Enhancing Bengali Question Answering: A Llama-Based Approach

***Abstract*—Natural language processing (NLP) has made remarkable strides in recent years, particularly in enhancing question-answering (QA) systems across well-resourced languages like English. However, the development of such systems for low-resource languages like Bengali remains underexplored. This paper presents a transfer-learning-based approach to building a Bengali QA system using the Llama-3.2-3B-Instruct model, leveraging transfer-learning techniques on a synthetic dataset derived from the SQuAD 2.0 benchmark. Our experiments yield an F1 score of 38.63% and an exact match score of 14.22%, comparing the performance of multilingual BERT variants and establishing a benchmark against human responses, highlighting the potential of transfer learning in enhancing QA capabilities for Bengali and similar languages.**

***Index Terms*—Natural Language Processing, Question Answering, Large Language Model, LLama Model, Fine-Tuning, Bengali Dataset.**

I. Introduction

Question Answering (QA) involves the creation of systems designed to automatically respond to human queries in natural language. Text-based QA tasks can be viewed as information retrieval challenges, where the goal is to find relevant documents, extract possible answers, and rank them by relevance. These tasks may focus on reading comprehension, with the aim of pinpointing the exact answer, often referred to as a "span," within a given passage. While QA tasks can encompass a wide range of modalities—such as visual contexts (images), open-domain queries, and multimodal inputs that combine images, videos, audio, and text, alongside commonsense reasoning—this work concentrates on text-based reading comprehension.

Despite substantial progress in developing QA systems for high-resource languages like English, similar advancements for Bengali—a language spoken by over 300 million people—have been limited due to a lack of comprehensive datasets and pre-trained models tailored for Bengali [1]. The absence of large-scale QA datasets for Bengali and the limited availability of skilled annotators have made it challenging to create high-quality reading comprehension datasets [2].

The multilingual BERT model [3] for zero-shot transfer learning and fine-tuning it, using synthetic training dataset for the Bengali reading comprehension task. Additionally,other BERT model variants, such as RoBERTa [4] and DistilBERT [5] for comparison in both zero-shot and fine-tuned settings. To train and test their model, they translated a large subset of SQuAD 2.0 [6] from English to Bengali. During translation, fuzzy matching was applied to maintain the quality of answers. Additionally, They developed a human-annotated Bengali reading comprehension dataset sourced from popular Bangla Wikipedia articles to evaluate their models.

In [7], introduces PAL-BERT, a first-order pruning model built upon the ALBERT[8] model, tailored to the characteristics of question-answering (QA) systems and language models.

For further development, this paper employs the Llama 3.2-3B model to develop a Bengali QA system based on a synthetic dataset derived from the SQuAD 2.0 benchmark [6]. By leveraging transfer learning techniques, we adapt this state-of-the-art transformer model for Bengali reading comprehension tasks. Transfer learning allows us to utilize pretrained models that have been trained on large linguistic corpora and adapt them for specific tasks without requiring extensive labeled data [9].

The contributions of this work include establishing a benchmark for Bengali QA systems using the Llama 3.2-3B model and evaluating its performance against human responses. This research aims to enhance accessibility and usability for diverse user groups, including students and individuals with learning difficulties [10], to save time and be accessible, benefiting groups such as children, and adults seeking precise answers from literature, which can otherwise be time-consuming to thoroughly review. By demonstrating the feasibility of Llama 3.2-3B in reading comprehension for Bengali, we hope to pave the way for further advancements in NLP applications for low-resource languages.

II. Literature Review

A. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) were among the earliest neural architectures used for processing sequential data, especially in natural language processing (NLP) tasks [11]. RNNs use feedback loops to retain a hidden state that captures information from previous inputs, making them well-suited for tasks like language modeling and sequence prediction [12]. However, RNNs face challenges such as vanishing and exploding gradients, which limit their ability to effectively learn long-term dependencies [13]. Despite these drawbacks, RNNs paved the way for later advancements in deep learning models.

