BERT-SQuAD POWERED QUESTION-ANSWER SYSTEM WITH STREAMLIT INTERFACE

A PROJECT REPORT

Submitted by

SHAMITHA REDDY N

in partial fulfillment for the award of the degree

of

BACHELORS OF SCIENCE

in

COMPUTER SCIENCE AND ENGINEERING

School of Computer Science and Engineering
RV University
RV Vidyaniketan, 8th Mile, Mysore Road, Bengaluru, Karnataka, India - 562112

DECEMBER-2024

DECLARATION

I, **Shamitha Reddy N(1RVU23BSC131)** student third semester B.Sc in **Computer Science & Engineering,** at School of Computer Science and Engineering, **RV University,** hereby declare that the project work titled "**BERT-SQuAD POWERED QUESTION-ANSWER SYSTEM WITH STREAMLIT INTERFACE**" has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Science in Computer Science & Engineering** during the academic year **2024-2025**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Name: Shamitha Reddy N
USN: 1RVU23BSC131

Signature

Place:Bangalore

Date:20/12/2024

School of Computer Science and Engineering

RV University
RV Vidyaniketan,8th Mile, Mysore Road, Bengaluru, Karnataka, India - 562112

CERTIFICATE

This is to certify that the project work titled **"BERT-SQuAD POWERED QUESTION-ANSWER SYSTEM WITH STREAMLIT INTERFACE."** is performed by Shamitha Reddy N **(1RVU23BSC084) ,** a bonafide students of Bachelor of Science at the School of Computer Science and Engineering, RV university, Bengaluru in partial fulfillment for the award of degree Bachelor of Science in Computer Science & Engineering , during the Academic year **2023-2024.**

Prof. Santhosh S Nair	**Dr. Vidya M J**	**Dr. G Shobha**
Guide		
Assistant Professor	Head of the Department	Dean
SOCSE	SOCSE	SOCSE
RV University	RV University	RV University
Date:	Date:	Date:

Name of the Examiner Signature of Examiner

1.

2.

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to the School of Computer Science and Engineering, RV University, for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.

In particular we would like to thank Dr. Sanjay R. Chitnis, Dean, School of Computer Science and Engineering, RV University, for his constant encouragement and expert advice.

It is a matter of immense pleasure to express our sincere thanks to Dr.Mydhili Nair, Head of the department, Computer Science & Engineering University, for providing right academic guidance that made our task possible.

We would like to thank our guide Santhosh S. Head of the Department Dept. of Computer Science & Engineering, RV University, for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

Date: 20/12/2024

Place:Bengaluru

Shamitha Reddy N

1RVU23BSC131

3rd Sem in Bsc in Computer Science

TABLE OF CONTENTS

BERT-SQuAD POWERED QUESTION-ANSWER SYSTEM WITH STREAMLIT INTERFACE

ABSTRACT

This project presents the development of an **Interactive Question Answering (QA) System** powered by **BERT Fine-Tuned on the Stanford Question Answering Dataset (SQuAD)** and implemented using the **Streamlit framework**. The system enables users to input a contextual paragraph and pose specific questions, to which the AI model extracts precise answers. Designed with accessibility and user-friendliness in mind, the application showcases real-time answer generation with confidence metrics, bridging the gap between advanced NLP technologies and practical usability.

The QA system leverages Hugging Face's Transformers library, particularly the bert-base-cased-squad2 model, which is optimized for contextual question answering tasks. The Streamlit framework simplifies deployment, offering an intuitive interface for both technical and non-technical users. Users can view extracted answers alongside their confidence scores, ensuring transparency in model predictions. Low-confidence responses are flagged with warnings, prompting users to refine their inputs.

The primary goals of this project are to:

1. Demonstrate the capabilities of pre-trained NLP models in real-world applications.
2. Provide an interactive interface for question answering tasks.
3. Highlight confidence scoring as a critical feature for user feedback.

