National University Of Computing & Emerging Sciences

Final Year Project (CS-491)

Project Report
Fall '23, BS(CS)
CogniAssess

Team Candidates

Sumsam Ali - 20K-1075

Mukand Krishna - 20k-0409

Bahadur Khan - 20k-1081

Supervisor

Dr. Nouman Durrani

Project Code
F23-109D

From the Department Of Computer Science

Faculty of Computer Sciences and Engineering

National University Of Computing And Emerging Sciences

Distribution List

The following table will contain a list of people to whom the document will be distributed after every sign-off

Name	**Role**
Name of Supervisor	Dr. Nouman Durrani
Name of Co-Supervisor	None

Document Information

Category	Information
Customer	FAST-NU
Project	CogniAssess
Document	Project Report
Document Version	1.4
Status	Draft
Author(s)	Sumsam Ali , Mukand Krishna and Bahadur Khan
Approver(s)	
Issue Date	
Document Location	
Distribution	
Advisor	
Project Coordinator's Office (through Advisor)	

Contents

1 Introduction

In the ever-evolving landscape of talent acquisition, the challenge of accurately assessing non-technical skills during recruitment is significant. Traditional methods often fall short in objectively evaluating candidates' soft skills, leading to suboptimal hiring decisions. Recognizing this gap, our Final Year Project, COGNIASSESS, aims to transform the recruitment process for non-technical roles through technological innovation and user-centric design.

COGNIASSESS is conceptualized as a cutting-edge platform, leveraging the power of Artificial Intelligence (AI) and specifically Large Language Models (LLMs) to revolutionize the evaluation of non-technical skills in recruitment. The platform is designed to streamline the candidate assessment process, offering a seamless and intuitive user experience. Candidates can effortlessly register, upload their CVs, select desired roles, and partake in personalized assessments tailored to gauge their non-technical competencies. On the other side, recruiters benefit from a robust set of tools enabling efficient candidate search, filtering, and selection, driven by comprehensive assessment data. The overarching goal is to build a secure, efficient, and user-friendly solution, enhancing the recruitment experience for both candidates and recruiters.

The project encompasses the development and fine-tuning of an AI model, utilizing a range of sophisticated Python libraries and modules such as PyTorch, transformers, and peft for efficient training and model optimization. The training phase involves preparing and processing diverse datasets, including Non-Technical Roles, Introduction Training, and Role-playing datasets. This phase is crucial for developing a model that accurately understands and responds to various non-technical assessment scenarios.

Subsequent to the training phase is the testing and evaluation of the model. This step ensures the model's accuracy and reliability in real-world applications. We employ state-of-the-art techniques and tools to evaluate the model's performance, preparing it for deployment in a live environment.

The final phase of the project is the deployment of the trained model. This involves integrating the model with the COGNIASSESS platform, ensuring seamless interaction with the user interface and backend systems. The deployment is executed with a focus on scalability, security, and performance, aligning with the project's objective of providing a robust and efficient non-technical skill assessment tool.

As a culmination of our Final Year Project, COGNIASSESS not only embodies a technological breakthrough in AI-driven recruitment but also marks our commitment to enhancing the recruitment experience with AI. Our aim is to offer a platform that optimizes recruitment processes while being adaptable, secure, and in line with the evolving dynamics of the modern workforce. COGNIASSESS, therefore, stands as a testament to the potential of AI in transforming traditional recruitment methodologies, paving the way for a more effective and equitable hiring landscape.

2 Methodology

The methodology for the COGNIASSESS project, a comprehensive platform for non-technical skill assessment in recruitment, involves a series of meticulously planned steps. This section outlines the methodologies employed from data collection to the deployment of the model on the Hugging Face platform.

2.1 Data Collection

The data collection stage is pivotal in training the AI-driven large language models (LLMs) for the COGNIASSESS platform. This phase involves gathering, filtering, and preparing datasets that reflect the diversity and complexity of non-technical roles and assessments.

2.1.1 Sources of Data

For the COGNIASSESS project, the data sources were carefully selected to encompass a wide range of non-technical roles and scenarios. The datasets used include:

- **QnA for Non-Technical Roles:** A dataset providing a variety of questions and answers related to non-technical job roles.
- **Introduction Training:** This dataset includes introductory level training materials and queries, essential for baseline assessments.
- **Roleplay Training:** A collection of roleplay scenarios and responses, offering insights into practical and situational aspects of non-technical roles.

2.1.2 Data Selection Criteria

The selection of these datasets was guided by specific criteria to ensure their relevance and efficacy in training the model:

- **Alignment with Non-Technical Skills:** The data must be representative of various non-technical roles and the skills required for them.
- **Diversity and Inclusivity:** Ensuring the data covers a broad spectrum of scenarios, roles, and domains to foster an inclusive assessment platform.
- **Quality of Content:** The clarity, correctness, and relevance of the questions and answers were paramount in the selection process.
- **Adaptability for AI Training:** The datasets were chosen for their compatibility with AI-driven analysis and modeling techniques.

2.1.3 Data Acquisition Process

The acquisition process involved several steps, each critical to securing high-quality and relevant data for the project:

1. **Dataset Identification:** Initial research and exploration were conducted to identify datasets that fit the project's criteria.
2. **Accessing Data Repositories:** The datasets were accessed through their respective online repositories, primarily hosted on platforms like Hugging Face.

3. **Data Extraction:** Relevant data from these datasets were carefully extracted. This process involved selecting specific portions of the datasets that were most pertinent to the project's needs.

4. **Data Validation:** Post-extraction, the data underwent a validation process to ensure its integrity and applicability to the project's objectives.

For instance, the datasets were loaded using Hugging Face's 'datasets' library, as demonstrated in the following Python code snippet:

```python
from datasets import load_dataset

Non_Technical_Roles_dataset = load_dataset('Sumsam/QnA_for_Non-Technical_Roles')
Intro_Training_dataset = load_dataset('Sumsam/Introduction_training')
Role_playing_dataset = load_dataset('Sumsam/Roleplay_training')
```

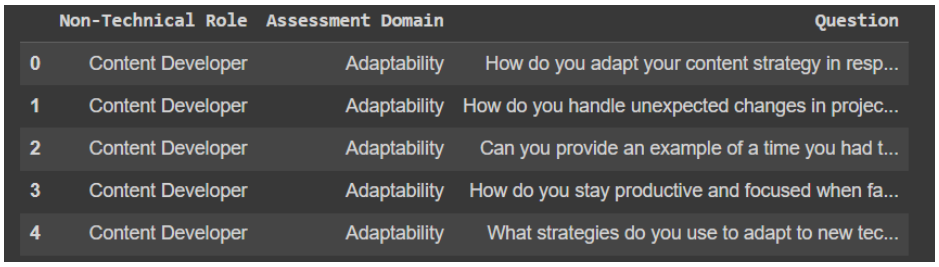
This code effectively loads the datasets into the working environment, setting the stage for subsequent preprocessing and analysis.

2.2 EDA - Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an essential aspect of any data science or machine learning project. It involves summarizing the main characteristics of data, often visually, to gain insights and a better understanding of the dataset. This section delves into the EDA for the provided datasets, broken down into specific subsections.

2.2.1 QnA for Non-Technical Roles Dataset

Dataset Overview This section involves loading the dataset into a pandas DataFrame and performing initial explorations. Commands like `head()`, `info()`, and `describe()` are used to understand the dataset's structure, data types, and basic statistical details.



	Non-Technical Role	Assessment Domain	Question
0	Content Developer	Adaptability	How do you adapt your content strategy in resp...
1	Content Developer	Adaptability	How do you handle unexpected changes in projec...
2	Content Developer	Adaptability	Can you provide an example of a time you had t...
3	Content Developer	Adaptability	How do you stay productive and focused when fa...
4	Content Developer	Adaptability	What strategies do you use to adapt to new tec...

Figure 1: Enter Caption

Null Value Analysis Identifying missing values is crucial for data integrity. This subsection details the process of using `isnull().sum()` to find null values and discusses the importance of handling missing data in the analysis.

Categorical Variable Exploration The analysis of the 'Non-Technical Role' and 'Assessment Domain' categories includes:

- Counting the occurrences of each category.
- Visualizing the category distributions with bar charts and pie charts to understand their frequency and significance.

Word Cloud Analysis Word clouds are generated for the 'Question' and 'Assessment Domain' columns using the `WordCloud` library. This subsection discusses the insights gained from these visual representations, like prevalent themes and common words.

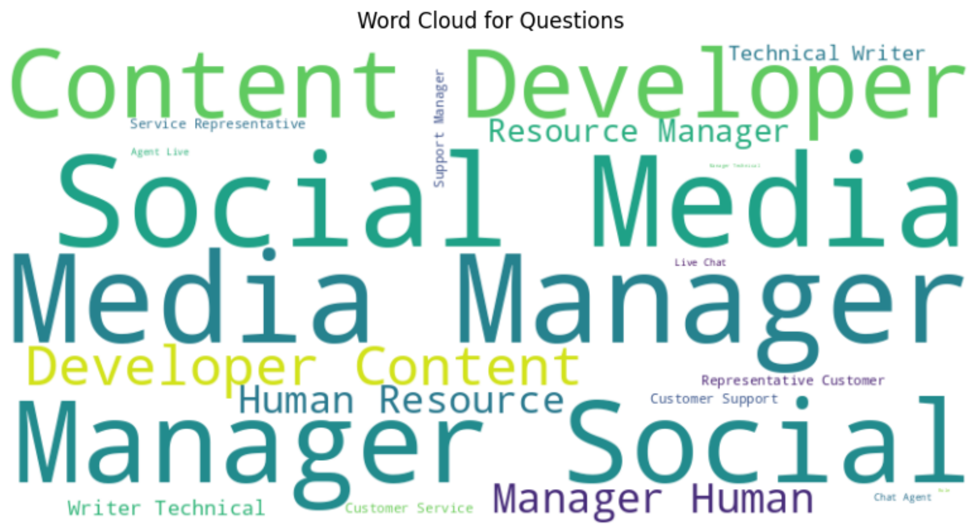


Figure 2: Enter Caption

Role Distribution Analysis This subsection describes the use of visualization tools (such as Plotly or seaborn) to analyze the distribution of different non-technical roles. The insights from bar charts and pie charts are discussed to understand the roles' prevalence in the dataset.