The architecture of a basic Recurrent Neural Network (RNN) is illustrated in Figure 1. In this model, each hidden state h_t is derived from the preceding hidden state h_{t-1} and the current input x_t. The mathematical formulation for updating the hidden state h_t is given by:

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$$

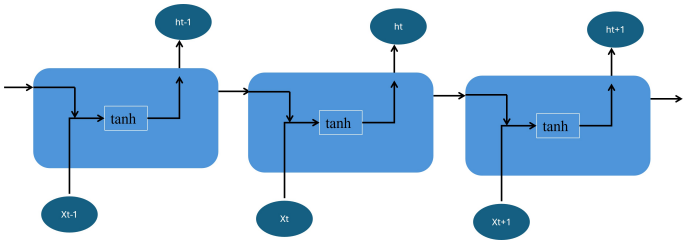

Fig. 1: Recurrent Neural Network Architecture

Here, W_h and W_x represent the weight matrices, while b denotes the bias term. Although RNNs are straightforward in design, they face significant challenges in preserving information over extended sequences due to issues related to gradient propagation. This limitation is effectively addressed by Long Short-Term Memory networks (LSTMs), which are specifically designed to overcome these difficulties.

B. Long Short-Term Memory

Long Short-Term Memory (LSTM) networks were introduced as a solution to the limitations of traditional RNNs [14].LSTMs incorporate memory cells and gating mechanisms that allow them to retain information over extended periods, addressing the vanishing gradient problem effectively. This architecture has been widely adopted in various NLP applications including machine translation and sentiment analysis [15]. LSTMs have demonstrated superior performance compared to standard RNNs in capturing contextual relationships within text data [16], making them a popular choice for sequential modeling tasks. In[17], this tutorial aims to cover the fundamental concepts of RNNs and LSTMs in one comprehensive document.

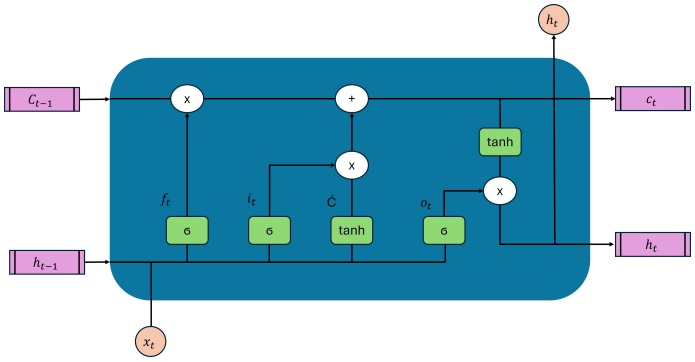

Fig. 2: Long Short Time Network Architecture

The Long Short-Term Memory (LSTM) cell is governed by several crucial components that work together to manage information effectively which is shown in Figure 2:

1) **Forget Gate** (f_t): This gate determines which information from the previous cell state should be discarded. It is mathematically represented as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

2) **Input Gate** (i_t): The input gate decides which new information should be stored in the cell state. Its formulation is given by:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

3) **Cell State Update** ($\tilde{C}_t$): This component generates new candidate values that could be added to the cell state, defined as:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

4) **Output Gate** (o_t): The output gate controls the output at each time step, computed as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

5) **Cell State Update**: The final cell state C_t is updated using the forget and input gates as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

6) **Final Output** (h_t): The final hidden state (output) of the LSTM is calculated using the output gate and the updated cell state:

$$h_t = o_t * \tanh(C_t)$$

These components enable LSTMs to retain information over long sequences while avoiding the gradient-related issues that often plague traditional RNNs.

C. Transformer

The Transformer architecture revolutionized NLP by introducing self-attention mechanisms that enable models to weigh the importance of different words in a sequence without relying on recurrence [18]. In their work, they discuss how this innovation allows Transformers to process entire sequences simultaneously, significantly improving training efficiency and model performance across various tasks. Transformers have become the foundation for state-of-the-art models such as BERT and GPT, which leverage large-scale pretraining on diverse datasets to achieve remarkable results in QA systems. The flexibility and scalability of Transformers make them particularly well-suited for developing robust QA systems capable of handling complex queries across multiple languages. Transformers are composed of two key components: the **encoder** and the **decoder**. The encoder processes the input sequence to create a contextual representation, while the decoder uses this representation to generate the output sequence.

