LLM Project Report Template

---

1. Student Information

- **Student Name: Alissa Tran**
- **Student ID (optional): 223699665**
- **Date Submitted: 14/05/2025**

---

2. Project Introduction

- **Title of the Project:** Medical Classification and Misinformation Detection using Llama-2-13B Model

- **What is the project about?**

  The project focuses on enhancing the performance of Large Language Models to detect misleading or incorrect medical advice.

- **Why is this project important or useful?**

This project addresses the issue of misinformation and incorrect medical information by leveraging large language models (LLMs), specifically Llama-2-13B, to detect and flag misleading or incorrect health-related claims. By improving the ability of machines to identify misinformation, this project supports safer information ecosystems, aids fact-checkers, and contributes to the development of trustworthy AI tools for healthcare communication.

---

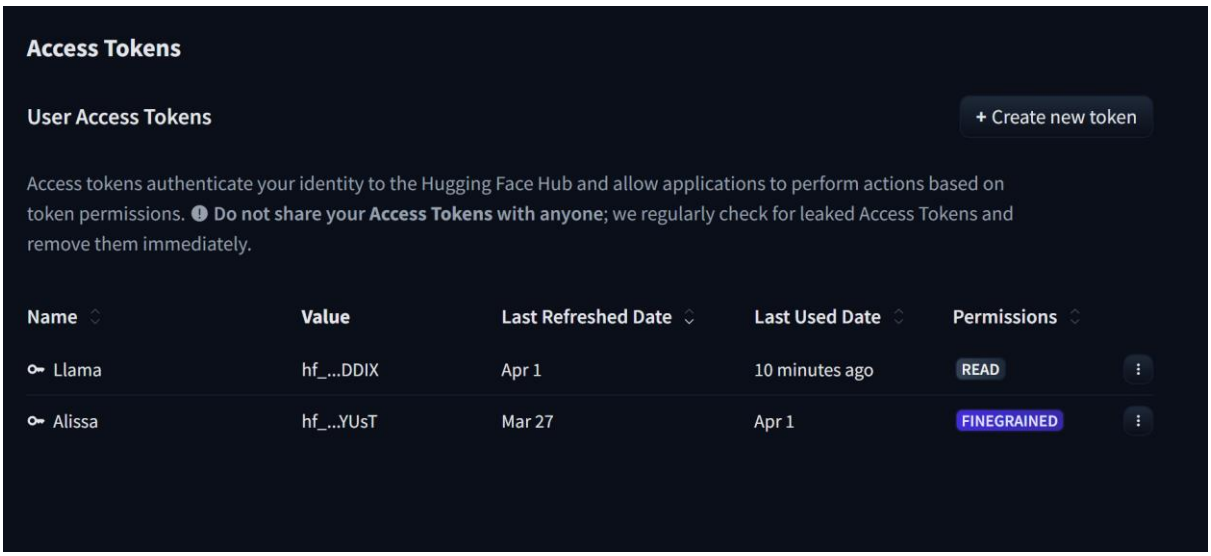
3. API/Token Setup — *Step-by-Step*

Objective: Show how you obtained and securely used an API token for LLM access.

Instructions:

1. Specify which provider you're using:
   - OpenAI
   - **Hugging Face**
   - Google Vertex AI
   - Other: _____
2. List the **steps you followed** to generate the token:
   - Step 1: Created account at https://huggingface.co/
   - Step 2: Navigated to the API/token section
   - Step 3: Clicked on "Create new key"
   - Step 4: Copied the key and securely saved it

3. **Screenshot or terminal output (required):**
   (Blur/redact the actual key)



4. **Secure Loading of Token in Code:**
   Avoid hardcoding tokens — use os.environ or .env files.

```
#Logging into Hugging Face using token
import torch
from transformers import (
    AutoModelForCausalLM,
    AutoTokenizer,
    TrainingArguments,
    Trainer,
    DataCollatorForLanguageModeling
)
import datasets
import peft
import numpy as np
import os

from huggingface_hub import login

hf_token = os.environ.get("Llama")
login(token=hf_token, add_to_git_credential=True)

print("Hugging Face login successful!")

# Check GPU
device = "cuda" if torch.cuda.is_available() else "cpu"
print("Using device:", device)
```

```
Hugging Face login successful!
Using device: cuda
```

