of texts, that is also sufficiently small to run on devices with low computational power.[19]

VII. CONCLUSION

This project explored and compared a wide range of models for extractive question answering using the SQuAD v1.1 dataset. From simpler recurrent neural networks like LSTMs to advanced transformer-based models such as BERT, T5, and DeepSeek, we examined how different architectures perform when tasked with finding answers directly from text.

Our findings show that transformer-based models consistently outperform traditional RNN models in both accuracy and generalization. Among them, BERT delivered the highest performance on SQuAD, while larger generative models like DeepSeek demonstrated impressive results even in zero-shot settings. However, these improvements often come with increased training time and hardware requirements. On the other hand, RNNs still offer lightweight, faster alternatives that may be better suited for environments with limited computing resources.

Overall, this project highlights the trade-offs between simplicity, speed, and performance in QA model selection. As NLP tools continue to advance, choosing the right model will depend not only on accuracy but also on the specific needs and constraints of the task at hand. Future work can focus on improving generalization, reducing model size, and making high-performance QA systems more accessible to real-world applications.

REFERENCES

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *EMNLP*, 2016.

[2] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," 2018. [Online]. Available: https://arxiv.org/abs/1611.01603

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *NAACL*, 2019.

[4] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[5] A. Radford, J. Wu, R. Child *et al.*, "Language models are unsupervised multitask learners," *OpenAI Blog*, 2019.

[6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, 2020.

[7] K. Nishida, I. Saito, K. Nishida, K. Shinoda, A. Otsuka, H. Asano, and J. Tomita, "Multi-style generative reading comprehension," 2019. [Online]. Available: https://arxiv.org/abs/1901.02262

[8] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2017, pp. 1870–1879. [Online]. Available: https://aclanthology.org/P17-1171

[9] S. Wang and J. Jiang, "Machine comprehension using match-lstm and answer pointer," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1608.07905

[10] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto, "Luke: Deep contextualized entity representations with entity-aware self-attention," 2020. [Online]. Available: https://arxiv.org/abs/2010.01057

[11] "Question answering on squad1.1," https://paperswithcode.com/sota/question-answering-on-squad11, accessed: April 2025.

[12] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[14] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. N. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[17] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research 21*, 2020.

[18] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018. [Online]. Available: https://arxiv.org/abs/1802.05365

[19] A. Gillioz, J. Casas, E. Mugellini, and O. A. Khaled, "Overview of the transformer-based models for nlp tasks," in *Proceedings of the Federated Conference on Computer Science and Information Systems*, vol. 21. ACSIS, 2020, pp. 179–183.