1) **Self-Attention Mechanism**: The self-attention mechanism is central to the Transformer's functionality. It computes attention scores for each word in relation to every other word in the input sequence, allowing the model to weigh their importance dynamically. This capability enables Transformers to capture complex relationships between words regardless of their distance from one another in the sequence.
2) **Multi-Head Attention**: By employing multiple attention heads, Transformers can focus on different aspects of the

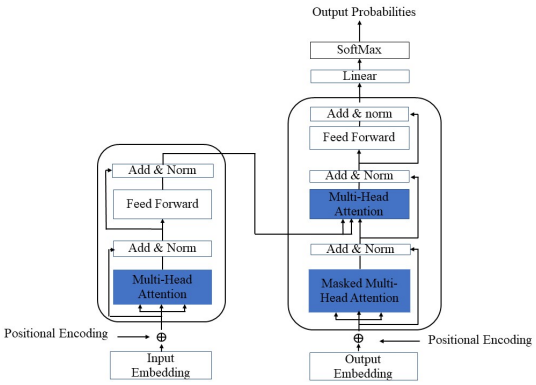

Fig. 3: Transformer Architecture

input simultaneously. Each head learns distinct representations, which are then concatenated and linearly transformed, enriching the model's understanding of context.

3) **Positional Encoding**: Unlike recurrent models, Transformers lack an inherent sense of order, so they use positional encodings to retain information about the position of words in a sequence. This encoding is added to the input embeddings, enabling the model to distinguish between words based on their position in the sequence.
4) **Feed-Forward Networks**: After self-attention layers, each encoder and decoder layer includes feed-forward networks that apply non-linear transformations to enhance representation learning.
5) **Layer Normalization and Residual Connections**: To stabilize training and improve convergence, Transformers utilize layer normalization and residual connections within each layer. These techniques help maintain gradient flow during backpropagation.

In summary, the Transformer architecture represents a significant advancement in NLP, enabling models to learn from vast amounts of data while effectively managing contextual relationships within sequences. Its introduction has paved the way for numerous innovations in language understanding and generation tasks.

As illustrated in Figure 3, the architecture comprises various layers that work together to process input sequences efficiently, highlighting its revolutionary approach to NLP tasks.

D. Large Language Model

Large Language Models (LLMs) have emerged as a transformative force in natural language processing (NLP), enabling machines to understand and generate human-like text. Meta AI has developed several state-of-the-art LLMs, including the Llama series, which focus on efficiency and performance across various NLP tasks. The architecture of these models is primarily based on transformers, which utilize self-attention mechanisms to process input data efficiently and capture long-range dependencies in text.

LLMs have been successfully applied in various domains, including chatbots for customer support and educational tools that provide personalized learning experiences. The versatility of these models allows them to adapt to different languages and contexts, making them valuable in low-resource settings where annotated data is scarce [19].

In[4], a series of foundational language models with sizes ranging from 7 billion to 65 billion parameters. The models are trained on trillions of tokens, demonstrating that state-of-the-art performance can be achieved using only publicly available datasets, without relying on proprietary or restricted data. Notably, Llama-13B surpasses GPT-3 (175B) in most benchmarks, and Llama-65B is on par with leading models like Chinchilla-70B and PaLM-540B.

In[5], introduce Llama 2, a set of large language models (LLMs) that have been pretrained and fine-tuned, with sizes ranging from 7 billion to 70 billion parameters. The fine-tuned versions, named Llama 2-Chat, are specifically optimized for conversational tasks. The models surpass most open-source chat models across various benchmarks, and based on human evaluations for helpfulness and safety, and could serve as viable alternatives to closed-source model

III. Proposed Methodology

This section outlines the steps taken to build and fine-tune the Bengali question answering (QA) system using the Llama-3.2-3B-Instruct model. The approach is divided into three key parts: Data Preprocessing, Model Architecture, and Training Process.

A. Data Preprocessing

The dataset used in this study is a combination of a synthetic Bengali QA dataset derived from the SQuAD 2.0 benchmark and a human-annotated QA dataset sourced from Bengali Wikipedia. The preprocessing pipeline included tokenization using the tokenizer from the model, followed by text normalization to remove special characters, punctuation, and unnecessary spaces. The context, questions, and answers were then formatted into a conversational structure suitable for input into the model. For tokenization, the Hugging Face `tokenizer` was used to prepare the dataset for the transformer model.

The dataset was split into training and validation sets, and the question-answer pairs were carefully aligned to maintain contextual integrity. Additionally, stop words were removed, and lemmatization was applied to the context and question fields to ensure consistency in word forms.

