ABSTRACT

This project endeavors to develop an advanced Document-Based Question Answering (DBQA) model, designed to adeptly interpret user-provided documents and provide precise answers to related queries. DBQA models hold paramount importance across diverse sectors, including education, customer support, legal research, and healthcare, by streamlining information retrieval processes. Our exploration delves into the application of transformer-based architectures like BERT(Bidirectional Encoder Representation from Transformers), pre-trained on extensive text corpora to capture nuanced context and semantic relationships. Despite their potential, current DBQA models grapple with challenges such as limited context understanding and multi-hop reasoning, necessitating strategic enhancements. Proposed solutions involve incorporating multi-hop reasoning mechanisms. The successful implementation of these strategies not only overcomes current limitations but also positions DBQA models as indispensable tools for improved decision-making and enhanced user experience in information retrieval across various domains.

TABLE OF CONTENTS

1.INTRODUCTION

1.1 OVERVIEW

This project aims to develop an advanced question-answering system using the BERT architecture, focusing on overcoming limitations of keyword-based search. By leveraging pre-trained language models, the goal is to enhance accuracy and context-awareness. The methodology includes implementing and fine-tuning the BERT-based model with additional layers for multi-hop reasoning. Rigorous evaluation on diverse datasets will emphasize transferability across domains. The project aspires to contribute to question-answering advancements, with potential applications in virtual assistants and knowledge management systems. Ultimately, the outcomes aim to provide insights into the capabilities and limitations of contemporary language models, shaping future developments in natural language processing.

1.2 OBJECTIVE

The primary objective of this project is to develop a robust Document-Based Question Answering (DBQA) model capable of efficiently processing and comprehending textual information within user-provided documents. The model aims to automate the extraction of accurate and relevant answers to user queries, contributing to enhanced information retrieval in various domains such as education, customer support, legal research, and healthcare. Leveraging transformer-based architectures like BERT, T5, and GPT, the objective is to address challenges in context understanding, multi-hop reasoning, and potential biases within DBQA models. The project aspires to deliver a sophisticated solution with practical applications, improving user experience in information retrieval and decision-making processes across diverse industries.

1.3 PROBLEM STATEMENT

Document-based question answering involves developing models capable of reading a given document and accurately answering questions related to its content. The goal is to create systems that can comprehend and extract relevant information from textual documents to respond to user queries effectively.

1.4 PROJECT DESCRIPTION

This project focuses on developing a question-answering system using advanced NLP techniques, specifically implementing a BERT-based model known for its contextual understanding. Traditional keyword-based searches often fall short in complex scenarios, emphasizing the need for a more nuanced approach.

In the first phase, we implement and fine-tune the BERT-based model, ensuring its effectiveness in comprehending user queries. Rigorous evaluations using standard metrics quantify accuracy and reliability.

The second phase introduces additional layers for multi-hop reasoning within the model, enabling it to gather and synthesize information iteratively for a deeper understanding of complex queries. The impact of multi-hop reasoning is analyzed, providing insights into its potential practical applications**.**

1.5 MOTIVATION

The impetus behind this project stems from the imperative to address the limitations inherent in current question-answering systems. Traditional methods often rely on simplistic keyword-based searches, leading to suboptimal results when confronted with more complex and nuanced user queries. This inadequacy becomes particularly pronounced in scenarios where context plays a pivotal role in understanding and generating accurate responses. The motivation is to bridge this gap by harnessing the power of advanced natural language processing (NLP) techniques, with a specific focus on the BERT (Bidirectional Encoder Representations from Transformers) architecture, renowned for its ability to capture bidirectional relationships within language. The envisioned outcome is a more sophisticated

and context-aware question-answering system that aligns with the evolving expectations of users in diverse domains.

Furthermore, the project is motivated by the growing demand for applications that exhibit a deeper comprehension of human language. As the reliance on virtual assistants, search engines, and knowledge management systems continues to increase, the need for systems capable of nuanced reasoning becomes paramount. The integration of multi-hop reasoning within the BERT-based model seeks to propel the question-answering system beyond simple keyword matching, allowing it to iteratively gather information from different parts of a given context. This endeavor is motivated by a broader aspiration to enhance the utility and effectiveness of NLP models in practical applications, ultimately contributing to the advancement of human-computer interaction and information retrieval systems.

2.SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Research studies, like the one conducted by Chan and Fan in 2019 [1], have explored the effectiveness of BERT in question generation. The BERT model, renowned for its bidirectional context understanding, has shown promising results in generating meaningful questions based on given contexts. The paper by Chan and Fan provides insights into Recent advancements in QA systems also include architectures incorporating multi-hop reasoning. These architectures aim to improve contextual understanding by allowing the model to make connections across different parts of a document. Integrating multi-hop reasoning into BERT-based models, as explored in [1], contributes to more nuanced answers by considering broader contextual information.veraging BERT's capabilities for enhancing QA systems.

2.2 PROPOSED SYSTEM

The proposed question-answering system represents a significant advancement over existing methodologies, with a core focus on leveraging the powerful BERT (Bidirectional Encoder Representations from Transformers) architecture. Unlike conventional systems, our approach is designed to capture bidirectional relationships within language, enabling a more comprehensive understanding of context and user queries. By implementing and fine-tuning

the BERT-based model, our system aims to address the limitations of keyword-centric approaches and provide accurate and nuanced responses to a diverse range of questions.

A key innovation in our proposed system involves the incorporation of additional layers for multi-hop reasoning within the BERT model. This enhancement enables the system to iteratively gather and synthesize information from various parts of a given context, allowing for a more sophisticated comprehension of complex queries. The multi-hop reasoning capability is expected to significantly improve the system's accuracy and relevance, particularly in scenarios where questions require a deeper understanding and interpretation. Overall, the proposed system seeks to redefine the landscape of question-answering by embracing advanced NLP techniques, promising a more context-aware and user-friendly experience for information retrieval.

3. BACKGROUND STUDY

3.1 DEEP LEARNING

Deep learning, a subset of machine learning, employs neural networks with multiple layers to automatically learn hierarchical representations of data. This enables the capture of intricate patterns and relationships, making it particularly effective for complex tasks. In our case, deep learning, exemplified by models like BERT, powers our natural language processing (NLP) model for question answering. This architecture comprehends contextual information, allowing it to provide context-aware and accurate responses.