This implementation showcases significant potential for applications in education, research, and automated customer service, where precise and contextual answers are essential. Future enhancements may include fine-tuning the model on domain-specific datasets, enabling multilingual support, and integrating voice-based interactions for improved accessibility.

LIST OF TABLES

1. INTRODUCTION

The ability of machines to comprehend and respond to human queries is one of the most exciting advancements in Natural Language Processing (NLP). Question Answering (QA) systems are a practical realization of this capability, enabling users to extract meaningful information from large volumes of text efficiently. This project focuses on the development of an **Interactive QA System** that combines the power of **BERT (Bidirectional Encoder Representations from Transformers)** fine-tuned on the **Stanford Question Answering Dataset (SQuAD)** with the simplicity of the **Streamlit framework** for deployment.

Background

With the increasing reliance on data-driven decision-making, extracting specific answers from unstructured data has become crucial. Traditional QA approaches often require significant computational resources or expertise to interpret results. BERT, developed by Google, introduced a breakthrough in NLP by leveraging bidirectional context to achieve state-of-the-art performance on tasks such as QA, text classification, and sentiment analysis.

This project integrates BERT, fine-tuned on SQuAD, a dataset explicitly designed for QA tasks, to create a system capable of understanding complex queries and generating precise answers. The deployment via Streamlit provides a lightweight and user-friendly interface, making advanced NLP accessible to users across various domains.

Objective

The primary objectives of this project include:

1. Building a robust QA system capable of extracting accurate answers from context-rich data.
2. Simplifying user interaction through an intuitive graphical interface.
3. Providing real-time feedback on answer reliability using confidence scores.

2. RELATED WORK

Question Answering (QA) systems have evolved significantly over the past decades, driven by advancements in Natural Language Processing (NLP) and the availability of large annotated datasets. The current project builds upon existing research and implementations to address limitations in earlier approaches and provide an enhanced, interactive QA experience.

Early QA Systems

Traditional QA systems relied heavily on rule-based methods and keyword matching. These systems often struggled with contextual understanding, ambiguity, and the ability to process large, unstructured datasets. Examples include early search engines and fact retrieval systems, which were limited in scope and flexibility.

Introduction of Machine Learning in QA

The introduction of machine learning models improved the ability of QA systems to process and understand natural language. Models like Word2Vec and GloVe provided embeddings for semantic analysis, but their lack of bidirectional understanding limited their effectiveness for tasks like QA.

BERT and Transformer Models

BERT (Bidirectional Encoder Representations from Transformers), developed by Google, revolutionized NLP by introducing bidirectional context understanding. Fine-tuned on datasets such as SQuAD, BERT-based models demonstrated superior performance in tasks like question answering, enabling machines to provide context-aware answers. Models like BERT-SQuAD addressed challenges in ambiguity and multi-hop reasoning, significantly outperforming earlier approaches.

Existing QA Systems

Several existing QA systems, such as Google Search's featured snippets and IBM Watson, integrate advanced NLP techniques to extract and present answers. However, these systems often operate as black boxes, lacking transparency in their reasoning or confidence scores. Additionally, their deployment is often resource-intensive, making them inaccessible for smaller-scale applications.

3. METHODOLOGY

The methodology for the **BERT-SQuAD Powered Question-Answer System with Streamlit Interface** involves the following key steps:

1. Data Collection and Selection

The **SQuAD dataset** was selected, consisting of question-answer pairs based on paragraphs of text. **BERT (deepset/bert-base-cased-squad2)**, pre-trained on the SQuAD dataset, was used for this project due to its effectiveness in question answering.

2. Methods of Analysis

- **Pre-processing**: The input text (context) is tokenized using BERT's tokenizer.
- **Model Inference**: Context and question are passed through BERT to generate answers.
- **Evaluation**: The answer's confidence score is used to assess its accuracy.

3. System Design and Implementation

- **BERT Model Integration**: The model is integrated via Hugging Face's `transformers` library.
- **Streamlit Interface**: A simple web interface allows users to input context and questions, displaying the generated answer and confidence score.