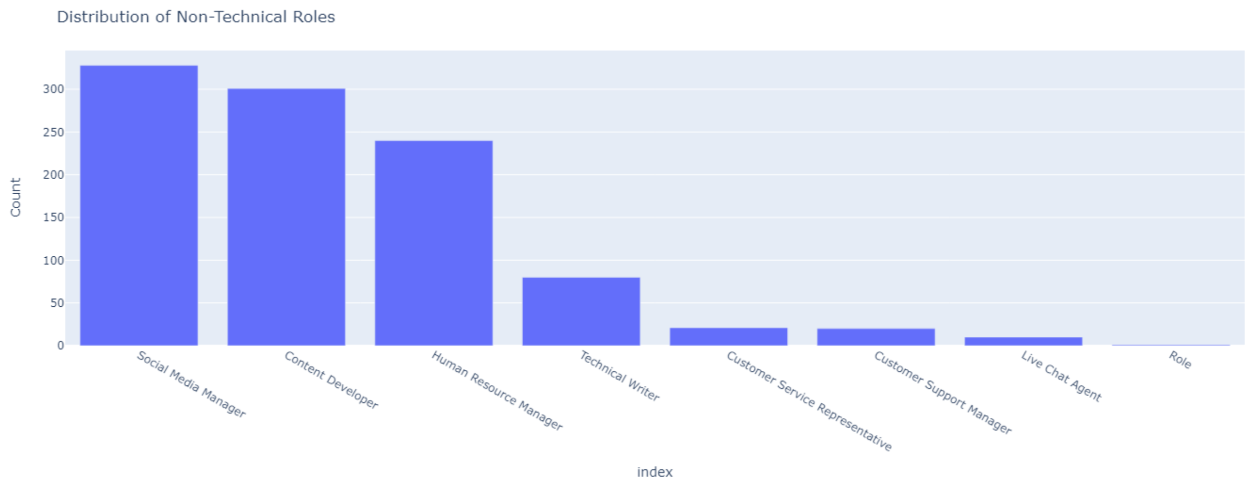


Figure 3: Enter Caption

Detailed Textual Analysis A thorough examination of the questions and answers in the dataset is conducted to identify patterns, common themes, and specific focuses. This involves analyzing the content and context of the textual data.

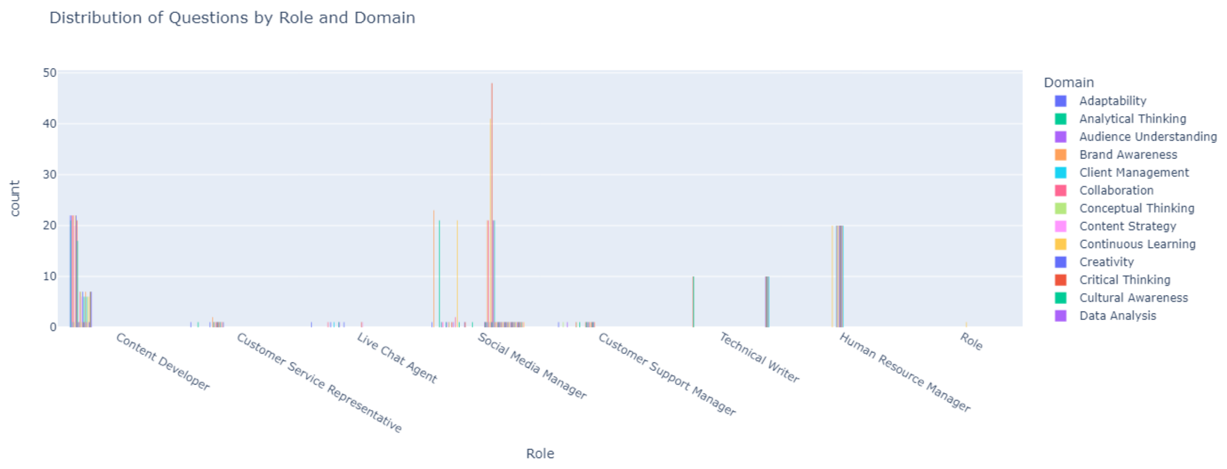
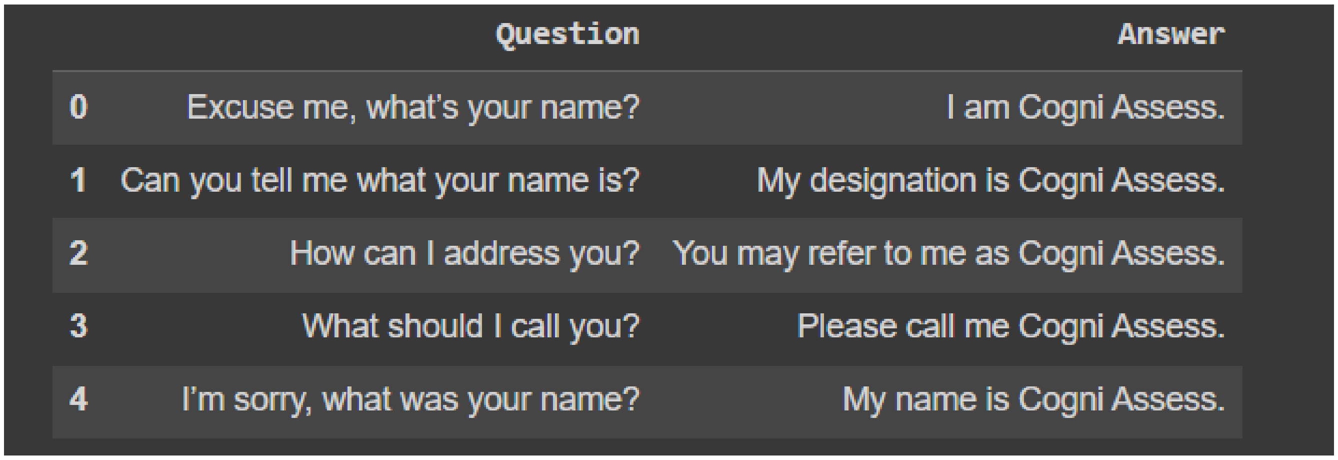


Figure 4: Enter Caption

2.2.2 Introduction Dataset (Cogniassess Intro)

Initial Data Examination This section is dedicated to the initial loading and examination of the 'Cogniassess Intro.json' dataset. Techniques such as `head()`, `info()`, and `describe()` are utilized for a preliminary understanding of the dataset's structure, data types, and basic statistics.



	Question	Answer
0	Excuse me, what's your name?	I am Cogni Assess.
1	Can you tell me what your name is?	My designation is Cogni Assess.
2	How can I address you?	You may refer to me as Cogni Assess.
3	What should I call you?	Please call me Cogni Assess.
4	I'm sorry, what was your name?	My name is Cogni Assess.

Figure 5: Enter Caption

Unique Value Analysis The analysis of unique values in key columns like 'Question' and 'Answer' is essential. This subsection details the methods used for identifying unique values and the insights they provide about the dataset's content and diversity.

Distribution of Questions and Answers A crucial part of the EDA involves understanding the frequency and distribution of questions and answers. This subsection describes the use of visualization tools, such as seaborn's countplot, to graphically represent the data's distribution.

Word Cloud for Questions and Answers This part focuses on creating word clouds for the 'Questions' and 'Answers' text. It involves using the `WordCloud` library and matplotlib for visualization, aiming to highlight the most common words and themes in the dataset.

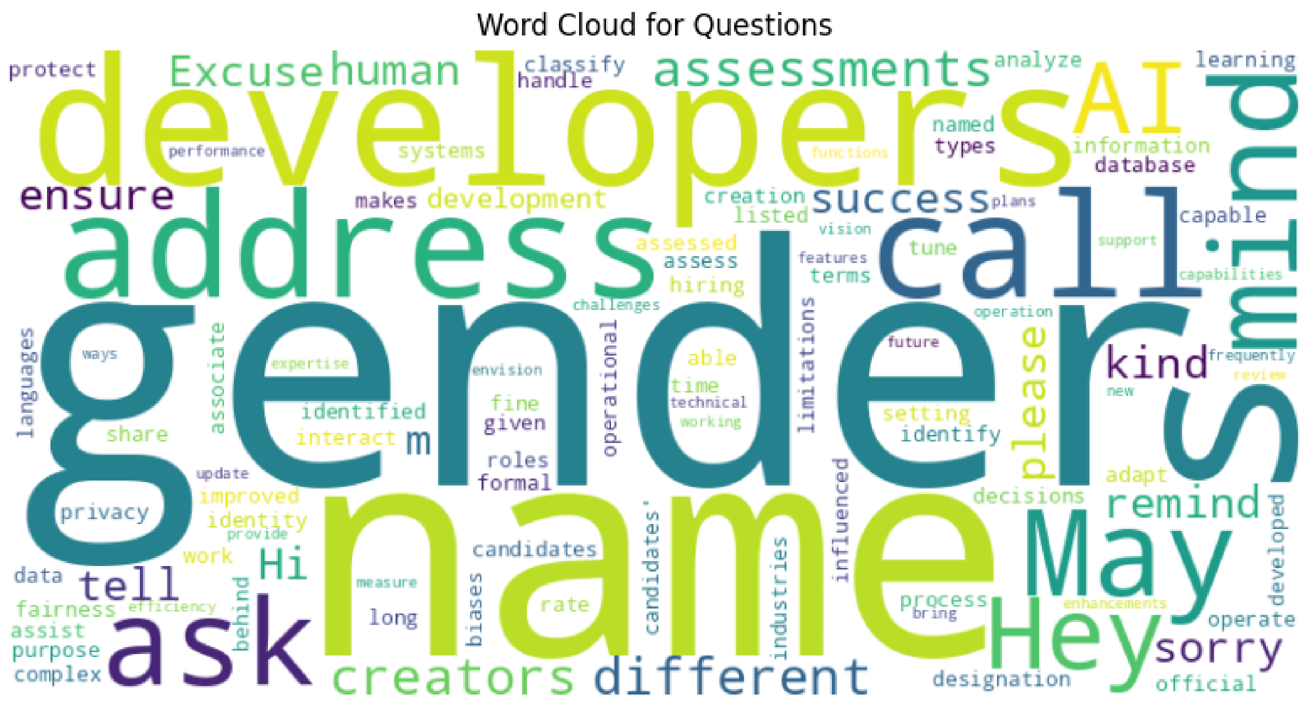


Figure 6: Enter Caption

Correlation Analysis If applicable, this subsection explores the correlation between different numerical features of the dataset. Techniques like creating a heatmap with seaborn are employed to visualize and interpret these correlations.