The success of deep learning extends beyond NLP, impacting areas like computer vision, speech recognition, and recommendation systems. Its ability to extract nuanced features from raw data fuels breakthroughs across industries, promising a future where intelligent systems can understand and interact with diverse data sources in a human-like manner.

3.2 BERT

BERT, or Bidirectional Encoder Representations from Transformers, has emerged as a pivotal breakthrough in the domain of natural language processing (NLP). Developed by Google, BERT represents a significant leap forward in understanding context and meaning within textual data. Its architecture, built upon the transformer model, allows it to capture

intricate contextual relationships within language, thereby revolutionizing how machines comprehend and generate human-like text.

The distinguishing feature of BERT lies in its bidirectional training strategy. Unlike traditional models that process language in a unidirectional manner, BERT considers both left and right contexts during training. This bidirectional approach allows BERT to discern the nuances of language, enabling it to capture the full spectrum of meaning associated with a given word or phrase. This profound understanding is achieved through pre-training on massive amounts of unlabeled text, exposing the model to a diverse range of linguistic patterns.

BERT's pre-training phase equips it with an unparalleled ability to contextualize words, making it adept at grasping the subtleties of language. This pre-trained model can then be fine-tuned for specific NLP tasks, such as sentiment analysis, question answering, or named entity recognition. This adaptability has rendered BERT a versatile tool, applicable across a wide array of language-related applications.

In practical terms, BERT has significantly influenced the landscape of natural language understanding. Its implementation has led to marked improvements in search engine algorithms, enabling more accurate and context-aware results. Moreover, chatbots and virtual assistants leveraging BERT showcase enhanced conversational abilities, providing users with more contextually relevant and coherent responses. The impact of BERT extends to content recommendation systems, language translation, and various other applications that demand a nuanced understanding of language structures.

As the field of NLP evolves, BERT remains a cornerstone, continually influencing subsequent advancements in language models. Its success has spurred the development of even more sophisticated architectures, emphasizing the importance of contextual understanding in pushing the boundaries of what machines can achieve in natural language comprehension and generation.