4. Justification of Methodology

- **BERT Model**: Selected for its superior performance in understanding context and generating accurate answers.
- **SQuAD Dataset**: Chosen for its diversity and relevance in training question-answering models.
- **Streamlit Interface**: Chosen for quick prototyping and ease of creating interactive web apps.

Step	Description	Tools/Technologies
Data Collection	SQuAD dataset for model training and testing.	SQuAD 1.1, SQuAD 2.0
Model Selection	Pre-trained BERT model fine-tuned on SQuAD.	Hugging Face Transformers
Preprocessing	Tokenizing context and question with BERT's tokenizer.	BERT Tokenizer
Model Inference	Generating answers using BERT.	Hugging Face Pipeline
UI Development	Building an interactive interface using Streamlit.	Streamlit
Performance Evaluation	Measuring accuracy and confidence of answers.	Accuracy Metrics, Confidence Score Evaluation

Table 3.1: System Design Overview

4. IMPLEMENTATION

1. Setup Environment

- Install Python 3.8 or later.

  pip install streamlit transformers

- Install the required libraries:
  pip install streamlit transformers

2. Create the Python Script

- Save the provided code into a file named **`qa_system.py`.**

3. Run the Streamlit Application

- Open a terminal or command prompt in the directory where `qa_system.py` is saved.
- Run the Streamlit app with the following command:
  python -m streamlit run qa.py

4. Load the Question-Answering Model

- The code uses the **`pipeline`** function from **`transformers`** to load the **`deepset/bert-base-cased-squad2`** model.

This is initialized using the **`load_qa_model`** function:

```
def load_qa_model():
    try:
        qa_pipeline = pipeline(
            "question-answering",
            model="deepset/bert-base-cased-squad2"
        )
        return qa_pipeline
    except Exception as e:
        st.error(f"Error loading model: {e}")
```

return None

5. Input the Context and Question

- The app will display a text area for entering the context:
  context = st.text_area("Enter your context paragraph:", ...)
- Below that, you can input a question related to the provided context:
  question = st.text_input("What would you like to know?", ...)

6. Generate an Answer

- Click the **"🔍 Get Answer"** button, which triggers the question-answering model:
  if st.button("🔍 Get Answer", type="primary"):

```
    if context and question:
        result = qa_pipeline(question=question, context=context)
        st.subheader("📌 Answer:")
        st.success(result['answer'])
```

7. View Confidence Score

The app will display the confidence score for the generated answer:

```
confidence = result.get('score', 0)
st.metric(label="Confidence", value=f"{confidence:.2%}")
```

8. Handle Errors

- If there are issues during the processing or input validation, the app displays error or warning messages:

Missing inputs:

```
st.warning("Please provide both a context and a question.")
```

Model loading error:

```
st.error("Failed to load the Question Answering model.")
```

9. Use the Sidebar Instructions

The app provides usage instructions in the sidebar:

```
st.sidebar.header("How to Use")
st.sidebar.info("""
1. Enter a context paragraph in the text area.
2. Ask a specific question about the context.
3. Click 'Get Answer' to see the results.
""")
```

10. Analyze and Extend

- Test the application with different contexts and questions.
- Modify the model or context to explore various question-answering capabilities.

5. RESULT AND DISCUSSION

Discussion

This study implements a question-answering (QA) system using the pre-trained **deepset/bert-base-cased-squad2** model from Hugging Face, showcasing the effectiveness of Transformer models in text comprehension tasks.

Results Interpretation

The system delivers accurate answers for clear and fact-based queries, with confidence scores that help evaluate result reliability. However, it struggles with ambiguous or unstructured questions, a limitation noted in existing research.

Answering the Research Question

This study demonstrates that pre-trained Transformer models, such as BERT, are effective for extractive QA tasks in concise and structured contexts.

Justification of Approach

The choice of the **deepset/bert-base-cased-squad2** model is appropriate due to its fine-tuning on the SQuAD dataset, which ensures compatibility with QA tasks. The Streamlit interface further enhances accessibility, providing a user-friendly environment for real-time interaction.