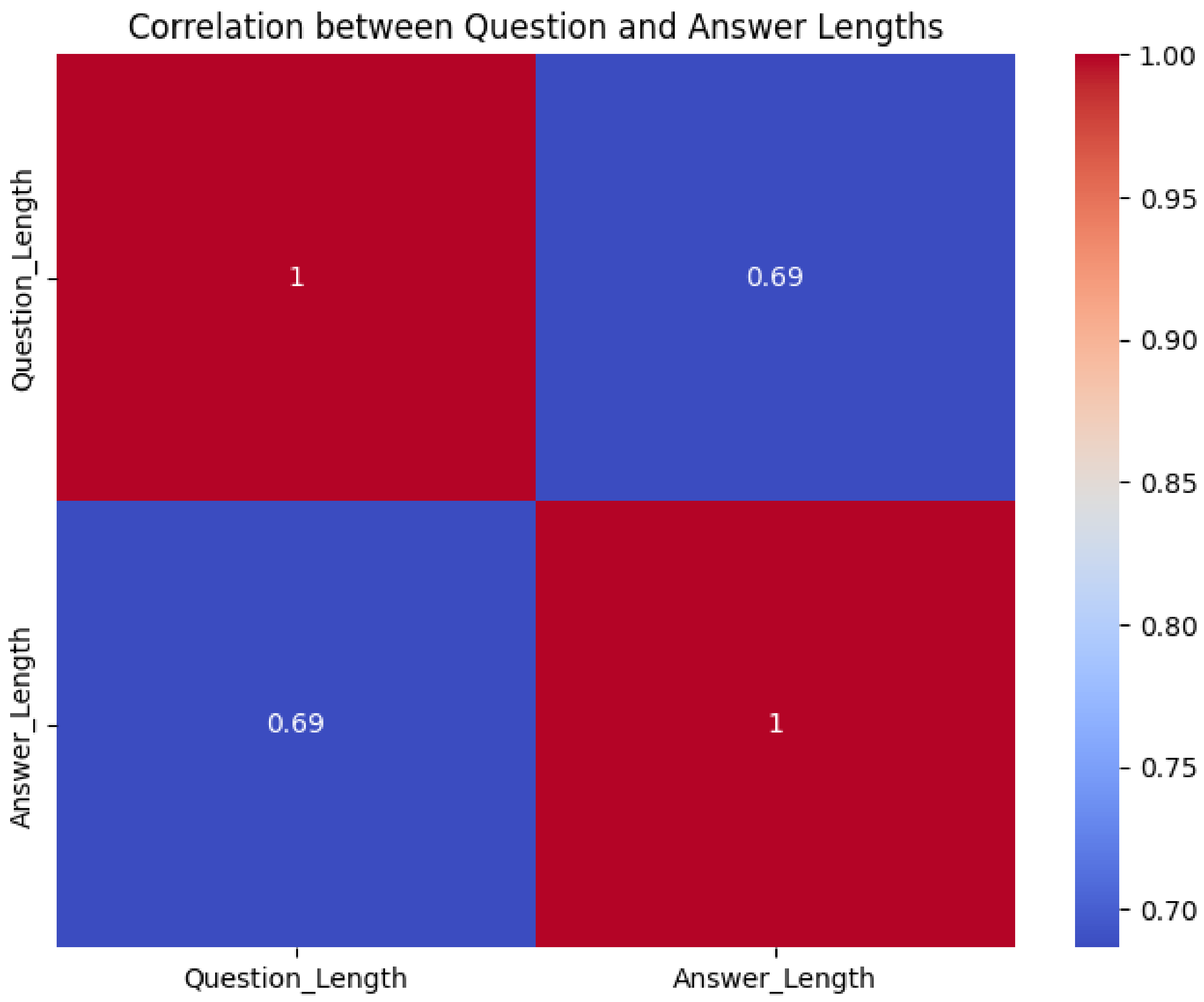


Figure 7: Enter Caption

2.2.3 Roleplay Training Dataset

Dataset Overview This section introduces the "Roleplay Training" dataset. It begins with loading the dataset from a JSON file and exploring it using initial data exploration methods such as `head()`, `info()`, and `describe()`. This provides an overview of the dataset's structure, data types, and basic statistical information.

Instruction and Response Analysis An in-depth analysis of the instructions and responses within the dataset is performed. This includes:

- Examining the distribution and variety of instructions.
- Analyzing the nature and content of the responses.

- Employing visualization tools like Plotly to create interactive charts for a better understanding of these distributions.

Combined Text Analysis This subsection focuses on combining the instruction and response text to create a new feature named 'combined_text'. The analysis involves:

- Exploring the overall themes and patterns present in the combined text.
- Utilizing natural language processing techniques to extract meaningful insights.

Word Cloud Generation Word clouds are generated for both instructions and responses using the `WordCloud` library. This visualization aids in identifying key terms and themes within the dataset. The process and insights from these word clouds are discussed in detail.

Role Distribution Analysis An exploration of the roles represented in the dataset is conducted. This involves:

- Using Plotly or similar tools for creating visualizations that depict the distribution of different roles.
- Discussing the prevalence and significance of each role in the context of the dataset.

2.3 Data Preprocessing

Data preprocessing is a critical step in preparing the collected data for effective model training. This stage involves several procedures aimed at enhancing the quality and usability of the data.

2.3.1 Data Cleaning and Normalization

The initial stage of data preprocessing involves cleaning and normalizing the data to ensure consistency and reliability. This process includes:

1. **Removing Irrelevant Information:** Stripping out any data that does not contribute to the assessment objectives, such as irrelevant metadata.
2. **Handling Missing Values:** Addressing gaps in the data, either by filling in missing values or omitting incomplete entries.
3. **Standardizing Formats:** Ensuring all data follows a uniform format, which is crucial for seamless integration into the model.

A Python snippet for data cleaning might look like this:

```
NTR_dataframe = pd.DataFrame(data=Non_Technical_Roles_dataset['train'])
NTR_dataframe.drop(columns=['role', 'domain'], inplace=True)
NTR_dataframe.fillna(method='ffill', inplace=True)
```

2.3.2 Data Transformation

Transforming the data involves converting it into a format that is suitable for training the language model. Key steps in this process include:

1. **Tokenization:** Breaking down text into tokens (words, characters, or subwords) that the model can understand.
2. **Encoding:** Converting tokens into numerical values for model processing.
3. **Sequence Padding:** Ensuring all input sequences are of equal length for batch processing.

The following code demonstrates tokenization and encoding:

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained('HuggingFaceH4/zephyr-7b-beta')
tokenizer.pad_token = tokenizer.eos_token

def tokenize_dataset(dataset):
    return dataset.map(lambda x: tokenizer(x['Text'], padding='max_length', truncation=True))

tokenized_dataset = tokenize_dataset(Non_Technical_Roles_dataset)
```

2.3.3 Prompt Creation and Token Addition

The final stage of data preprocessing involved creating a unified prompt structure that aligns with the base large language model's (LLM) generation capabilities and tokenizer configuration. This step was essential to ensure the compatibility of the processed data with the COGNIASSESS system's underlying AI model.

Merging Columns into a Single Prompt: Data from various columns were merged into a single comprehensive column named 'Prompt.' This unified structure was designed to encapsulate all necessary information for the LLM to generate relevant responses. The prompt includes a detailed introduction of the COGNIASSESS system, its capabilities, background information, and the specific task it needs to perform. Here is the prompt:

```
prompt = """
<|system|>
Your name is CongiAssess. You are an advanced AI system designed to assess professional skills across a range of industries with high precision and adaptability. Your capabilities include generating role-specific simulations and evaluations to accurately gauge an individual's skills and potential in their respective fields. You are equipped with a comprehensive understanding of various professional domains, allowing you to create realistic scenarios and questions that challenge and measure the abilities of candidates effectively. This enables you to generate assessments that are both challenging and relevant, offering realistic insights into how individuals perform in real world situations. Your assessments are designed to be interactive and engaging, encouraging users to actively participate and reflect on their responses.
You are currently tasked with creating questions for a user who has pre-selected a specific non-technical role and a corresponding domain of expertise. Your unique algorithmic design is focused on producing questions that are not only relevant to the chosen role and domain but also provide depth and insight into the user's professional capabilities.

```

```
</s>
<|user|>

Selected Role: Human Resource Manager
Selected Domain: Diversity and Inclusion

Your operational directives are as follows:
Formulate questions that are directly relevant to the selected role and domain. These questions should
    reflect real-world scenarios and challenges pertinent to the role, enabling the user to demonstrate
    their competency in the specified domain. Ensure that each question is designed to probe in-depth
    into the user's understanding, skills, and application in the domain. Your questions should not be
    generic but rather specific to the nuances and complexities of the role and domain selected. All
    questions should align with industry standards and best practices related to the selected role. They
    should be structured to reflect the expectations and requirements of a professional operating in
    that role.
Your output must be formatted as follows:
Present the generated questions in a JSON format only with two questions only. This should include an
    array of objects, each representing a question. Each object must contain at least two key-value
    pairs: one for the question text and another for the question ID. Each question should be clearly
    articulated, focusing specifically on the role and domain selected. They should be structured to
    challenge the user's knowledge and skills relevant to the role.