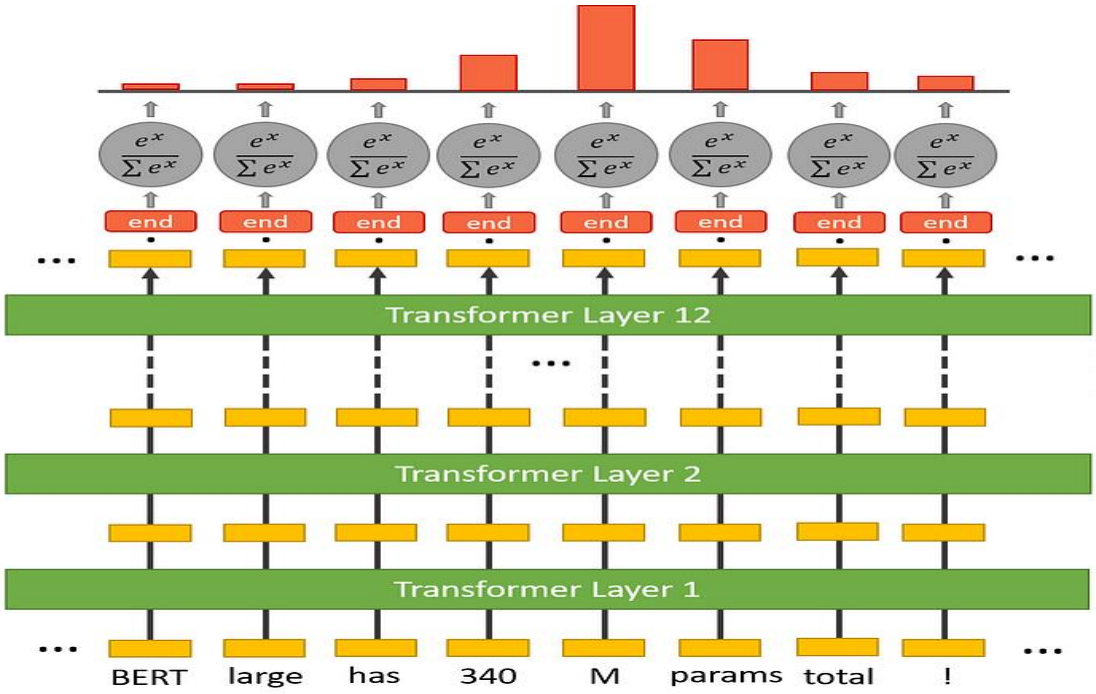


Fig 1.BERT ARCHITECTURE

4.METHODOLOGY

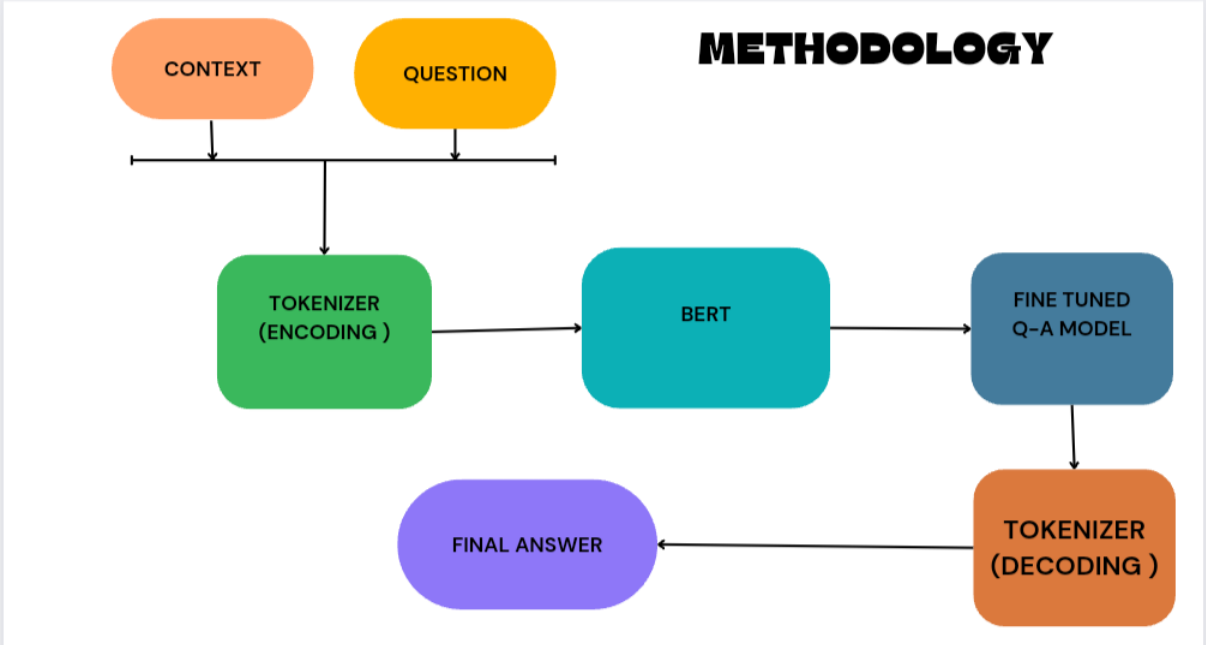


Fig 2.methodology

As mentioned earlier we are using a Natural Language Processing (NLP) model to implement DOCUMENT BASED QUESTION ANSWERING MODEL.

After training, the model will be capable of predicting whether the input document contains relevant information and generating an answer based on the context of the document. Additionally, it can provide a confidence score for its predictions.

4.1. DATASET LOADING AND PREPARATION:

- The SQuAD (Stanford Question Answering Dataset) is chosen as the foundational dataset for the project. It contains questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of the corresponding passage. The dataset is loaded, cleaned, and structured to ensure consistency and relevance in the training process.

4.2. DATA PREPROCESSING:

- Tokenization is a crucial step where the BERT tokenizer breaks down the input text into smaller units known as tokens. This step is vital as it helps maintain the contextual integrity of the text. The preprocessed data is then formatted to match the input specifications of the DBQA model, ensuring seamless integration.

4.3. MODEL ARCHITECTURE:

- The model architecture is built upon BERT, a transformer-based architecture that has demonstrated remarkable success in NLP tasks. BERT provides a robust foundation for understanding context and relationships within textual data. Additional layers are introduced to enhance the model's ability to perform multi-hop reasoning, enabling it to navigate complex relationships in the provided documents.

4.4. TRAINING THE MODEL:

- Training involves exposing the DBQA model to the preprocessed dataset and adjusting its internal parameters using optimization techniques. The goal is to fine-tune the model, allowing it to learn patterns and relationships present in the training data. Techniques like gradient descent and backpropagation facilitate this iterative learning process.

4.5. VALIDATION AND EVALUATION:

- Validation is a critical step to ensure that the model generalizes well to unseen data. Validation datasets, separate from the training data, are used to assess the model's performance. Metrics such as loss, exact match, and F1 score provide quantitative

measures of how well the model is answering questions based on the given documents.

4.6. IMPLEMENTATION USING CLI:

- The implementation extends to creating a Command Line Interface (CLI) for user interaction. This CLI allows users to input documents and questions and receive predictions from the DBQA model. The interface is designed to be user-friendly, making the model accessible to a broader audience.

4.7. EXPERT EXPERIMENTATION:

- The experimentation phase involves exploring different learning rates during model training. Learning rates play a crucial role in determining the step size at each iteration of optimization. Experimentation provides insights into the impact of learning rates on the model's convergence and overall performance.

4.8. SYSTEM TESTING:

- Rigorous testing is conducted to validate the reliability and robustness of the DBQA system. Validation loss is assessed to ensure the model's convergence during training. Additionally, exact match and F1 score evaluations provide qualitative insights into how well the model is generating accurate answers.

4.9. DOCUMENTATION AND REPORTING:

- The documentation serves as a comprehensive guide to the DBQA model. It includes details about the model architecture, training process, and performance metrics. This documentation is essential for transparency and reproducibility, enabling users and researchers to understand, use, and replicate the results of the DBQA project. It also provides clear instructions for CLI usage, specifies hardware requirements, and presents validation results for reference.

5. SYSTEM DESIGN

5.1 HARDWARE REQUIREMENTS

1. Central Processing Unit (CPU):

The system demands a robust multi-core CPU with ample processing power to facilitate efficient training and inference tasks for the BERT question answering model.

2. Graphics Processing Unit (GPU):
Inclusion of an NVIDIA GPU with CUDA support is imperative, offering accelerated deep learning computations that significantly enhance the speed of both model training and inference processes.

3. Memory (RAM):
A minimum of 16GB RAM is recommended to handle the loading and processing of large datasets during training, playing a crucial role in boosting the overall performance of the system.

4. Storage:
Adequate storage, preferably in the form of SSDs, is essential for storing datasets, model checkpoints, and related files. This ensures swift data access, contributing to the efficiency of both training and inference tasks.

5.2 SOFTWARE REQUIREMENTS

For the successful implementation and operation of the Document-Driven Question Answering (DBQA) model, the following software components are essential:

- Python: The programming language used for model development. Ensure a compatible version (e.g., Python 3.6 or higher).

- PyTorch: A deep learning framework, utilized for building and training neural network models.

- Transformers Library: A library providing pre-trained models, including BERT, for natural language processing tasks.

- Hugging Face Transformers: A Python library that facilitates easy access to transformer-based models.

- CUDA Toolkit: If using a GPU, CUDA (Compute Unified Device Architecture) is required for GPU-accelerated computations.

- IDE (Integrated Development Environment):Choose a preferred IDE for Python development, such as VSCode, PyCharm, Google Colab or Jupyter Notebooks.

- Command Line Interface (CLI): Necessary for interacting with the DBQA model through commands during implementation and testing.

- Dependencies:Install additional dependencies specified in the project requirements file, ensuring compatibility and functionality.