B. Model Architecture

The model architecture follows a transformer-based sequence-to-sequence structure. The Llama-3.2-3B-Instruct model was employed, which incorporates a transformer encoder-decoder mechanism. RoPE scaling was applied for handling long sequences, and the LoRA (Low-Rank Adaptation) method was leveraged to fine-tune specific layers with low-rank updates, reducing computational overhead.

The key components of the architecture include:

- **Tokenization and Embedding Layer**: Responsible for converting input sequences into embeddings that the transformer model can process.

- **Transformer Encoder-Decoder**: Utilizes multiple layers of self-attention and feed-forward networks to encode the input context and generate a response.
- **Prediction Layer**: Outputs the answer to the question based on the context using learned representations.

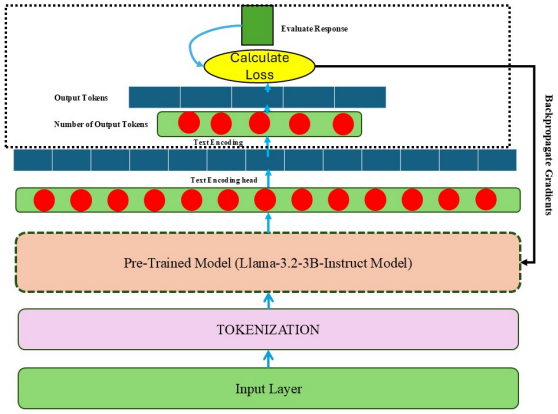

Fig. 4: Model Architecture for Bengali QA System

C. Training Process

The model was trained using the preprocessed dataset, and the transformer model was fine-tuned with the Bengali dataset using the LoRA fine-tuning technique. Gradient checkpointing and 4-bit quantization were used to optimize memory usage. Training was conducted over 60 steps with a learning rate of 2×10^{-4}, a batch size of 2, and a weight decay of 0.01.

The following steps summarize the training process:

1) Load the pre-trained Llama-3.2-3B-Instruct model and apply the LoRA technique for parameter-efficient training.
2) Fine-tune the model using the Bengali QA dataset with the defined hyperparameters.
3) Use the AdamW optimizer and a linear learning rate scheduler with warm-up steps.
4) Train the model for 60 steps, logging the loss values and evaluating on the validation set periodically.

D. Architecture Overview

1) Input Layer (Tokenization): - **Input**: User provides a **context** and **question**. - **Tokenizer**: The tokenizer processes the text into tokens that the model can understand.

2) Pre-trained Language Model (Llama-3.2-3B-Instruct): - The pre-trained model, 'Llama-3.2-3B-Instruct', processes the input sequences. This model has been optimized to handle language tasks and is quantized to **4-bit** precision for reduced memory usage. - **RoPE Scaling** is internally supported to extend the model's sequence length handling capabilities.

3) LoRA Layers (Low-Rank Adaptation): - **LoRA** (Low-Rank Adaptation) is applied to certain key projection layers like 'q_proj', 'k_proj', 'v_proj', etc. - LoRA introduces efficient fine-tuning by injecting trainable low-rank matrices. - Configuration:

- **Rank (r)**: 16
- **LoRA Alpha**: 32 (recommended to be twice the rank for optimal performance).
- **LoRA Dropout**: 0

4) Training Loop (Supervised Fine-Tuning - SFT): - The **SFTTrainer** is initialized with the model and tokenizer. The model is trained using **Supervised Fine-Tuning (SFT)**, with parameters:

- **Training Loss**: 'AdamW 8-bit' optimizer.
- **Batch Size**: 2.
- **Steps**: 60.

- The fine-tuning updates the model on specific tasks, like question-answering based on formatted conversation prompts.

5) Evaluation Layer: - After fine-tuning, the model generates responses for the validation dataset. - **Evaluation Metrics**:

- **Exact Match (EM)**: Measures if the model's prediction matches the ground truth exactly.
- **F1 Score**: Evaluates the overlap between predicted and ground truth tokens.

- The tokenizer decodes the model's outputs and compares them with the validation dataset answers to compute **F1** and **Exact Match** scores.

6) Output (Chat-based Template): - Final output is a structured conversation formatted as:

- **System** (context),
- **User** (question),
- **Assistant** (generated answer).

- The results are returned in a tabular format showing in Table I and Figure 6 and Figure 7 prediction, ground truth, F1 score, and exact match.

This simplified architecture focuses on applying LoRA for efficient fine-tuning on top of a pre-trained Llama model and generating context-based question-answering with minimal resource usage.