---

4. Environment Setup

- **Development Platform:**
  - **Google Colab**
  - Local Machine (Windows/Linux/macOS)

- GPU Available? [ ] **Yes** [ ] No
- GPU Type (if applicable):

- **Python Version:** 3.11.12
- **Other Tools Used (e.g., VS Code, Jupyter, etc.):**

Code: Environment & GPU Check

```
import torch
print ("GPU available:", torch.cuda.is_available())
```

```
GPU available: True
```

---

5. LLM Setup

- **Model Name :LLaMA-2-13B**
- **Provider : Hugging Face**
- **Key Libraries & Dependencies (with versions): transformers, datasets, peft, numpy, torch, pandas, gradio**
- **Libraries and Dependencies Required:**
  (Include all relevant Python packages. Provide requirements.txt if available.)

Code: Install & Import

```
pip install google-generativeai transformers datasets accelerate peft trl
```

```
Requirement already satisfied: google-generativeai in /usr/local/lib/python3.11/dist-packages (0.8.5)
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.51.3)
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-packages (2.14.4)
Requirement already satisfied: accelerate in /usr/local/lib/python3.11/dist-packages (1.6.0)
Requirement already satisfied: peft in /usr/local/lib/python3.11/dist-packages (0.15.2)
Collecting trl
  Downloading trl-0.17.0-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: google-ai-generativelanguage==0.6.15 in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (0.6.15)
Requirement already satisfied: google-api-core in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (2.24.2)
Requirement already satisfied: google-api-python-client in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (2.169.0)
Requirement already satisfied: google-auth>=2.15.0 in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (2.38.0)
```

```
#Logging into Hugging Face using token
import torch
from transformers import (
    AutoModelForCausalLM,
    AutoTokenizer,
    TrainingArguments,
    Trainer,
    DataCollatorForLanguageModeling
)
import datasets
import peft
import numpy as np
import os
```

```
[ ]  # Load model directly
     from transformers import AutoTokenizer, AutoModelForCausalLM
     import torch
```

```
#Create the classification layer for the model
from transformers import AutoModelForCausalLM
from torch import nn
from peft import get_peft_model, LoraConfig
```

---

6. Dataset Description

- **Dataset Name & Source:**

    1. **Healthfact :** This dataset of medical misinformation was collected and is published by Kempelen Institute of Intelligent Technologies (KInIT). It consists of approx. 317k news articles and blog posts on medical topics published between January 1, 1998 and February 1, 2022 from a total of 207 reliable and unreliable sources **(**https://github.com/kinit-sk/medical-misinformation-dataset)
    2. **Scifi**: a dataset of 1.4K expert-written scientific claims paired with evidence-containing abstracts, and annotated with labels and rationales. (https://huggingface.co/datasets/allenai/scifact)
    3. **Covid-19 Fake News:** As the COVID-19 virus quickly spreads around the world, unfortunately, misinformation related to COVID-19 also gets created and spreads like wild fire. To help researchers combat COVID-19 health misinformation, this dataset created**. (https://www.kaggle.com/datasets/arashnic/covid19-fake-news)**

- **Feature Dictionary / Variable Description:**

    1. Healthfact:
        - claim_id: An id of claim (int)
        - Claim: A medical claim (text)
        - Explanation: The explanation of the claim (text)
        - Label: True/False/Misleading
    2. Scifi:
        - Claim: a medical claim (int)
        - Evidence_text: the evidence for the claim (text)
        - Label: True/False/Misleading
    3. Covid-19 Fake News
        - Id: an id of a tweet (int)
        - Tweet: a tweet about covid-19 (text)
        - Label : True/False

- **Was preprocessing done? If yes, describe:**

  My team collaborated to preprocess data so its variable structures are described as above which ensures to include both claim and label, so it is suitable for the classification problem.