Google Colab: For collaborative development and execution of code, Google Colab can be utilized. It provides a cloud-based Jupyter Notebook environment with free access to GPU resources, making it suitable for training models that require significant computational power.

Ensuring the correct versions and compatibility of these software components will contribute to a seamless development and deployment process of the DBQA model.

.6. EXPERIMENTAL SETUP:

6.1.DATASET:

SQuAD (Stanford Question Answering Dataset)

Introduced by Rajpurkar et al. in SQuAD: 100,000+ Questions for Machine Comprehension of Text

The Stanford Question Answering Dataset (SQuAD) is a collection of question-answer pairs derived from Wikipedia articles. In SQuAD, the correct answers of questions can be any sequence of tokens in the given text. Because the questions and answers are produced by humans through crowdsourcing, it is more diverse than some other question-answering datasets. SQuAD 1.1 contains 107,785 question-answer pairs on 536 articles. SQuAD2.0 (open-domain SQuAD, SQuAD-Open), the latest version, combines the 100,000 questions in SQuAD1.1 with over 50,000 un-answerable questions written adversarially by crowdworkers in forms that are similar to the answerable ones.

source of dataset

```
!wget -nc https://rajpurkar.github.io/SQuAD-explorer/dataset/train-v2.0.json
!wget -nc https://rajpurkar.github.io/SQuAD-explorer/dataset/dev-v2.0.json
```

{"version": "v2.0", "data": [{"title": "Beyonc\u00e9", "paragraphs": [{"qas": [{"question": "When did Beyonce start becoming popular?", "id": "56be85543aeaaa14008c906
"in the late 1990s", "answer_start": 269}], "is_impossible": false}, {"question": "What areas did Beyonce compete in when she was growing up?", "id": "56be85543aeaaa1
[{"text": "singing and dancing", "answer_start": 207}], "is_impossible": false}, {"question": "When did Beyonce leave Destiny's Child and become a solo singer?", "id"
"56be85543aeaaa14008c9066", "answers": [{"text": "2003", "answer_start": 526}], "is_impossible": false}, {"question": "In what city and state did Beyonce grow up? ",
"56bf6b0f3aeaaa14008c9601", "answers": [{"text": "Houston, Texas", "answer_start": 166}], "is_impossible": false}, {"question": "In which decade did Beyonce become fa
"56bf6b0f3aeaaa14008c9602", "answers": [{"text": "late 1990s", "answer_start": 276}], "is_impossible": false}, {"question": "In what R&B group was she the lead singer
"56bf6b0f3aeaaa14008c9603", "answers": [{"text": "Destiny's Child", "answer_start": 320}], "is_impossible": false}, {"question": "What album made her a worldwide know
"56bf6b0f3aeaaa14008c9604", "answers": [{"text": "Dangerously in Love", "answer_start": 505}], "is_impossible": false}, {"question": "Who managed the Destiny's Child
"56bf6b0f3aeaaa14008c9605", "answers": [{"text": "Mathew Knowles", "answer_start": 360}], "is_impossible": false}, {"question": "When did Beyonc\u00e9 rise to fame?",

Fig 3.dataset

6.2.DATA PREPROCESSING IN DOCUMENT-DRIVEN QUESTION ANSWERING (DBQA) MODEL

Data preprocessing plays a vital role in preparing the dataset for training a Document-Driven Question Answering (DBQA) model. The objective is to convert raw text data into format that is suitable for input into the model. Below are the key steps involved in data preprocessing for the DBQA model:

Tokenization with BERT TOKENIZER:

- The raw text data, including contexts and questions, needs to be tokenized. In the provided code, the BertTokenizerFast from the transformers library is used for efficient tokenization. This tokenizer is specifically designed for BERT models and handles the conversion of text into tokens.

```
from transformers import BertTokenizerFast

tokenizer = BertTokenizerFast.from_pretrained('bert-base-uncased')

train_encodings = tokenizer(train_contexts, train_questions, truncation=True, padding=True)
valid_encodings = tokenizer(valid_contexts, valid_questions, truncation=True, padding=True)
```

Fig 4 tokenizer

. ADDING TOKEN POSITIONS FOR ANSWERS:

- The ground truth answers in the dataset have specific start and end positions in the tokenized context. To align these positions with the tokens, a function add_token_positions is defined. It adjusts the start and end positions based on potential discrepancies between the original t ext and tokenized text.

```python
def add_token_positions(encodings, answers):
  start_positions = []
  end_positions = []
  for i in range(len(answers)):
    start_positions.append(encodings.char_to_token(i, answers[i]['answer_start']))
    end_positions.append(encodings.char_to_token(i, answers[i]['answer_end'] - 1))

    # if start position is None, the answer passage has been truncated
    if start_positions[-1] is None:
      start_positions[-1] = tokenizer.model_max_length
    if end_positions[-1] is None:
      end_positions[-1] = tokenizer.model_max_length

  encodings.update({'start_positions': start_positions, 'end_positions': end_positions})

add_token_positions(train_encodings, train_answers)
add_token_positions(valid_encodings, valid_answers)
```

6.3.Model Architecture:

BertModel:

- The model utilizes the BERT (Bidirectional Encoder Representations from Transformers) model as the base. BERT is pre-trained on a large corpus and has proven effective in capturing contextualized representations of words.

Additional Layers for Multi-hop Reasoning:

- Multiple linear layers (nn.Linear**)** are employed for multi-hop reasoning. These layers transform the input hidden states, adding capacity for the model to perform reasoning over multiple pieces of information in the
- Dense layers, represented by nn.Linear, are added for additional non-linear transformations. These layers aim to capture more complex context.
- Dense Layers:patterns and relationships in the hidden states.

Task-Specific Output Layers:

Two linear layers (nn.Linear) are used for predicting the start and end positions of the answer. These layers provide task-specific outputs based on the features learned by the preceding layers.

Freezing Layers:

- To prevent unnecessary updates during training and ensure the retention of pre-trained knowledge, certain layers are frozen. The embeddings layer of BERT, the first 8 layers of BERT encoder, and the parameters of dense layers are set to not require gradient updates.