IV. Experimental Analysis

To assess the performance of the Bengali QA system, the model was evaluated on a validation set containing 7,488 question-answer pairs. The evaluation used Exact Match (EM) and F1 Score as metrics. The results indicated that the model achieved an F1 score of 38.63% and an exact match score of 14.22%, reflecting a reasonable performance given the challenges associated with a low-resource language.

A. Evaluation Metrics

The performance was evaluated using two metrics:

- **Exact Match (EM)**: This is a binary (true/false) metric that calculates the percentage of predictions that exactly match any of the ground-truth answers. For each question, if the predicted answer is identical to one of the ground-truth answers, the score is 1; otherwise, it is 0. For instance, if the predicted answer is "Einstein" and the ground truth is "Albert Einstein," the Exact Match (EM)

score would be 0. The formula for Exact Match is given by:

$$EM = \frac{\sum_{i=1}^{N} F(x_i)}{N}$$

where

$$F(x_i) = \begin{cases} 1 & \text{if predicted answer = correct answer} \\ 0 & \text{otherwise} \end{cases}$$

- **F1 Score**: The F1 score is a less strict metric than exact match, calculated as the harmonic mean of precision and recall. Precision and recall are computed while treating the prediction and ground-truth answers as bags of tokens. Precision is the fraction of predicted text tokens that are present in the ground truth text, while recall is the fraction of ground truth tokens that are present in the prediction. The formula for F1 Score is:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

For example, in the case of 'Einstein', if the prediction is a subset of the ground truth, precision would be 100

$$F1 = 66.67\%$$

If a question has multiple answers, the answer that yields the maximum F1 score is taken as ground truth. The F1 score is also averaged over the entire dataset to obtain a total score.

The table below presents the detailed evaluation results for the Bengali QA system.

TABLE I: Evaluation Results for the Bengali QA System

Metric	Score
Exact Match (EM)	14.22%
F1 Score	38.63%

Here is the comparison of results on which manuscript we have implemented our models for further improvement, which shows significantly better results. Here, our models' F1 score is better than BERT and DistilBERT, very close to RoBERTA. It is important to mention that Llama 3.2-3B is a lightweight model(low numbers of parameters), and needs 3.4GB of GPU memory.

TABLE II: Comparison between BERT and Llama

Metric	BERT	RoBERTA	Llama 3.2-3B
F1 Score%	36.38	19.16	**38.63**

As illustrated in Figure 5, the training loss indicates how well the model has learned from the training data.

Below are two sample predictions made by the trained Bengali QA system Figure 6 shows the correctly predicted answers where 7 shows the incorrect predicted answers respectively.

These examples illustrate how effectively the model can generate answers based on input questions.

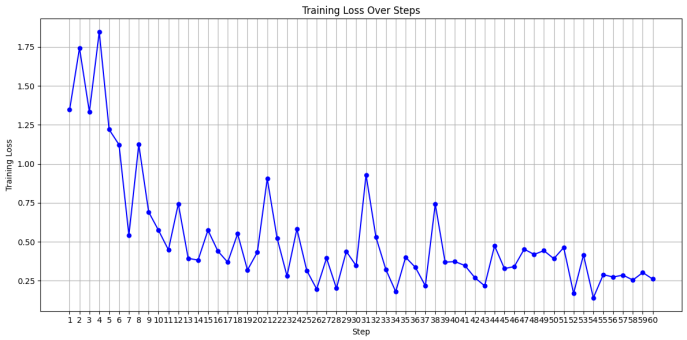

Fig. 5: Training Loss

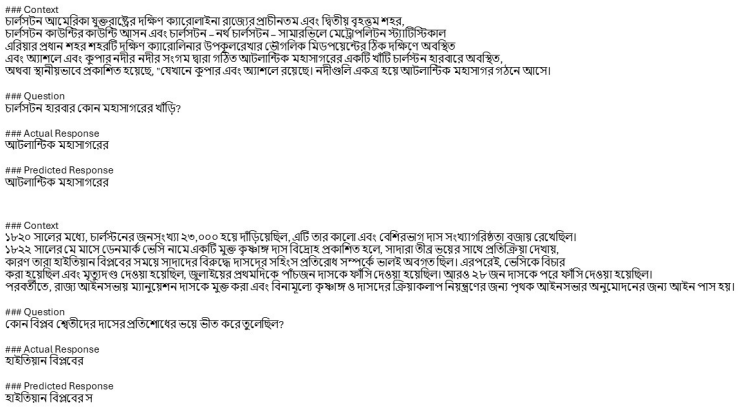

Fig. 6: Samples of correctly predicted answers.