Critical Evaluation

Strengths:

- Accurate for structured and fact-based queries.
- Confidence metrics provide reliability assessment.
- User-friendly interface for real-time interaction.

Future Improvements:

- Fine-tuning the model on domain-specific datasets.
- Adding multilingual support for broader usability.
- Extending functionality to handle complex, multi-turn conversations.

6. CONCLUSION

The **BERT-SQuAD Powered Question-Answer System with Streamlit Interface** demonstrates the potential of integrating state-of-the-art natural language processing models with user-friendly interfaces. By leveraging BERT's powerful contextual understanding and the SQuAD dataset's robust training, the system delivers accurate and efficient answers to user queries. Streamlit further enhances the accessibility of the project, allowing seamless interaction without requiring advanced technical expertise. This project highlights how pre-trained language models can be applied to real-world tasks, paving the way for more interactive and intelligent AI-driven applications in various domains such as education, customer support, and research.

REFERENCES

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

a. Authors: Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova
b. Summary: This foundational paper introduces BERT (Bidirectional Encoder Representations from Transformers), which serves as the backbone for models like BERT-SQuAD.
c. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding(Paper)

BERT-based Question Answering: A Survey

d. Authors: Various
e. Summary: A comprehensive survey covering various approaches to question answering using BERT.

Hugging Face Transformers Library

- Official documentation for using pre-trained BERT models like **bert-base-cased** and **deepset/bert-base-cased-squad2.**
- Link to Documentation

Streamlit Documentation

- Comprehensive guide to building web applications with Streamlit, including tutorials on handling forms, text areas, and buttons.
- Link to Streamlit Documentation