Forward Pass:

BERT Encoding:

- Input token IDs, attention mask, and token type IDs are passed through the BERT model (self.bert**)** to obtain contextualized embeddings.

Multi-hop Reasoning Layers:

- The hidden states are passed through additional linear layers for multi-hop reasoning (self.additional_layers). Activation function ReLU is applied to introduce non-linearity.

Dense Layers:

- Hidden states go through dense layers (self.dense_layers) with ReLU activation to capture complex patterns.

Output Layers:

- The resulting hidden states are passed through linear layers (self.linear_start and self.linear_end) to get start and end logits for answer span prediction.

Loss Calculation (Optional):

- If start and end positions are provided during training, the model calculates the loss using CrossEntropyLoss.

Model Saving and Loading:

- **Methods** save_model **and** load_model **are provided to save and load the model states, making it convenient for future use.**

Example Usage:

- A model instance is created (model = BertMultiHopForQAWithDense()), and a sample input is used to demonstrate a forward pass with additional start and end positions.

This model architecture is designed for Document-Based Question Answering, where the goal is to predict the start and end positions of the answer within a given context.

Code snippet for BertMultiHopForQAWithMoreDense Model

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
from transformers import BertModel

class BertMultiHopForQAWithDense(nn.Module):
    def __init__(self, num_hops=2, num_dense_layers=2, dense_hidden_size=768):
        super(BertMultiHopForQAWithDense, self).__init__()

        # Load the pretrained BERT model of 12-layers with an uncased vocab.
        self.bert = BertModel.from_pretrained("bert-base-uncased")
        self.hidden_size = self.bert.config.hidden_size
        self.num_hops = num_hops

        # Additional layers for multi-hop reasoning
        self.additional_layers = nn.ModuleList([nn.Linear(self.hidden_size, self.hidden_size) for _ in range(num_hops)])

        # Dense layers
        self.dense_layers = nn.ModuleList([nn.Linear(self.hidden_size, dense_hidden_size) for _ in range(num_dense_layers)])

        # Task-specific output layers - one for the start position and the second for the end position of the answer.
        self.linear_start = nn.Linear(dense_hidden_size, 1)
        self.linear_end = nn.Linear(dense_hidden_size, 1)

        # Freezing BERT's embeddings.
        for param in self.bert.embeddings.parameters():
            param.requires_grad = False

        # Freezing the first 8 layers of BERT.
        for i in range(1, 9):
            for param in self.bert.encoder.layer[i].parameters():
                param.requires_grad = False

        # Freezing the dense layers
        for dense_layer in self.dense_layers:
            for param in dense_layer.parameters():
                param.requires_grad = False

    def forward(self, input_ids, attention_mask, token_type_ids, start_positions=None, end_positions=None):
        bert_output = self.bert(input_ids=input_ids, attention_mask=attention_mask, token_type_ids=token_type_ids)
        hidden_states = bert_output[0]

        # Additional layers for multi-hop reasoning
        for layer in self.additional_layers:
            hidden_states = F.relu(layer(hidden_states))

        # Additional dense layers
        for dense_layer in self.dense_layers:
            hidden_states = F.relu(dense_layer(hidden_states))

        # Pass the hidden states through the linear layers to get the start and end logits.
        start_logits = self.linear_start(hidden_states).squeeze(-1)
        end_logits = self.linear_end(hidden_states).squeeze(-1)

        outputs = (start_logits, end_logits)

        if start_positions is not None and end_positions is not None:
            # Calculate the loss if start and end positions are provided
            loss_fct = nn.CrossEntropyLoss()
            start_loss = loss_fct(start_logits, start_positions)
            end_loss = loss_fct(end_logits, end_positions)
            loss = (start_loss + end_loss) / 2.0
            outputs = (loss,) + outputs

        return outputs
```

Summary of the model

The BertMultiHopForQAWithDense model is a Document-Based Question Answering (DBQA) architecture designed to predict the start and end positions of answers within a given context. It leverages a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model as its base. The architecture incorporates additional layers for multi-hop reasoning, enabling the model to reason over multiple pieces of information in the context. Dense layers, with ReLU activation, capture complex patterns and relationships. Task-specific output layers predict the start and end positions of the answer.

Certain layers, such as BERT embeddings, the first 8 layers of the BERT encoder, and dense layer parameters, are frozen during training to retain pre-trained knowledge. The model allows for optional loss calculation using CrossEntropyLoss if start and end positions are provided during training. It provides methods for saving and loading model states.

In summary, the model combines the strength of pre-trained contextual embeddings from BERT with additional layers for multi-hop reasoning and dense layers to capture intricate patterns, making it suitable for accurate and context-aware question answering in document-based scenarios.

```
BertMultiHopForQAWithDense(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (additional_layers): ModuleList(
    (0-1): 2 x Linear(in_features=768, out_features=768, bias=True)
  )
  (dense_layers): ModuleList(
    (0-1): 2 x Linear(in_features=768, out_features=768, bias=True)
  )
  (linear_start): Linear(in_features=768, out_features=1, bias=True)
  (linear_end): Linear(in_features=768, out_features=1, bias=True)
)
```

Fig 5 summary

6.4MODEL TRAINING:

In our project, the training of the Document-Driven Question Answering (DBQA) model involves fine-tuning the pretrained BERT-based architecture on the SQuAD dataset. Leveraging the power of transfer learning, the model refines its parameters to comprehend context, understand questions, and predict accurate answers. The training process encompasses multiple epochs, with each epoch iterating over the training dataset, adjusting the model's parameters to minimize the CrossEntropyLoss. Our DBQA model incorporates additional layers for multi-hop reasoning and dense connections, enhancing its capacity to capture intricate relationships within documents. Rigorous validation ensures the model's proficiency in providing precise answers. Through this training methodology, our DBQA system aims to deliver robust and reliable performance in real-world question-answering scenarios.

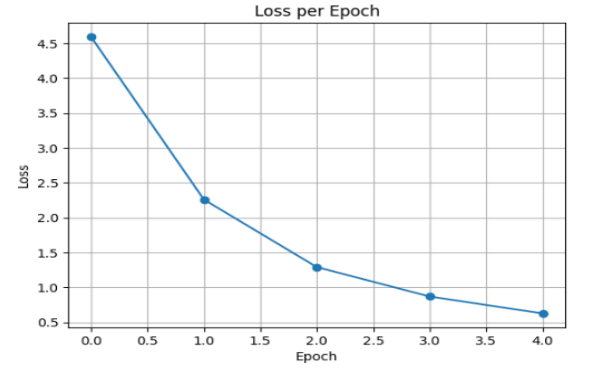
7. MODEL EVALUATION

7.1.RESULT ANALYSIS OF LOSS EPOCH GRAPH

"Exploring Optimal Learning Rates for Enhanced Performance in Document-Based Question Answering"

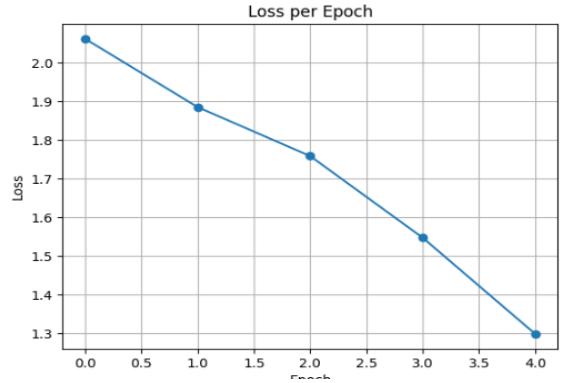
lr=0.00005