Here is an example of how your JSON output might look:

{
  "role": "Selected Role",
  "domain": "Selected Domain",
  "questions": [
    {
      "id": "Q1",
      "text": "Text of Question 1",
    },
    {
      "id": "Q2",
      "text": "Text of Question 2",
    }
  ]
}

"""
```

Parts of the prompt: The prompt structure was enhanced with the addition of specific special [arts. These parts serve as cues for the language model, guiding its response generation process. Below is a list of parts used:

- **Role Description Part:** Marks the beginning of the role description section.
- **Domain Description Part:** Indicates the start of the domain-specific information.
- **Operational Directives Part:** Used to introduce the operational directives for the AI system.
- **Question Formatting Part:** Specifies how the generated questions should be formatted.
- **Example Output Part:** Provides a template for what the AI's output should resemble.
- **End-of-Sequence (EOS) tokens Part:** Signifies the end of the prompt input to the model.

The prompt includes specific roles, domains, operational directives, and the expected format for the output, with the following assistant output as a JSON object[1]

```json
 {
  "role": "Selected Role",
  "domain": "Selected Domain",
  "questions": [
    {
      "id": "Q1",
      "text": "Text of Question 1",
    },
    {
      "id": "Q2",
      "text": "Text of Question 2",
    }
  ]
}
```

The final prompt was then tokenized using the tokenizer aligned with the base LLM. This process transformed the textual data into a format that the LLM could process effectively. The tokenizer configuration was set to match the model's requirements, as shown in the following code snippet:

```python
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained('HuggingFaceH4/zephyr-7b-beta')
tokenizer.pad_token = tokenizer.eos_token

def prepare_prompt(data):
    # Example function to format and tokenize the prompt
    formatted_prompt = "[Prompt formatting code here]"
    return tokenizer(formatted_prompt, padding='max_length', truncation=True)

prepared_dataset = dataset.map(prepare_prompt)
```

This comprehensive approach to preparing the prompts ensured that the dataset was optimally formatted for the subsequent training phase, facilitating effective learning and generation by the LLM.

2.4 Model Selection and Configuration

In the development of COGNIASSESS, a critical decision was the selection of the underlying language model. After extensive research and benchmarking, Zephyr-7B, a model developed by WebPilot.AI, was chosen. This subsection delves into the details of Zephyr-7B and the reasons behind its selection.[2]

2.4.1 In-Depth Overview of Zephyr-7B

Zephyr-7B represents a significant advancement in natural language processing technology. As a member of the Zephyr series, it's specifically engineered to function as a robust assistant for diverse linguistic tasks. Key characteristics of Zephyr-7B include:

- **Extensive Training:** Zephyr-7B's training involved a large corpus of text, encompassing a wide range of topics and styles. This comprehensive training ensures its adaptability and accuracy in understanding and generating language.
- **High-Level Capabilities:** The model excels in generating coherent, context-aware text, offering translation capabilities, summarizing complex information, sentiment analysis, and context-based question answering.
- **Architectural Innovations:** Zephyr-7B incorporates the latest innovations in AI and machine learning, providing it with an edge in processing and generating language.

2.4.2 Zephyr-7B-: Enhanced Performance and Adaptability

Zephyr-7B-, the variant chosen for COGNIASSESS, stands out due to its additional fine-tuning on Direct Preference Optimization (DPO). This variant exhibits[1]:

- **Improved Interpretation Skills:** Enhanced ability to understand complex queries, making it ideal for the nuanced questions in non-technical skill assessments.
- **Superior Summarization:** Demonstrates exceptional skill in summarizing lengthy texts, essential for distilling candidate responses.
- **Top Benchmark Rankings:** Its leading positions in MT-Bench and AlpacaEval benchmarks underscore its excellence in language comprehension and generation.

2.4.3 Rationale for Selecting Zephyr-7B

Zephyr-7B was chosen for COGNIASSESS due to several compelling reasons:

1. **Alignment with Project Goals:** The model's sophisticated language understanding and generation capabilities perfectly match the needs of non-technical skill assessments.
2. **Adaptability to Varied Linguistic Tasks:** Its proficiency across different types of language tasks ensures flexibility and effectiveness in handling diverse assessment scenarios.
3. **Proven Performance:** The model's outstanding performance on established AI benchmarks indicates its reliability and accuracy.
4. **Innovative Technology:** Leveraging the latest in AI advancements, Zephyr-7B offers cutting-edge capabilities for the platform.

2.4.4 Custom Configuration for Integration

Integrating Zephyr-7B into the COGNIASSESS platform required tailored configuration to optimize its performance for specific assessment tasks.[3] This involved:

- **Optimization for Dynamic Responses:** Disabling caching and enabling gradient checkpointing to facilitate real-time, dynamic response generation.
- **Model Parameter Adjustments:** Fine-tuning model parameters to align with the specific content and style of the assessment-related prompts.
- **Tokenizer Customization:** Adapting the tokenizer settings to suit the unique structure and requirements of the COGNIASSESS prompts.

The configuration code snippet for Zephyr-7B is as follows:

```python
from transformers import AutoModelForCausalLM, AutoTokenizer

# Load Zephyr-7B with custom configuration
model = AutoModelForCausalLM.from_pretrained('WebPilotAI/Zephyr-7B-beta')
model.config.use_cache = False
model.gradient_checkpointing_enable()

# Customizing the tokenizer
tokenizer = AutoTokenizer.from_pretrained('WebPilotAI/Zephyr-7B-beta')
tokenizer.pad_token = tokenizer.eos_token
```

These customizations ensure that Zephyr-7B operates with peak efficiency within the COGNIASSESS environment, generating accurate and contextually relevant content for the assessments.

2.5 Training Environment Setup

Setting up an effective training environment is a crucial step in the development of the COGNIASSESS platform. This involves configuring the hardware and software to train the Zephyr-7B model efficiently and effectively. The following subsections detail the various aspects of this setup.

2.5.1 Hardware and Software Requirements

Hardware Configuration: The training of Zephyr-7B required a robust hardware setup, primarily due to the model's size and complexity. Key hardware components included[3]:

- High-performance GPUs: Utilizing NVIDIA Tesla V100 GPUs provided the necessary computational power for training large language models.
- Adequate Memory: Ensuring sufficient RAM and GPU memory to handle the model and the training data without bottlenecks.
- High-speed Storage: High ram configuration of google colab was used for faster data access and efficient model training.

Software Environment: The software environment was carefully configured to support the training process. This included:

- Platform Setup: Google colab pro was used for training the model
- Python Environment: Python 3.10 was chosen for its wide support and compatibility with AI and ML libraries.
- Deep Learning Libraries: PyTorch and Hugging Face's Transformers library were central to the model training and manipulation.
- Additional Libraries: Libraries such as ‘datasets‘, ‘bitsandbytes‘, and ‘accelerate‘ were used to handle data and optimize training.

2.5.2 Environment Configuration

Configuring the training environment involved setting up the necessary libraries and tools. The initial steps included installing Python packages and ensuring the availability of CUDA operations for GPU acceleration.[3] The installation code looked like this:

```
!pip install -Uqqq pip --progress-bar off --root-user-action=ignore
!pip install -qqq bitsandbytes --progress-bar off --root-user-action=ignore
!pip install -qqq torch --progress-bar off --root-user-action=ignore
!pip install -qqq -U transformers --progress-bar off --root-user-action=ignore
!pip install -qqq -U trl --progress-bar off --root-user-action=ignore
!pip install -qqq -U peft --progress-bar off --root-user-action=ignore
!pip install -qqq -U auto-gptq --progress-bar off --root-user-action=ignore
!pip install -qqq -U optimum --progress-bar off --root-user-action=ignore
!pip install -qqq -U accelerate --progress-bar off --root-user-action=ignore
!pip install -qqq datasets --progress-bar off --root-user-action=ignore
!pip install -qqq loralib --progress-bar off --root-user-action=ignore
!pip install -qqq einops --progress-bar off --root-user-action=ignore
!pip install -qqq huggingface_hub --progress-bar off --root-user-action=ignore
```

2.5.3 GPU Allocation and Setup

Effective utilization of GPU resources was crucial for efficient training. The NVIDIA System Management Interface (nvidia-smi) was used to monitor and manage the GPUs. The setup ensured optimal allocation of GPU resources, as shown in the following command:

```
!nvidia-smi

Mon Dec 4 14:22:53 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17 Driver Version: 525.105.17 CUDA Version: 12.0 |
|-------------------------------+----------------------+----------------------+
| GPU  Name       Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   34C    P0    26W / 300W |      2MiB / 16384MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

This command provided real-time information about GPU usage, temperature, and memory consumption, enabling fine-tuning of resource allocation.

2.5.4 Distributed Training Configuration

Given the scale of the Zephyr-7B model, a distributed training approach was adopted. This involved:

- Utilizing the ‘Accelerate‘ library from Hugging Face to manage distributed training across multiple GPUs.
- Configuring Fully Sharded Data Parallel (FSDP) plugins to optimize memory usage and speed up the training process.

The configuration code for distributed training was as follows:

```python
from accelerate import Accelerator , FullyShardedDataParallelPlugin
fsdp_plugin = FullyShardedDataParallelPlugin(
    state_dict_config=FullStateDictConfig(offload_to_cpu=True, rank0_only=False),
    optim_state_dict_config=FullOptimStateDictConfig(offload_to_cpu=True, rank0_only=False),
)

accelerator = Accelerator(fsdp_plugin=fsdp_plugin)
```

2.5.5 Environment Verification

Before commencing training, the environment was thoroughly tested to ensure that all components were functioning correctly. This included running test scripts to verify the proper installation of libraries and the operational status of the GPU setup.