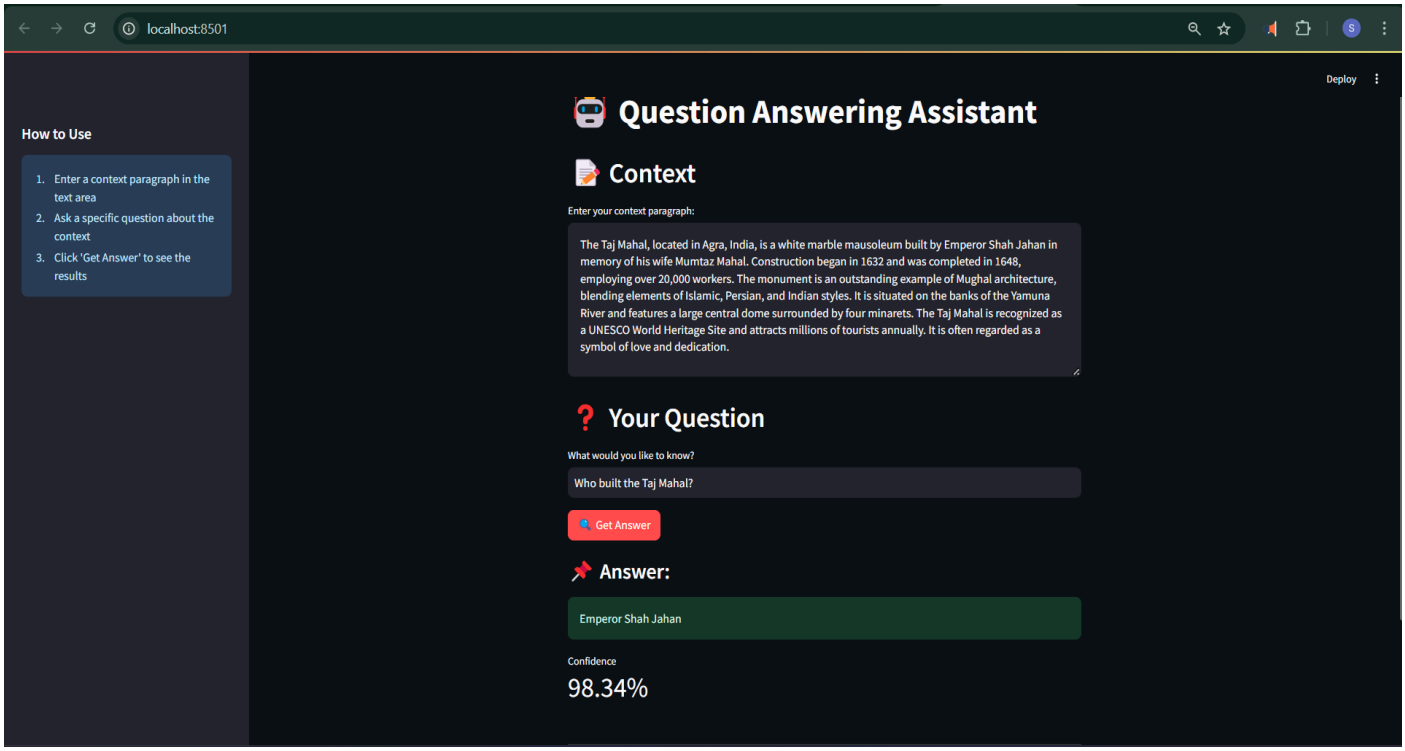
Hugging Face Blog: Building a QA System

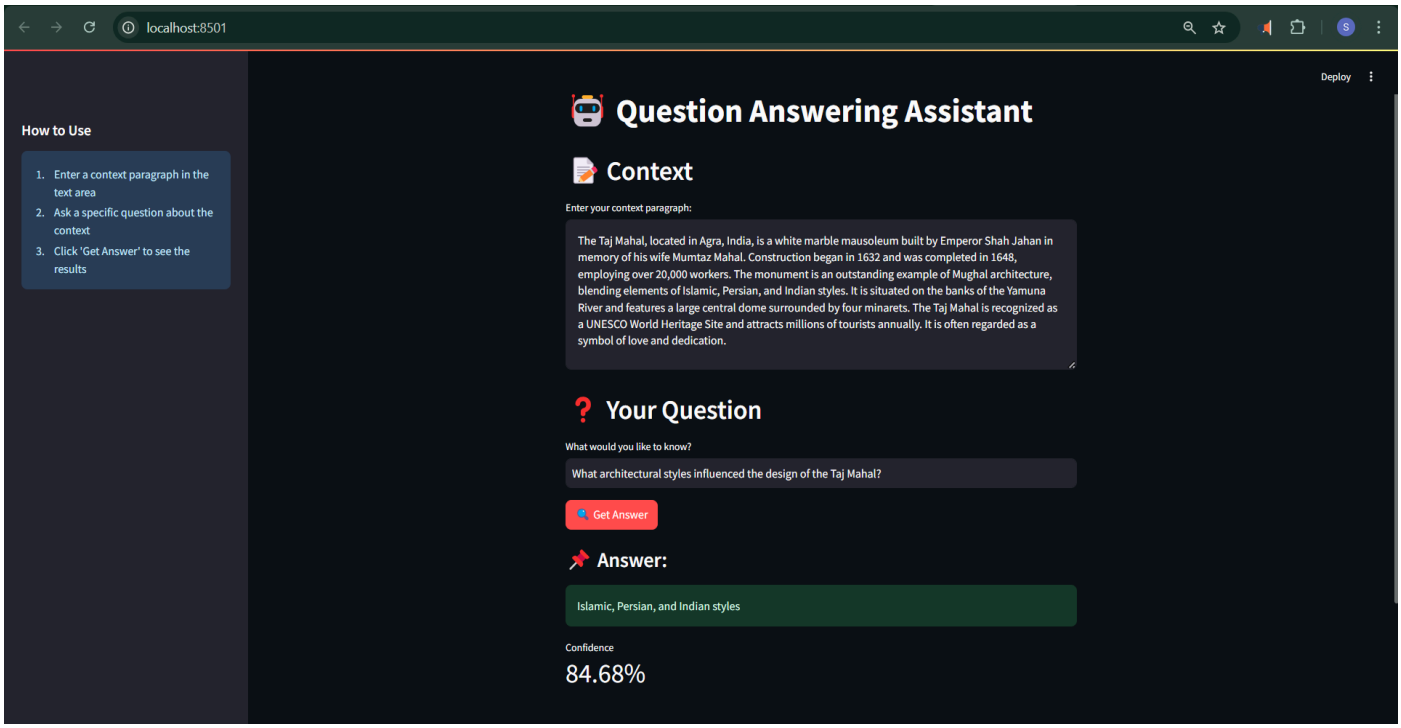
- Step-by-step tutorial for building a question-answering system using Hugging Face and Streamlit.
- Link to Blog