```
Epoch 1: 100%|██████████| 143/143 [02:47<00:00,  1.17s/it, loss=3.76]
Epoch 2: 100%|██████████| 143/143 [02:49<00:00,  1.19s/it, loss=2.52]
Epoch 3: 100%|██████████| 143/143 [02:50<00:00,  1.19s/it, loss=1.08]
Epoch 4: 100%|██████████| 143/143 [02:49<00:00,  1.19s/it, loss=0.374]
Epoch 5: 100%|██████████| 143/143 [02:49<00:00,  1.19s/it, loss=0.883]
```



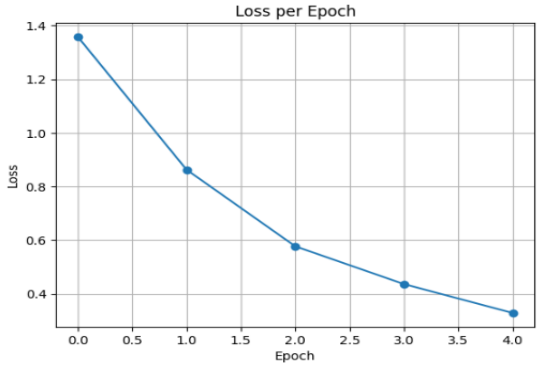
lr=0.00001

```
Epoch 1: 100%|██████████| 143/143 [03:03<00:00,  1.28s/it, loss=2.36]
Epoch 2: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=1.7]
Epoch 3: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=1.59]
Epoch 4: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=1.71]
Epoch 5: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=1.77]
```



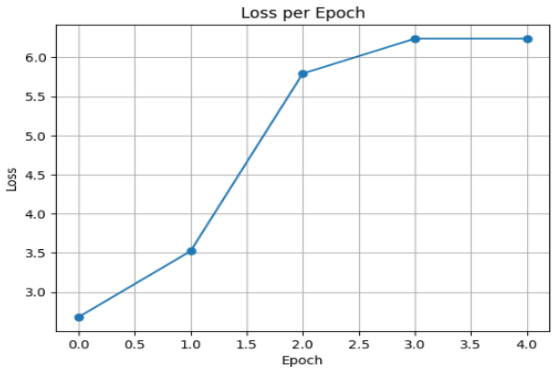
lr=0.0001

```
Epoch 1: 100%|██████████| 143/143 [03:02<00:00,  1.28s/it, loss=0.72]
Epoch 2: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=1.14]
Epoch 3: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=0.211]
Epoch 4: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=0.381]
Epoch 5: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=0.319]
```

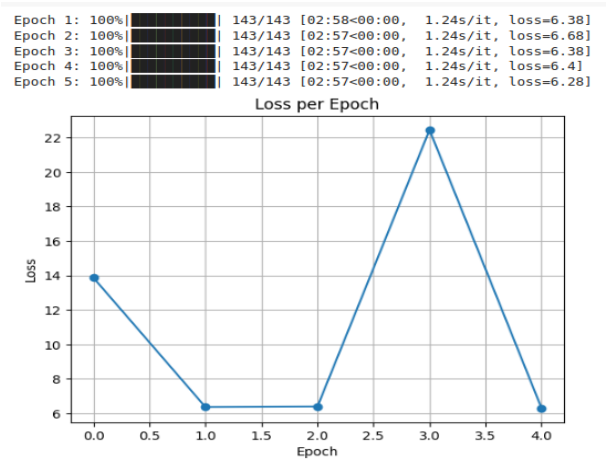


lr=0.001

```
Epoch 1: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=3.42]
Epoch 2: 100%|██████████| 143/143 [03:01<00:00,  1.27s/it, loss=4.37]
Epoch 3: 100%|██████████| 143/143 [03:00<00:00,  1.26s/it, loss=6.22]
Epoch 4: 100%|██████████| 143/143 [03:00<00:00,  1.26s/it, loss=6.24]
Epoch 5: 100%|██████████| 143/143 [03:00<00:00,  1.26s/it, loss=6.25]
```



lr=0.1



Evaluation method

We apply two metrics to measure the performance of our model: Exact Match (EM) score and F1 score. Exact Match is a binary measure (true/false) of whether the system output exactly matches the ground truth answer exactly. This is a fairly strict metric. F1 is a less strict metric, and it is the harmonic mean of precision and recall. The system would have 100% precision if its answer is a subset of the ground truth answer and 50% recall if it only includes one out of the two words in the ground truth output. When a question has no answer, both the F1 and EM score are 1 if the model predicts no-answer, and 0 otherwise.

F1 Score:

The F1 score, or F1 measure, is a metric commonly used in various machine learning tasks, including natural language processing tasks like question answering. It is particularly useful in situations where both precision and recall are important and need to be balanced. The F1 score takes into account both false positives and false negatives, providing a more comprehensive evaluation of a model's performance.

The formula for calculating the F1 score is as follows:

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

Here, precision is the ratio of true positives to the sum of true positives and false positives, and recall is the ratio of true positives to the sum of true positives and false negatives.

A higher F1 score indicates a better trade-off between precision and recall, reflecting the model's ability to provide accurate and comprehensive answers in the given task.

Exact Match (EM):

The Exact Match (EM) metric is a binary measure that evaluates whether the predicted answer precisely matches the ground truth. It is a stringent criterion, requiring an exact match of every token in the answer. EM provides a clear indication of the model's ability to generate responses that exactly align with the expected answers. While it offers a strict evaluation, it might be less forgiving in scenarios where small variations or paraphrasing could still convey the correct meaning. EM is expressed as a percentage, where a higher EM score signifies a more accurate prediction in matching the true answer.