Code: Load & Preprocess Dataset

```
[ ] # Load train, validation and test data of HealthFact
    import pandas as pd
    healthfact_train = pd.read_csv("/content/HealthFact_traindata_.csv")

    healthfact_test = pd.read_csv("/content/Cleaned_Test_Dataset.csv")
```

```
# prompt: get 100 random rows from healthfact_test

import pandas as pd

# Assuming healthfact_test is already loaded as a pandas DataFrame
# If not, load it first: healthfact_test = pd.read_csv("/content/Cleaned_Test_Dataset.csv")

# Get 100 random rows
healthfact_test_sample = healthfact_test.sample(n=100)

# Now 'random_rows' contains 100 randomly selected rows from healthfact_test
healthfact_test_sample.head(10)
```

	Unnamed: 0	claim_id	claim	explanation	label
541	542	26263	Facebook post Says a warning label on a box of...	The N95 respirator offers the most protection ...	false
564	565	1886	Peru's giant jungle fish hooks conscientious g...	Move over Chilean sea bass, Peruvians are rais...	true
676	678	24588	Preventive care does not save the government m...	Brooks claims that preventive care will cost t...	true
642	644	14799	Since 2004, more than 2,000 suspected terroris...	Rep. Mike Thompson said recently on the House ...	true
632	634	11057	Oregano, Thyme May Hold the Cure for Wasting S...	This release suffers from over-reach and obfus...	false
267	267	24117	The New England Journal of Medicine released a...	Bachmann claims New England Journal of Medicin...	false
925	927	5152	UN agency: Tanzania not sharing details on Ebo...	The World Health Organization has issued an un...	true
1227	1229	27134	A photograph shows an eagle catching a drone.	In nature, birds of prey often overpower large...	true
953	955	32860	Pantene shampoo and conditioner leave a build-...	In the near-decade between the first appearanc...	false
1125	1127	18040	Wendy Davis Says that each year, about 25,000 ...	Davis said, "Each year, about 25,000 American ...	MISLEADING

```
[ ] healthfact_test_sample
```

	Unnamed: 0	claim_id	claim	explanation	label
256	256	18217	As a student at Occidental College in Los Ange...	Obama said that as a student at Occidental Col...	true
682	684	6822	Chicago residents use kits to test for lead co...	Hundreds of Chicago residents have used free t...	true
1181	1183	15218	Planned Parenthood is "not actually doing wome...	Bush said, Planned Parenthood is "not actually...	false
505	506	13391	Mike Coffman Says state Sen. Morgan Carroll's ...	Coffman says Carroll's votes on state legislat...	false

Load Healthfact dataset

```
[ ] #Upload Scifi datasets
    import pandas as pd

    # Load JSON file
    scifi_train = pd.read_json("/content/train_3class.jsonl",lines=True)
    scifi_val = pd.read_json("/content/dev_3class.jsonl",lines=True)
```

```
[ ] scifi_train.head(5)
```

	claim	evidence_text	label
0	0-dimensional biomaterials lack inductive prop...		Misleading
1	1 in 5 million in UK have abnormal PrP positiv...	RESULTS Of the 32,441 appendix samples 16 were...	False
2	1-1% of colorectal cancer patients are diagnos...		Misleading
3	10% of sudden infant death syndrome (SIDS) dea...		Misleading
4	32% of liver transplantation programs required...	Policies requiring discontinuation of methadon...	True

```
[ ] # prompt: Using dataframe scifi_train: i want to get 300 samples from scif_train

    # Sample 300 rows from the scifi_train DataFrame.
    scifi_train_sample = scifi_train.sample(n=300, random_state=42) # Setting a random state ensures reproducibility

    # Display the first few rows of the sample
    #print(scifi_train_sample.head())
```

```
    # prompt: Using dataframe scifi_train: want to turn rows in column 'label' to lower case

    # Convert the values in the 'label' column to lowercase.
    scifi_train['label'] = scifi_train['label'].str.lower()
    # Convert the values in the 'label' column to lowercase.
    scifi_val['label'] = scifi_val['label'].str.lower()
```

```
[ ] scifi_train_sample
```

	claim	evidence_text	label
209	Ca2+ cycling is a UCP1-independent thermogenic...	Here we report a robust UCP1-independent therm...	true
331	Early patent ductus ateriosus (PDA) screening ...	Exposed infants had a lower hospital death rat...	true
1116	The myocardial cell lineage originally develop...	Studies in the mouse embryo and the mouse embr...	false
987	Stroke patients with prior use of direct oral ...	Compared with patients with prior use of warfa...	false