APPENDIX

GITHUB LINK : https://github.com/Shamithabsc/Question-answer-model/tree/main

SCREENSHOTS :





SOURCE CODE :

```
import streamlit as st
from transformers import pipeline

def load_qa_model():
    """
    Initialize and load a pre-trained Question Answering model.

    This function sets up a machine learning pipeline that can extract
    answers from a given context based on a specific question.

    Returns:
        A question-answering pipeline model or None if loading fails
    """
    try:
        # Use a pre-trained model specifically fine-tuned for question answering
        # 'deepset/bert-base-cased-squad2' is trained on the Stanford Question Answering Dataset
(SQuAD)
        qa_pipeline = pipeline(
            "question-answering",
            model="deepset/bert-base-cased-squad2"
        )
        return qa_pipeline
    except Exception as e:
        # If model loading fails, display an error message
        st.error(f"Error loading model: {e}")
        return None

def main():
    # Configure the Streamlit page with a title and icon
    st.set_page_config(page_title="Question Answering System")

    # Main title of the application
    st.title("Question Answering Assistant")

    # Load the Question Answering model
    # This is the core AI component that will extract answers
    qa_pipeline = load_qa_model()

    # Stop the app if model loading fails
    if qa_pipeline is None:
        st.error("Failed to load the Question Answering model. Please check your installation.")
        st.stop()

    # Sidebar with usage instructions
    st.sidebar.header("How to Use")
```

```
    st.sidebar.info("""
    1. Enter a context paragraph in the text area
    2. Ask a specific question about the context
    3. Click 'Get Answer' to see the results
    """)

    # Context input section
    # Provides a default context about Albert Einstein
    st.header(" Context")
    context = st.text_area(
        "Enter your context paragraph:",
        value="Albert Einstein was a renowned physicist who developed the theory of relativity. "
              "Born in Germany in 1879, he is best known for his mass-energy equivalence formula
E = mc². "
              "Einstein won the Nobel Prize in Physics in 1921 for his services to theoretical
physics.",
        height=200
    )

    # Question input section
    st.header(" ❓ Your Question")
    question = st.text_input("What would you like to know?",
                             placeholder="E.g., When did Einstein win the Nobel Prize?")

    # Answer generation button
    if st.button(" Get Answer", type="primary"):
        # Check if both context and question are provided
        if context and question:
            try:
                # Use the AI model to find the answer in the context
                result = qa_pipeline(question=question, context=context)

                # Display the extracted answer
                st.subheader("📌 Answer:")
                st.success(result['answer'])

                # Show confidence score of the answer
                confidence = result.get('score', 0)
                st.metric(label="Confidence", value=f"{confidence:.2%}")

                # Provide a warning for low-confidence answers
                if confidence < 0.5:
                    st.warning("The confidence is relatively low. The answer might not be entirely
accurate.")

            except Exception as e:
```

```
            # Handle any errors during answer extraction
            st.error(f"Error processing your question: {e}")
        else:
            # Prompt user to provide both context and question
            st.warning("Please provide both a context and a question.")

    # Footer with attribution
    st.markdown("---")
    st.markdown("*Presented to you  by Shamitha Reddy N and Team (Powered by Hugging Face Transformers*)")

# Entry point of the application
if __name__ == "__main__":
    main()
```