**Jainil Patel: Implementation of BERT for Context-Based Question Answering**

The purpose of this project is to develop a question-answering (QA) system using BERT, a state-of-the-art natural language processing model, to enable AI scouts to efficiently extract relevant information from large text passages. This work supports the broader goal of the CSC 5542 class project, which aims to build an AI-powered scouting assistant capable of analyzing diverse data sources (e.g., player statistics, game footage, and online articles) to generate actionable insights for sports teams. The QA system implemented here allows users to input a context paragraph (e.g., a scouting report or article) and ask specific questions to retrieve precise answers, reducing the time spent manually searching for information.

### Model and Implementation

The system leverages **BERT-Large-Uncased (Whole Word Masking)**, a pre-trained model fine-tuned on the Stanford Question Answering Dataset (SQuAD). Unlike generative models, this extractive QA model identifies the exact span of text within the provided context that answers the question, ensuring factual consistency.

A Streamlit web application was developed to demonstrate the system’s capabilities:

* Users input a **context passage** (e.g., a paragraph about player tactics or injury reports).
* Users submit a **question** (e.g., “What are the key indicators of fatigue in athletes?”).
* The model processes the inputs and returns an **answer** along with a **confidence score** (0–100%), indicating its certainty.

**Example Workflow**:

* **Context**: “GPS trackers on players monitor heart rate variability, sprint frequency, and deceleration patterns to assess fatigue levels during matches.”
* **Question**: “How is player fatigue measured?”
* **Answer**: “GPS trackers monitor heart rate variability, sprint frequency, and deceleration patterns.”
* **Confidence**: 89.5%

### Results and Evaluation

The system performed robustly in tests, accurately extracting answers when the context contained relevant information. For example, when asked “What does big data analytics involve?” with a context about data science, it correctly identified “examining large data sets to uncover hidden patterns, correlations, and customer preferences.” with a 92% confidence score.

**Strengths**:

1. **Speed**: Answers are generated in real-time, even for lengthy contexts.
2. **Transparency**: Confidence scores help users gauge answer reliability.
3. **Accuracy**: Faithfully extracts verbatim answers, avoiding hallucinations.

**Limitations**:

1. **Context Dependency**: The model cannot answer questions outside the provided context.
2. **Paragraph Quality**: Performance degrades with poorly structured or ambiguous text.
3. **Single-Span Answers**: Currently limited to extracting one contiguous text span.

### Relevance to AI Scouting

This tool enables AI scouts to:

* Rapidly query large documents (e.g., player profiles, medical reports, or tactical guides).
* Identify key details (e.g., injury recovery timelines, playstyle tendencies) without manual review.
* Integrate with web scraping tools to analyze real-time data from online sources.

### Future Improvements

To better serve scouting needs, the model could be:

1. **Fine-tuned on sports-specific texts** (e.g., scouting reports, match summaries).
2. **Extended to multi-span answers** for complex queries.
3. **Integrated with a retrieval system** to automatically fetch relevant context from databases or the web.

**Video Demo**:

<https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=2f4f7dd3-4929-4a68-b0a9-b28c00462d8e>

<https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6a436d22-7bd1-4a2a-8140-b28c004154ca>

**Github repo**:  
<https://github.com/jainilpatel98/Colab-PyTorch-NeuralNetwork/tree/main/Day%205%3A%20Transformers>

**README Instructions**:

1. Clone the github repo.
2. Open a terminal and navigate to the file location: cd Github\_repo\_location.
3. Run streamlit run bert.py to launch the app.
4. Also we have the app for the AI Scout.