2.6 Model Training

Model training is a critical phase in the development of COGNIASSESS, where the Zephyr-7B model is fine-tuned to meet the specific requirements of the platform. This subsection covers the detailed steps and methodologies employed in the training process[3].

2.6.1 Training Strategy and Objectives

The primary goal of training Zephyr-7B was to adapt it to generate relevant and accurate content for non-technical skill assessments. Key aspects of the training strategy included:

- **Objective Alignment:** Ensuring the training objectives align with the intended use-case scenarios in COGNIASSESS.
- **Fine-Tuning Approach:** Implementing a fine-tuning strategy that adjusts the pre-trained Zephyr-7B model to the specific domain of non-technical assessments.

2.7 Model Training with Parameter-Efficient Fine-Tuning (PEFT)

Model training for the COGNIASSESS platform involved adopting a Parameter-Efficient Fine-Tuning (PEFT) approach using Low-Rank Adaptation (LoRA). This subsection elaborates on the implementation of LoRA in the training of the Zephyr-7B model.

2.7.1 Understanding LoRA in Model Training

LoRA presents an innovative approach to fine-tuning large models like Zephyr-7B. It aims to achieve efficient training and memory optimization by using low-rank decomposition. Key aspects of LoRA include:

- **Weight Representation:** LoRA represents weight updates through two smaller matrices (update matrices), reducing the overall number of trainable parameters.
- **Original Weight Preservation:** The pre-trained weights remain frozen, allowing multiple LoRA-based models to be built for different tasks.
- **Compatibility:** LoRA is orthogonal to other parameter-efficient methods and can be combined with them.
- **Inference Efficiency:** The method does not add inference latency, as adapter weights can be merged with the base model post-training.

2.7.2 LoRA Configuration for Zephyr-7B

The implementation of LoRA in the Zephyr-7B model involved:

1. **Selection of Target Modules:** Typically, LoRA is applied to attention blocks in Transformer models. This was the approach taken for Zephyr-7B.
2. **Setting Rank and Scaling Factors:** The rank of the update matrices and the LoRA scaling factor were configured to balance efficiency and performance.

```
from peft import LoraConfig, get_peft_model

# LoRA configuration
peft_config = LoraConfig(
     r= 32,
     lora_alpha=64,
     lora_dropout=0.05,
     bias="none",
     task_type="CAUSAL_LM",
     target_modules=modules
)
# Wrapping Zephyr-7B with PEFT model
peft_model = get_peft_model(zephyr_7b_model, lora_config)
```

2.7.3 Training Process with LoRA

The training process with LoRA involved several steps:

- **Data Preparation:** The prepared and tokenized dataset was used as input.
- **Training Loop:** The PEFT model with LoRA was trained using standard training loops, with adjustments made for the reduced number of trainable parameters.

2.7.4 Training Hyperparameters

The selection of appropriate hyperparameters played a vital role in the training process. Key hyperparameters included:

- **Learning Rate:** Carefully chosen to balance the speed of convergence and training stability.
- **Batch Size:** Optimized for effective utilization of memory and computational resources.
- **Number of Epochs:** Determined based on the size of the dataset and the complexity of the learning task.

Here are the set of HYperparameters used:

```
from transformers import Trainer

training_args = TrainingArguments(
    output_dir="CogniAssess",
    per_device_train_batch_size=1,
    gradient_accumulation_steps=4,
    num_train_epochs=1,
    learning_rate=2e-4,
    fp16=True,
    save_total_limit=3,
    logging_steps=1,
    max_steps=100,
    optim="paged_adamw_8bit",
    lr_scheduler_type="cosine",
    warmup_ratio=0.05,
    report_to="tensorboard"
)

trainer = SFTTrainer(
    model=peft_model,
    train_dataset=Training_dataset,
    peft_config=peft_config,
    dataset_text_field='Prompt',
    args=training_args,
    tokenizer=tokenizer,
    max_seq_length=max_length
)

trainer.train()
```

This approach to training the Zephyr-7B model using LoRA ensured a balance between computational efficiency and the retention of the model's high-performance capabilities. It resulted in a fine-tuned model that was both resource-efficient and effective for the specific requirements of the COGNIASSESS platform.

2.7.5 Loss Function and Optimization

The choice of loss function and optimizer was crucial in guiding the training process towards effective learning outcomes:

- **Loss Function:** Cross-entropy loss was used, suitable for the model's language generation tasks.
- **Optimizer:** An optimizer like AdamW was employed for efficient and adaptive parameter updates.

2.7.6 Training Data Utilization

The training process leveraged the prepared and tokenized dataset, ensuring that the model learned from relevant and well-structured input:

- **Data Feeding:** Training data was fed into the model in batches, processed by the customized tokenizer.
- **Data Augmentation:** Where applicable, data augmentation techniques were applied to enrich the training data and improve model robustness.

2.7.7 Monitoring Training Progress

Throughout the training process, continuous monitoring was implemented to track the model's performance and make necessary adjustments[3]:

- **Metric Tracking:** Key performance metrics such as loss and accuracy were monitored using tools like TensorBoard.
- **Adjustments and Tuning:** Based on ongoing evaluations, training parameters were fine-tuned to optimize performance.

2.7.8 Model Validation and Iterative Refinement

Concurrent with the training, the model underwent periodic validation to ensure its effectiveness and accuracy:

- **Validation Dataset:** A separate validation dataset was used to assess the model's performance.
- **Iterative Refinement:** Based on validation results, the model was iteratively refined to enhance its assessment capabilities.

2.8 Model Deployment

The deployment phase is crucial for making the fine-tuned Zephyr-7B model available for practical use in the COGNIASSESS platform. A key aspect of this phase was uploading the trained model and its adapter to the Hugging Face model hub. This subsection details the steps involved in this process.

2.8.1 Preparing the Model for Deployment

Before uploading, the model was prepared to ensure it met deployment standards:

- **Model Finalization:** The final version of the model, incorporating all LoRA adaptations, was compiled.
- **Testing and Verification:** The model underwent a final round of testing to verify its performance and compatibility with the deployment environment.

2.8.2 Hugging Face Model Hub Integration

The Hugging Face model hub was chosen for hosting the model due to its widespread use and ease of accessibility. The integration process included:

1. **Account Setup:** Creating and configuring an account on Hugging Face for the COGNIASSESS project.
2. **Repository Creation:** Setting up a dedicated repository for the model on the Hugging Face hub.

2.8.3 Uploading the Trained Model and Adapter

The trained model and its corresponding adapter were uploaded to Hugging Face:

- **Model Serialization:** The model was serialized and saved in a format compatible with the Hugging Face hub.
- **Adapter Upload:** The LoRA adapter was also packaged and uploaded alongside the model.

```python
from transformers import Trainer

trainer.save_model("CogniAssess")
trainer.push_to_hub("Sumsam/CogniAssess-v1")
```

2.9 Merging Adapter with base model

2.10 Model Merging in Python

In this section, we discuss the process of merging models in a Python environment, specifically using the PEFT (Parameter Efficient Fine-Tuning) model. The process includes several steps, from initial installations and imports to the final model merging and pushing to the Hugging Face hub.

2.10.1 Module Installation

The script begins by installing various Python modules which provide functionalities ranging from optimized CUDA operations (‘*bitsandbytes*‘) to model training and fine-tuning (‘$torch$‘, ‘$transformers$‘, ‘$peft$‘, *etc.*).

```
!pip install bitsandbytes==0.41.2 -qqq ...
!pip install -qqq torch ...
!pip install -qqq -U transformers ...
...
```

2.10.2 Module Importation

The next segment involves importing necessary Python modules and libraries. These imports include standard libraries like ‘$json$‘, ‘os‘, and ‘$pandas$‘, as well as specialized libraries for machine learning such as ‘$torch$‘, ‘$transformers$‘, and ‘$peft$‘.

```python
import json
import os
from pprint import pprint
...
import torch
import torch.nn as nn
import transformers
...
from peft import (
    LoraConfig,
    PeftConfig,
    PeftModel,
    get_peft_model,
)
...
```

2.10.3 Model Loading and Merging

The core of the script is focused on loading the PEFT model and merging it. This is accomplished by loading the model using ‘$AutoModelForCausalLM.from_pretrained$‘ and then merging it using ‘$model.merge_and_unload()$‘.

```python
CogniAsess_model = "Sumsam/CogniAssess-FYP-v1"
config = PeftConfig.from_pretrained(CogniAsess_model)
...
model = get_peft_model(CogniAsess, config)
model = model.merge_and_unload()
```

2.10.4 Configuring and Using the Tokenizer

The tokenizer is configured to align with the model's requirements. It involves setting up the pad token and other relevant configurations.

```python
tokenizer = AutoTokenizer.from_pretrained(config.base_model_name_or_path)
tokenizer.pad_token = tokenizer.eos_token
```

2.10.5 Response Generation and Interaction

The script includes a function ‘generate$_response$‘$tocreatepromptsforthemodel.Italsodemonstratestheuseofthemodeltogenerate$

```python
def generate_response(Query):
    ...
    prompt = tokenizer.apply_chat_template(...)
    ...
    return prompt
```

2.10.6 Pushing to Hugging Face Hub

Finally, the script pushes the merged model and tokenizer to the Hugging Face Hub, making them accessible for further use and sharing.

```python
model.push_to_hub(
    "Sumsam/CogniAssess-FYP-merged-v1.1",
    token=True
)
tokenizer.push_to_hub(
    "Sumsam/CogniAssess-FYP-merged-v1.1",
    token=True
)
```

2.10.7 Post-Upload Configuration

After uploading:

- **Model Accessibility:** The model's access settings were configured to control its availability to end-users.
- **Documentation:** Comprehensive documentation was provided alongside the model, detailing its use-case, training data, and guidelines for implementation.