V. Conclusion

In this paper, we presented a Bengali question answering system using the Llama-3.2-3B-Instruct model, incorporating transfer learning and LoRA techniques to handle the low-resource nature of the Bengali language. We introduced a synthetic dataset based on SQuAD 2.0 and a human-annotated Bengali dataset sourced from Wikipedia, addressing the cultural relevance of questions. Through fine-tuning the pre-trained model on this dataset, we achieved an F1 score of 38.63% and an Exact Match (EM) score of 14.22%. These results serve as a foundational benchmark for future work in Bengali QA systems.

The experimental results highlight the potential of leveraging large language models like Llama for low-resource languages, showing that transfer learning can significantly improve QA performance even with limited annotated data. Future work can focus on enhancing dataset quality, exploring different model architectures, and integrating advanced fine-tuning techniques such as RLHF (Reinforcement Learning with Human Feedback). Our approach opens the door for further research into NLP systems for underrepresented languages, providing a framework that can be adapted for other low-resource languages beyond Bengali.

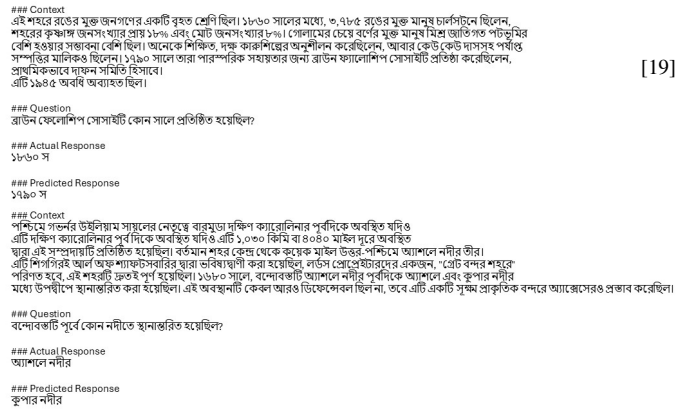

Fig. 7: Samples of incorrectly predicted answers.

REFERENCES

[1] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] S. Ilić, E. Marrese-Taylor, J. A. Balazs, and Y. Matsuo, "Deep contextualized word representations for detecting sarcasm and irony," *arXiv preprint arXiv:1809.09795*, 2018.

[3] T. Tahsin Mayeesha, A. Md Sarwar, and R. M. Rahman, "Deep learning based question answering system in bengali," *Journal of Information and Telecommunication*, vol. 5, no. 2, pp. 145–178, 2021.

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: a robustly optimized bert pretraining approach. corr 2019," *arXiv preprint arXiv:1907.11692*, 1907.

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

[6] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," *arXiv preprint arXiv:1806.03822*, 2018.

[7] W. Zheng, S. Lu, Z. Cai, R. Wang, L. Wang, and L. Yin, "Palbert: an improved question answering model," *Computer Modeling in Engineering & Sciences*, pp. 1–10, 2023.

[8] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," 2020.

[9] A. Conneau and G. Lample, "Cross-lingual language model pretraining," *Advances in neural information processing systems*, vol. 32, 2019.

[10] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," *arXiv preprint arXiv:1902.10909*, 2019.

[11] L. Hirschman, M. Light, E. Breck, and J. Burger, "Deep read: a reading comprehension system. acl in: Proceedings of the 37th annual meeting of the association for computational linguistics (1999)."

[12] M. Richardson, C. J. Burges, and E. Renshaw, "Mctest: A challenge dataset for the open-domain machine comprehension of text," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 193–203, 2013.

[13] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," *Advances in neural information processing systems*, vol. 28, 2015.

[14] S. Hochreiter, "Long short-term memory," *Neural Computation MIT-Press*, 1997.

[15] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, Ieee, 2013.

[16] P. Rajpurkar, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[17] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need. advances in neural information processing systems," *Advances in neural information processing systems*, vol. 30, no. 2017, 2017.

[19] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," (New Orleans, Louisiana), pp. 2227–2237, Association for Computational Linguistics, June 2018.