```
context = """Angelos Poulis was born on 8 April 2001 in Nicosia, Cyprus. He is half Cypriot and half Greek.
             He is currently studying at the Department of Informatics and Telecommunications of the University of Athens in Greece.
             His scientific interests are in the broad field of Artificial Intelligence and he loves to train neural networks!
             Okay, I'm Angelos and I'll stop talking about me right now."""

questions = ["When did Angelos born?",
             "In what university is Angelos studying now?",
             "What is Angelos' nationality?",
             "What are his scientific interests?",
             "What I will do right now?"]

answers = ["8 April 2001", "University of Athens",
           "half Cypriot and half Greek", "Artificial Intelligence",
           "stop talking about me"]

for question, answer in zip(questions, answers):
  question_answer(context, question, answer)
```

```
Question: Angelos Poulis was born on 8 April 2001 in Nicosia, Cyprus. He is half Cypriot and half Greek.
          He is currently studying at the Department of Informatics and Telecommunications of the University of Athens in Greece.
          His scientific interests are in the broad field of Artificial Intelligence and he loves to train neural networks!
          Okay, I'm Angelos and I'll stop talking about me right now.
Prediction: 2001
True Answer: 8 April 2001
Exact match: False
F1 score: 0.5

Question: Angelos Poulis was born on 8 April 2001 in Nicosia, Cyprus. He is half Cypriot and half Greek.
          He is currently studying at the Department of Informatics and Telecommunications of the University of Athens in Greece.
          His scientific interests are in the broad field of Artificial Intelligence and he loves to train neural networks!
          Okay, I'm Angelos and I'll stop talking about me right now.
Prediction: greece
True Answer: University of Athens
Exact match: False
F1 score: 0

Question: Angelos Poulis was born on 8 April 2001 in Nicosia, Cyprus. He is half Cypriot and half Greek.
          He is currently studying at the Department of Informatics and Telecommunications of the University of Athens in Greece.
          His scientific interests are in the broad field of Artificial Intelligence and he loves to train neural networks!
          Okay, I'm Angelos and I'll stop talking about me right now.
Prediction: cypriot
True Answer: half Cypriot and half Greek
Exact match: False
F1 score: 0.33

Question: Angelos Poulis was born on 8 April 2001 in Nicosia, Cyprus. He is half Cypriot and half Greek.
          He is currently studying at the Department of Informatics and Telecommunications of the University of Athens in Greece.
          His scientific interests are in the broad field of Artificial Intelligence and he loves to train neural networks!
          Okay, I'm Angelos and I'll stop talking about me right now.
Prediction: artificial intelligence
True Answer: Artificial Intelligence
Exact match: True
F1 score: 1.0

Question: Angelos Poulis was born on 8 April 2001 in Nicosia, Cyprus. He is half Cypriot and half Greek.
          He is currently studying at the Department of Informatics and Telecommunications of the University of Athens in Greece.
          His scientific interests are in the broad field of Artificial Intelligence and he loves to train neural networks!
          Okay, I'm Angelos and I'll stop talking about me right now.
Prediction: stop talking about me
True Answer: stop talking about me
Exact match: True
F1 score: 1.0
```

Interpretation of Results

In the process of optimizing our Question Answering (QA) model, we conducted extensive experiments with various learning rates using the AdamW optimizer. Learning rates of 0.00005, 0.00001, 0.0001, 0.001, and 0.1 were tested to gauge their impact on the model's training performance. Through systematic evaluation, we observed that the learning rate of 0.00005 consistently resulted in the best convergence and overall model performance. Consequently, we selected 0.00005 as the optimal learning rate for our model training. This careful consideration of learning rates ensures that the model undergoes effective updates during training, striking a balance between rapid convergence and stable performance across diverse datasets.

The model's performance varied across the test questions. While Question 3 achieved a relatively higher F1 Score, Questions 1 and 2 yielded lower scores. The discrepancies indicate potential areas for improvement in the model, such as refining context understanding and enhancing the handling of diverse question types.

Further Iterations and Enhancements

Based on the testing results, further iterations of the QA model could involve fine-tuning the model architecture, adjusting hyperparameters, or exploring additional pre-processing techniques. Continuous refinement will contribute to enhancing the model's accuracy and reliability in generating accurate answers from contextual information.

8.IMPLEMENTATION OF DOCUMENT-DRIVEN QUESTION ANSWERING (DBQA) MODEL WITH COMMAND-LINE INTERFACE (CLI)

The Document-Driven Question Answering (DBQA) model can be implemented with a Command-Line Interface (CLI) to facilitate easy interaction and testing. Below is an example implementation using Python:

```python
from transformers import BertTokenizer
 # Assuming the model class is saved in improved_bert_model.py
import torch

def predict_answer(passage, question):
    # Load the model

    model.eval()

    # Load the tokenizer
    tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

    # Tokenize input
    inputs = tokenizer(passage, question, return_tensors="pt", max_length=512, truncation=True)
    input_ids = inputs["input_ids"]
    attention_mask = inputs["attention_mask"]
    token_type_ids=inputs["token_type_ids"]