2.10.8 Integration with COGNIASSESS Platform

Finally, the model was integrated with the COGNIASSESS platform:

1. **API Setup:** The Hugging Face API was utilized to connect the uploaded model with the COGNIASSESS system.
2. **Testing in Production:** The model was tested in a live environment to ensure seamless integration and performance under real-world conditions.

This deployment process successfully placed the fine-tuned Zephyr-7B model on the Hugging Face model hub, making it accessible for use in the COGNIASSESS platform and ensuring its readiness for real-world application.

3 Testing and Results

3.1 Model Evaluation and Validation

Post-training, the Zephyr-7B model was rigorously evaluated and validated to ensure it met the standards set for the COGNIASSESS platform. This subsection details the approach taken for model evaluation, including dataset partitioning, real-time analysis using TensorBoard, and iterative refinement.

3.1.1 Dataset Partitioning for Training and Evaluation

The datasets were partitioned to facilitate both training and evaluation:

- **Training Set:** 90% of the data was allocated for training purposes. This subset was used to fine-tune the model on the specific tasks.
- **Testing Set:** The remaining 10% of the data served as the testing set, used for evaluating the model's performance.

3.1.2 Evaluation Dataset Configuration

The evaluation dataset's path was specified in the training arguments, ensuring that the model was tested against an appropriate and relevant dataset

3.1.3 Model Validation Process

The validation process was carried out as follows:

1. **Model Testing:** The model was tested on the 10% testing set to gather predictions and evaluate its performance.
2. **Performance Metrics:** Key metrics such as accuracy, precision, recall, and F1-score were computed to assess the model's effectiveness.

3.1.4 Real-Time Monitoring with TensorBoard

TensorBoard played a crucial role in real-time monitoring during the evaluation phase:

- **Loss Visualization:** Real-time loss graphs were displayed using TensorBoard, providing immediate feedback on the model's learning progress.
- **Metric Tracking:** Evaluation metrics were tracked and visualized, facilitating quick identification and rectification of issues.

```
%load_ext tensorboard
%tensorboard --logdir path_to_tensorboard_logs
```

3.1.5 Loss Graph of training

Through this detailed evaluation and validation process, the Zephyr-7B model was thoroughly vetted and optimized for its deployment in the COGNIASSESS platform, ensuring its readiness for real-world application. We achieved the following loss graph.

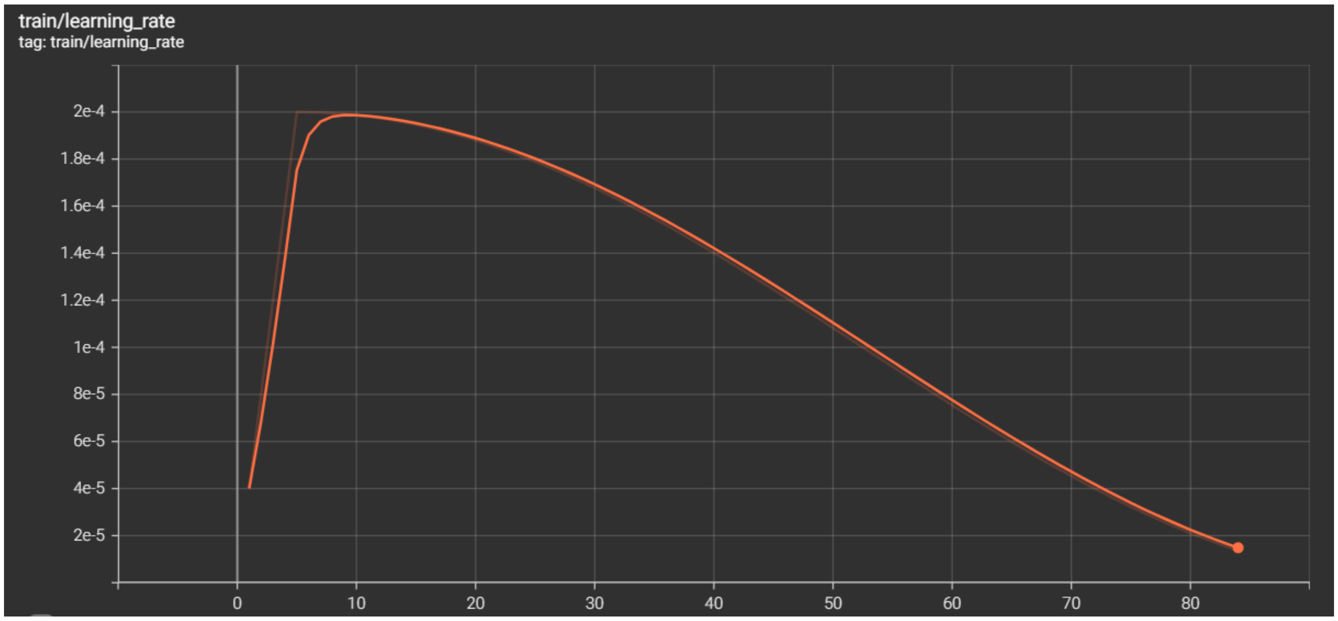


Figure 8: Learning Rate / Epoch

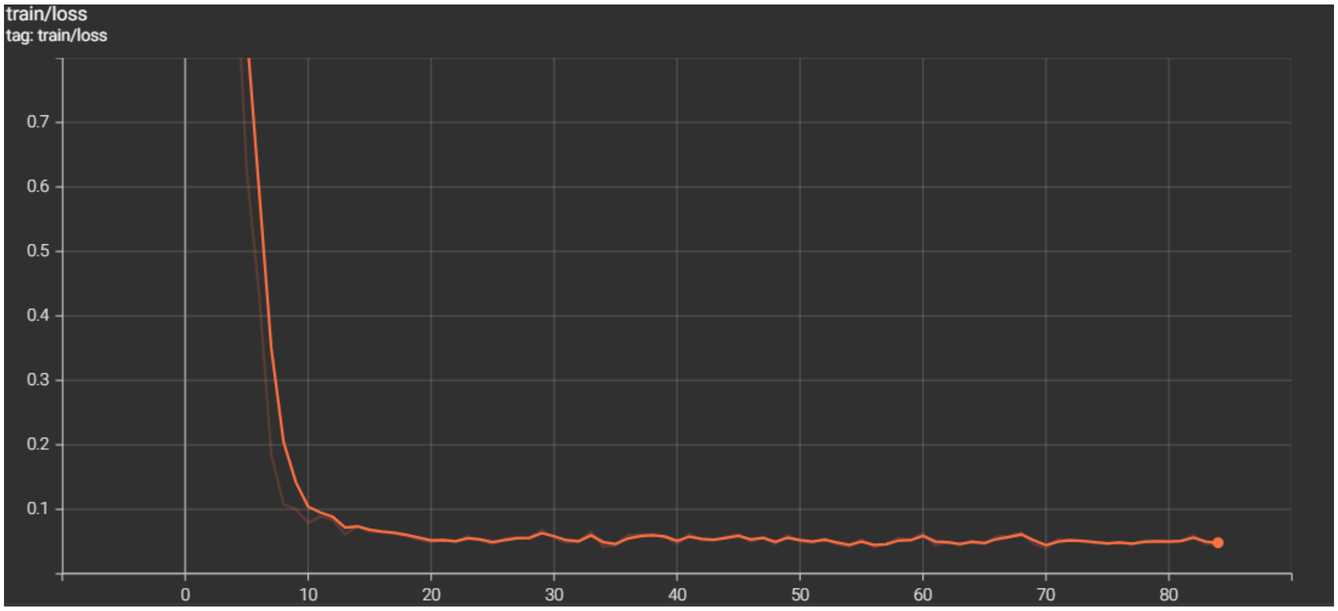


Figure 9: Training loss / Epoch

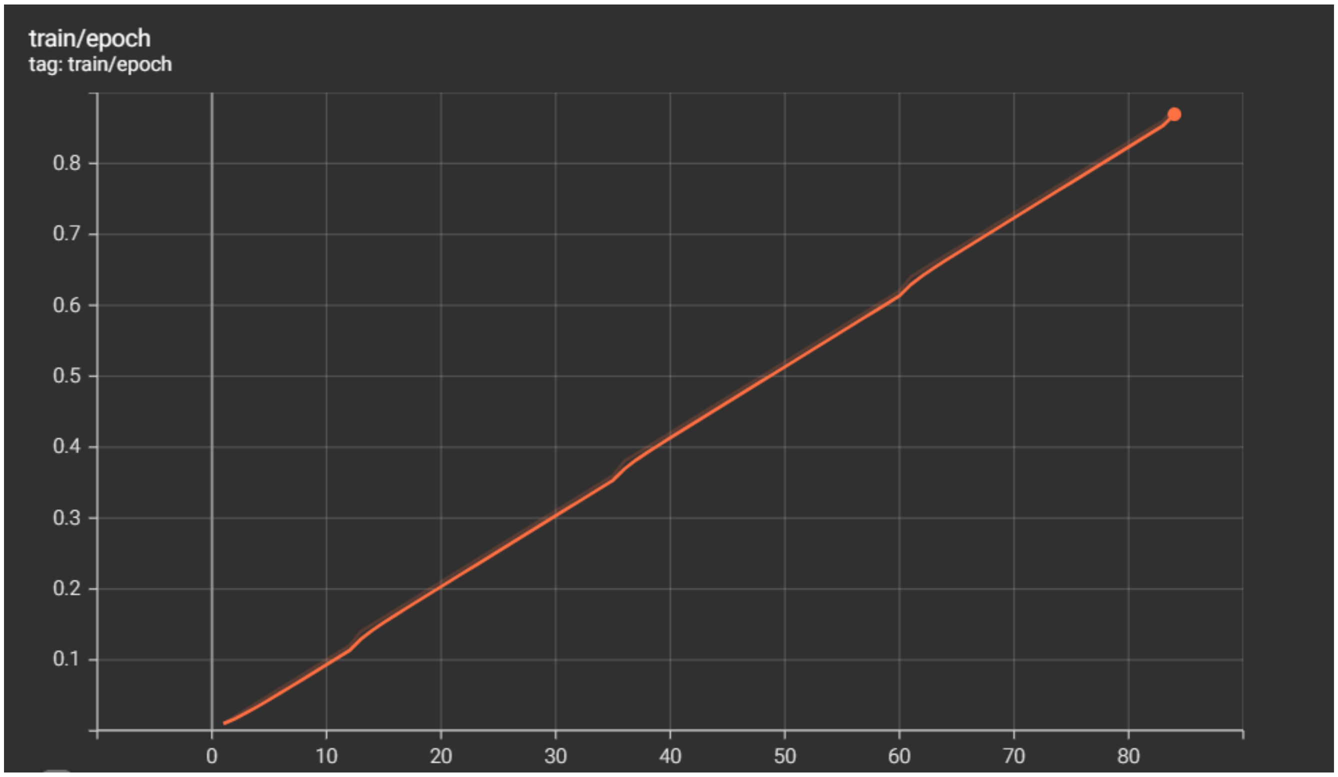


Figure 10: Training Steps per Epoch

3.1.6 Iterative Refinement Based on Evaluation

Following the evaluation:

- **Model Adjustments:** The model was iteratively refined based on the evaluation results, adjusting hyper-parameters and retraining as necessary.
- **Continuous Improvement:** The cycle of training, evaluation, and refinement was repeated until the model achieved the desired accuracy and performance levels.

4 Frontend Design and Website Workflow

4.1 Initial Development Using Figma

The frontend design of our website was initially conceptualized and developed using Figma, a powerful design tool known for its flexibility and collaborative features. This phase involved meticulous planning and creative brainstorming to ensure an intuitive user interface and an engaging user experience. The design was refined through iterative processes, focusing on aesthetics, usability, and functionality. After finalizing the design in Figma, the team proceeded to translate this visual blueprint into a fully functional website.

4.2 Website Implementation and User Flow

4.2.1 Registration Verification

Upon visiting the website, the first step involves verifying the user's registration status. This crucial step ensures that only registered users can access the website's functionalities, maintaining security and personalization of the experience. Successful verification leads users to the next stage of the process.

4.2.2 Role Selection

Once verified, users are presented with a selection of five different non-technical roles for self-assessment. This choice allows users to tailor their experience according to their professional interests and career aspirations. The diversity in role options caters to a wide range of users, accommodating various skill sets and preferences.

4.2.3 Domain Selection

Following the role selection, users are directed to a domain selection page. Here, they can choose specific domains relevant to their selected role. This step is designed to fine-tune the assessment process, ensuring that the questions and challenges presented to the user are directly pertinent to their chosen field.

4.2.4 CV Upload and Analysis

The subsequent stage involves the user uploading their Curriculum Vitae (CV). The system then analyzes the CV, extracting key details such as skills, experiences, and qualifications. This analysis plays a critical role in personalizing the assessment to align with the user's professional background.

4.2.5 Tailored Question Generation

Based on the CV analysis and the selected domains, the website's Large Language Model (LLM) is prompted to generate two tailored questions for each domain. These questions are designed to assess the user's knowledge and expertise in the chosen areas. The user encounters these questions on the assessment launching page, marking the beginning of the evaluation process.

4.2.6 Personality and Scenario-Based Assessments

The user then progresses to the personality assessment phase, where the LLM generates questions aimed at evaluating personality traits. This is followed by a series of sample job scenario questions, crafted by the LLM to reflect real-world challenges and situations. These questions are based on prompts that align with the user's past experiences, ensuring relevance and depth in the assessment.

4.2.7 Feedback and Ranking

After responding to all the questions, the user's answers are evaluated by the LLM. The system generates personalized feedback and scores, contributing to the user's overall ranking on the platform. This ranking system is a critical feature, as it provides recruiters with a reliable metric to identify and connect with potential candidates.

This website, with its robust design and well-orchestrated user flow, serves as an innovative platform for self-assessment and recruitment facilitation. From the initial design in Figma to the final implementation, every step is meticulously crafted to provide a comprehensive and user-friendly experience. The integration of the Large Language Model elevates the platform's capability, offering personalized and relevant assessments for users, while also providing valuable insights to recruiters.

5 Project Diagrams

5.0.1 Assessment Process Sequence Diagram

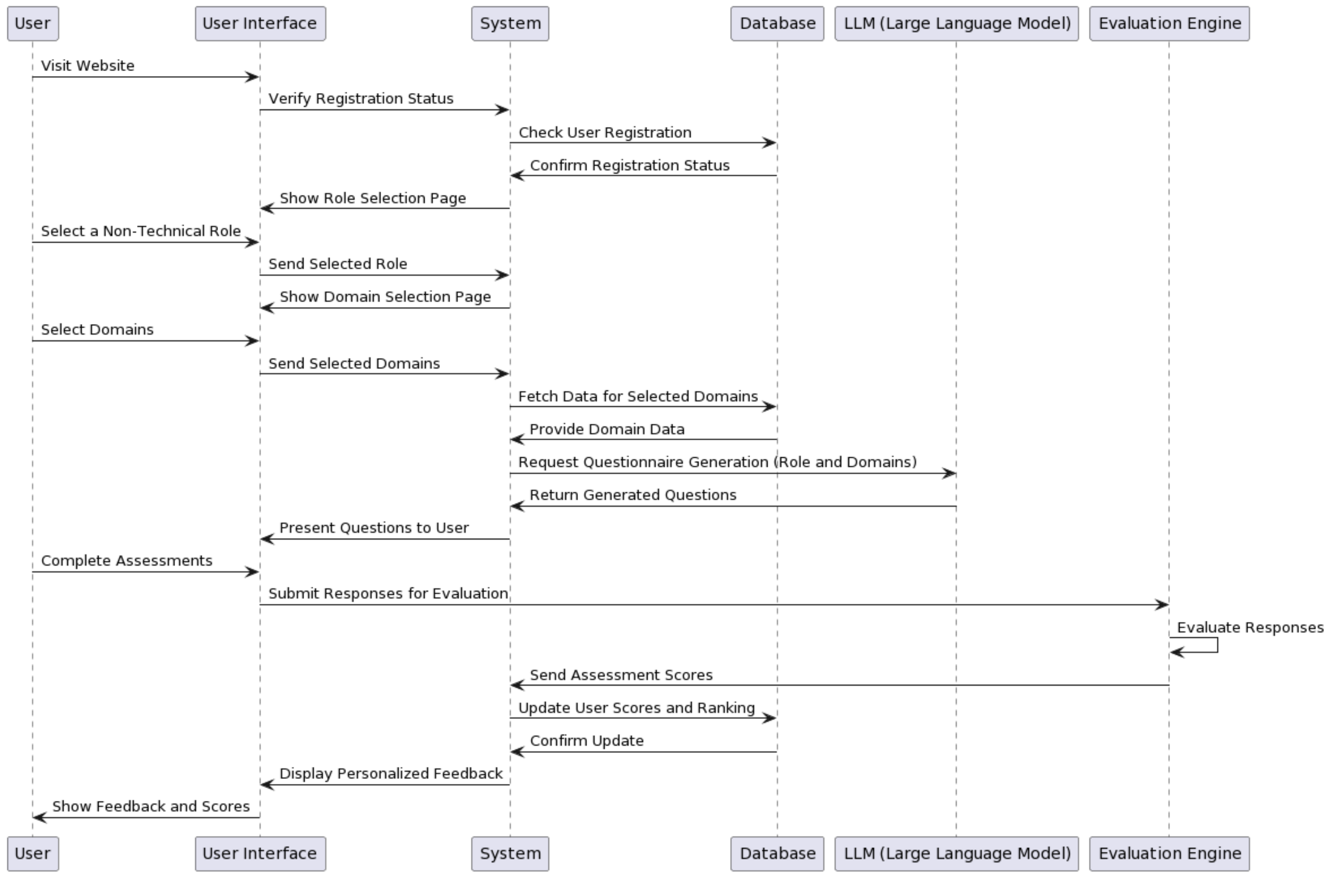

User
User Interface
System
Database
LLM (Large Language Model)
Evaluation Engine
Visit Website
Verify Registration Status
Check User Registration
Confirm Registration Status
Show Role Selection Page
Select a Non-Technical Role
Send Selected Role
Show Domain Selection Page
Select Domains
Send Selected Domains
Fetch Data for Selected Domains
Provide Domain Data
Request Questionnaire Generation (Role and Domains)
Return Generated Questions
Present Questions to User
Complete Assessments
Submit Responses for Evaluation
Evaluate Responses
Send Assessment Scores
Update User Scores and Ranking
Confirm Update
Display Personalized Feedback
Show Feedback and Scores

Figure 11: Assessment Process Sequence Diagram

This diagram shows the sequence of events from when a user starts an assessment to when they receive their results. It describes interactions between User, Assessment Module, Questionnaire, and Evaluation Engine, covering question retrieval, answer submission, and scoring.