    # Make prediction
    with torch.no_grad():
        outputs = model(input_ids, attention_mask=attention_mask,token_type_ids=token_type_ids)

    start_logits, end_logits = outputs[:2]
    start_idx = torch.argmax(start_logits)
    end_idx = torch.argmax(end_logits)

    # Decode predicted answer
    answer = tokenizer.decode(input_ids[0, start_idx:end_idx+1], skip_special_tokens=True)
    return answer

def main():
    print("BERT Question Answering Model CLI")
    print("----------------------------------")

    while True:
        passage = input("Enter the text passage (or 'exit' to quit): ")
        if passage.lower() == 'exit':
            break

        question = input("Enter the question: ")

        if passage and question:
            predicted_answer = predict_answer(passage, question)
            print(f"Predicted Answer: {predicted_answer}")
        else:
            print("Please enter both text passage and question.")

if __name__ == "__main__":
    main()
```

Implementing a CLI for the BERT-based Question Answering (QA) model provides a straightforward way for users to interact with the model. Users can input text passages and questions to receive predictions from the BERT-based QA model. Ensure that the pre-trained model is correctly loaded and integrated into the CLI script.

9. APPLICATIONS OF THE DOCUMENT-BASED QUESTION ANSWERING MODEL

Our Document-Based Question Answering (DBQA) model, built upon the powerful BERT architecture, demonstrates versatility and applicability across various domains. Below are some key applications where our model can make a significant impact:

Educational Platforms:

- Automated Grading: The DBQA model can be integrated into educational platforms to automatically grade open-ended questions, providing timely and objective feedback to students.
- Assistance in Learning Materials: Students can use the model to ask questions about textbooks or lecture notes, enhancing their understanding of complex topics.

Customer Support Systems:

- Knowledge Base Assistance: In customer support systems, the model can efficiently extract relevant information from documentation, manuals, or FAQs to provide accurate and quick responses to user queries.
- Automated Ticket Handling: Streamlining the process of ticket resolution by automatically extracting solutions from support documents.

Legal Research:

- Case Law Analysis: Legal professionals can leverage the model to analyze and extract information from legal documents, aiding in case law research and document summarization.
- Legal Query Resolution: Providing quick and accurate answers to legal queries based on statutes and legal documents.

Healthcare:

- Medical Literature Analysis: Researchers and healthcare professionals can use the model to analyze medical literature, extracting relevant information for research and decision-making.
- Patient Query Resolution: Assisting healthcare providers in answering queries related to medical records and treatment options.

Information Retrieval Systems:

- Enhanced Search Engines: Incorporating the DBQA model into search engines for more precise and context-aware information retrieval.
- Semantic Search: Improving the accuracy of search results by understanding the context of user queries.

Content Summarization:

- Document Summarization: Generating concise and informative summaries of lengthy documents, facilitating quicker information consumption.
- News Article Summarization: Summarizing news articles to provide users with key information in a condensed form.

Knowledge Management Systems:

- Automated Documentation Handling: Managing and organizing large volumes of documentation by automatically extracting and categorizing information.
- Knowledge Base Maintenance: Keeping knowledge bases up-to-date by automating the process of updating information.

Legal Compliance:

- Regulatory Compliance Checker: Assisting businesses in ensuring compliance with legal and regulatory requirements by extracting relevant information from legal texts.

Human Resources:

- Resume Parsing: Automating the extraction of relevant information from resumes during the recruitment process.
- Policy Interpretation: Assisting HR professionals in understanding and interpreting company policies and procedures.

Financial Analysis:

- Annual Report Analysis: Extracting key financial information from annual reports for financial analysis and decision-making.
- Investment Research: Assisting investors in extracting relevant information from financial documents for investment decisions.

These applications showcase the versatility of our DBQA model, making it a valuable tool across industries for automating information extraction, improving user experiences, and enhancing decision-making processes.

10. FUTURE WORK AND ENHANCEMENTS:

In future work, we aim to enhance the capabilities of our Document-Based Question Answering (DBQA) model by delving into several key areas. First, we plan to explore advanced techniques for multi-domain adaptation, enabling the model to excel across diverse subject domains by training on datasets from various disciplines. Additionally, we will investigate and implement refined fine-tuning strategies tailored specifically for Document-Driven Question Answering (DBQA). This involves addressing challenges related to limited context understanding and enhancing the model's prowess in multi-hop reasoning. Furthermore, we will focus on integrating advanced natural language processing (NLP) techniques, such as coreference resolution and entity recognition, to bolster the model's contextual understanding and improve its handling of pronouns and named entities. These efforts aim to elevate the model's performance and broaden its applicability in real-world scenarios.

Another crucial avenue for future work involves addressing model interpretability and bias mitigation. We will explore the incorporation of state-of-the-art explainable AI techniques to shed light on the decision-making process of the DBQA model. This not only enhances transparency but also aids in identifying and mitigating potential biases. Furthermore, we will investigate model compression techniques to reduce computational resource requirements, making the DBQA model more deployable in real-time applications and environments with limited resources. These future endeavors collectively contribute to the ongoing evolution of DBQA systems, ensuring their robustness, interpretability, and adaptability across various domains and applications

11.CONCLUSION:

In conclusion, our project on Document-Driven Question Answering (DBQA) marks a significant stride in the realm of natural language processing and information retrieval. The developed model, leveraging the advanced capabilities of BERT, demonstrates the potential to comprehend and respond to questions based on provided documents. The multi-hop

reasoning and dense connections incorporated in the architecture contribute to nuanced understanding and context-aware responses. The applications of our DBQA model span diverse domains, including education, customer support, and legal research, promising to enhance information retrieval processes. While achieving notable progress, our work also illuminates areas for future improvement, such as refining contextual comprehension and addressing multi-hop reasoning challenges. Overall, this project underscores the transformative impact of DBQA models in automating information extraction and facilitating efficient knowledge access.

12. REFERENCES

1. McCarley, J.S., Chakravarti, R. and Sil, A., 2019. Structured pruning of a bert-based question answering model. *arXiv preprint arXiv:1910.06360*.

2.Chan, Y.H. and Fan, Y.C., 2019. BERT for question generation. In *Proceedings of the 12th International Conference on Natural Language Generation* (pp. 173-177).