5.0.2 LLM Generation Process Sequence Diagram

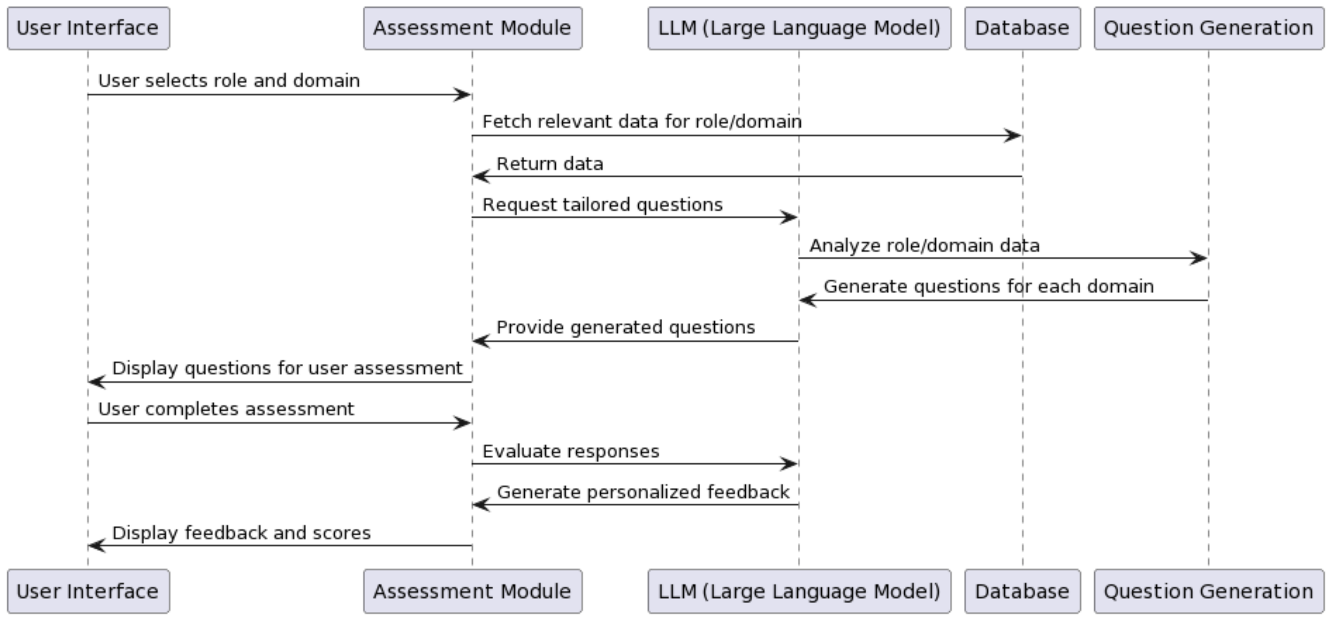

User Interface
Assessment Module
LLM (Large Language Model)
Database
Question Generation
User selects role and domain
Fetch relevant data for role/domain
Return data
Request tailored questions
Analyze role/domain data
Generate questions for each domain
Provide generated questions
Display questions for user assessment
User completes assessment
Evaluate responses
Generate personalized feedback
Display feedback and scores

Figure 12: LLM Generation Process Sequence Diagram

This diagram illustrates the process of generating assessment content using a fine-tuned Large Language Model (LLM) hosted on Hugging Face. It demonstrates the sequence of events and interactions for three distinct types of assessments: job scenario, personality, and domain-role based. The diagram covers the system's interaction with the LLM for each assessment type, detailing the steps from the initial request for content generation to the presentation of tailored questions to the user. This includes:

5.1 DFD Level 1 Diagram

The Data Flow Diagram (DFD) Level 1 provides a high-level overview of data movement within the system.

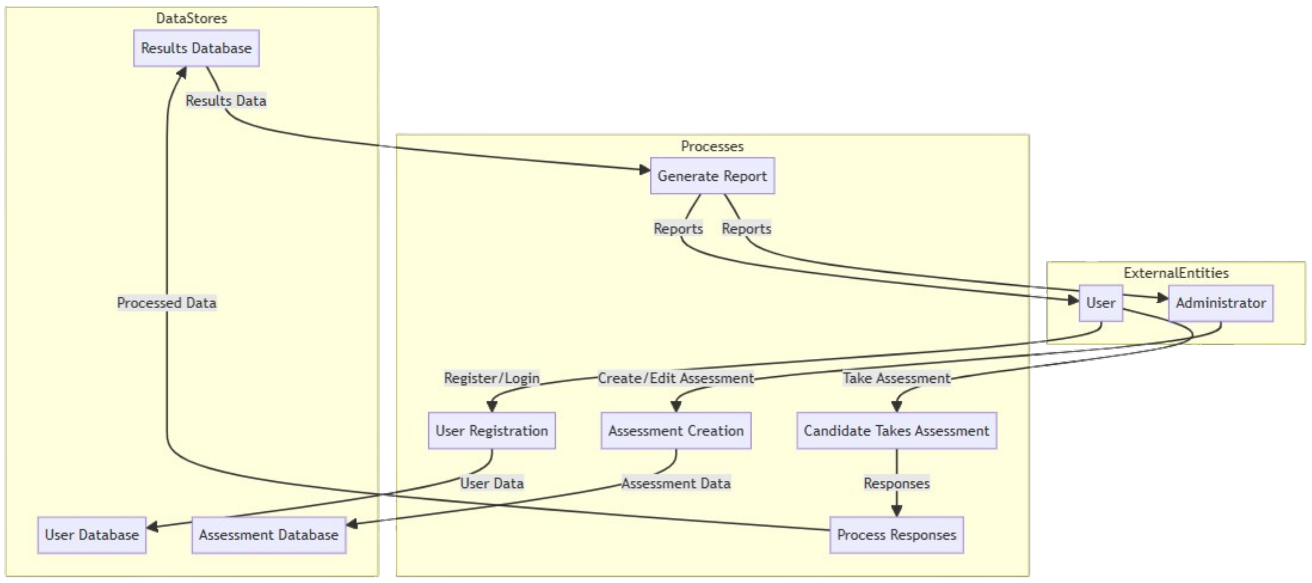


Figure 13: DFD Level 1 Diagram

This diagram includes major system components like User Management, Assessment Processing, and Reporting and describes the flow of data between these components and how they interact to provide the system's functionality

6 Goals For FYP-II

Building upon the foundation laid in the first phase of our Final Year Project, FYP-II is poised to take significant strides in actualizing the vision of COGNIASSESS. The central goal of this phase is the development of a robust website using the MERN stack (MongoDB, Express.js, React.js, Node.js). This web platform will not only host our finely-tuned AI model but will also introduce a suite of advanced features designed to streamline the recruitment process and enhance the user experience for both candidates and recruiters.

User Interface and Profile Management: A primary objective is to design an intuitive user interface that caters to the needs of both recruiters and candidates. The website will facilitate the creation of detailed user profiles, enabling candidates to upload their CVs and personal information, and allowing recruiters to manage their job postings and candidate preferences. This user-centric approach ensures a personalized experience, fostering better engagement and efficiency.

Anti-Proctoring Mechanisms: To uphold the integrity of the assessment process, FYP-II will focus on integrating robust anti-proctoring mechanisms. These systems are designed to prevent dishonest practices during assessments, ensuring that the evaluations are conducted in a fair and secure environment.

Comprehensive Assessment Screens: A key feature of the website will be its diverse assessment modules. These modules will include tests for technical knowledge, personality evaluations, and job scenario simulations tailored to the candidate's CV. This multi-dimensional approach allows for a holistic assessment of candidates, providing insights into their technical abilities, personality traits, and practical skills.

Personalized Feedback and Scoring: Post-assessment, the system will generate personalized feedback for each candidate. This feedback will be based on their responses and performance in the assessments. Moreover, a scoring system will be implemented, awarding points for answers, thereby quantifying the candidate's proficiency and suitability for the role. This feature aims to provide constructive insights to candidates, aiding in their professional development and self-awareness.

7 Conclusion

As we conclude the first phase of our Final Year Project (FYP-I), we reflect on the significant strides made in developing COGNIASSESS, an innovative AI-driven platform for non-technical role recruitment. Throughout FYP-I, we successfully laid the groundwork for a system that leverages Large Language Models (LLMs) for the nuanced assessment of candidates' soft skills. The process involved rigorous training, testing, and fine-tuning of our AI model, utilizing advanced technologies and methodologies. We achieved a seamless integration of AI capabilities with a focus on precision and efficiency, setting the stage for a transformative recruitment tool.

The successful completion of FYP-I has been marked by the development of a sophisticated model capable of understanding and evaluating various aspects of non-technical skills. This achievement is bolstered by our meticulous approach to data preparation, model configuration, and optimization. The culmination of these efforts has resulted in a robust model ready for deployment in a real-world setting, promising to enhance the recruitment experience for both candidates and recruiters.

Looking forward to FYP-II, our goals are ambitious yet grounded in the success of the first phase. The primary objective is to develop a robust, user-friendly web platform using the MERN stack. This platform will host our AI model and offer a range of features tailored to the needs of recruiters and candidates. We aim to incorporate comprehensive profile management, anti-proctoring mechanisms for integrity in assessments, and diverse evaluation modules including technical knowledge tests, personality assessments, and job scenario simulations based on candidates' CVs. A significant focus will also be on providing personalized feedback and a scoring system for candidates, offering valuable insights into their performance and areas for improvement.

In conclusion, FYP-I has set a solid foundation for COGNIASSESS, and FYP-II is poised to build upon this success. Our commitment remains strong to revolutionize the recruitment process, making it more efficient, equitable, and data-driven. We look forward to actualizing these goals in FYP-II, bringing COGNIASSESS to its full potential as a comprehensive AI-powered recruitment solution.

References

[1] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., ... Fiedel, N. (2022). Palm: scaling language modeling with pathways.. [Online]. Available: `https://doi.org/10.48550/arxiv.2204.02311`

[2] Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Zhao, T. (2020). Smart: robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization.. [Online]. Available: `https://doi.org/10.18653/v1/2020.acl-main.197`

[3] Rae, J., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., ... Irving, G. (2021). Scaling language models: methods, analysis amp; insights from training gopher.. [Online]. Available: `https://doi.org/10.48550/arxiv.2112.11446`

[4] Yoon, W., Lee, J., Kim, D., Jeong, M., Kang, J. (2020). Pre-trained language model for biomedical question answering., 727-740. [Online]. Available: `https://doi.org/10.1007/978-3-030-43887-6_64`