Load Scifi dataset

```
[ ] #Loading data
    import pandas as pd
    covid_train = pd.read_csv("/content/Cleaned_Covid_Train.csv")
    covid_val= pd.read_csv("/content/Cleaned_Covid_Dev.csv")
    covid_test= pd.read_csv("/content/Cleaned_Answers_To_Test_Dataset.csv")
```

```
# prompt: i want to get 100 sample random from covid_test

# Assuming covid_test is already loaded as a pandas DataFrame
# If not, load it first:
# covid_test = pd.read_csv("/content/Cleaned_Answers_To_Test_Dataset.csv")

# Get 100 random samples from covid_test
covid_test_sample = covid_test.sample(n=100, random_state=42)  # random_state for reproducibility

print(covid_test_sample.head())
```

```
        id                                              tweet  label
1532  1533  We also wanted to call your attention to the b...   True
1726  1727  There is currently not enough evidence to sugg...   True
1414  1415  The gallery like many arts institutions across...   True
2064  2065  Eight weeks into #COVID19 #CoronaVirus outbrea...   True
930    931  South Dakota has yet to issue any directives t...   True
```

```
covid_val
```

	id	tweet	label
0	1	Chinese converting to Islam after realising th...	False
1	2	11 out of 13 people (from the Diamond Princess...	False
2	3	COVID-19 Is Caused By A Bacterium, Not Virus A...	False
3	4	Mike Pence in RNC speech praises Donald Trump'...	False
4	5	6/10 Sky's @EdConwaySky explains the latest #C...	True
...	...	...	...
2135	2136	Donald Trump wrongly claimed that New Zealand ...	False
2136	2137	Current understanding is #COVID19 spreads most...	True
2137	2138	Nothing screams "I am sat around doing fuck al...	False
2138	2139	Birx says COVID-19 outbreak not under control ...	False
2139	2140	Another 4422 new coronavirus cases have been c...	True

2140 rows × 3 columns

Load Covid-19 dataset

---

7. Improving LLM Performance

Describe each improvement step (e.g., zero-shot → few-shot → tuned prompts → fine-tuning). Include **before and after** scores and your **code for each step**.

Config A: r=16, lora_alpha=32, target modules=["q_proj", "v_proj“]

Config B – hyperparamer tuning: r=32, lora_alpha =64, target modules=**[**"q_proj","k_proj","v_proj“,"o_proj"]

Step #	Method	Dataset	Accuracy (Config A)	Accuracy (Config B – hyperparameter tuning)
1	Base model (without fine-tuning)	Healthfact (100 samples)	21%	N/A
2	Model after fine-tuning 300 samples of Healthfact	Healthfact (100 samples)	48%	60%
		Scifi (100 samples)	25%	28%
3	Model after fine-tuning 300 samples of Healthfact + 300 samples of Scifi	Healthfact (100 samples)	54%	52%
		Scifi (100 samples)	60%	60%
		Covid-19 (100 samples)	63%	52%
4	Model after fine-tuning 300 samples of Healthfact + 300 samples of Scifi+ 300 samples of Covid-19	Healthfact (100 samples)	32%	46%
		Scifi (100 samples)	26%	54%
		Covid-19 (100 samples)	78%	87%

Code Snippets for Each Step

Step 1: Base model (without fine-tuning)

```python
def predict_claim(claim, tokenizer, model):
    """Predict the label for a given claim."""

    # Create the input prompt
    prompt = f""" You are a fact-checking AI.
    Classify the following claim as exactly one of these labels: "true", "false", or "misleading".

    Claim: "{claim}".
    Answer: Only respond with "true", "false", or "misleading". Do not add extra text.
    Answer:
    """

    # Tokenize the input prompt
    inputs = tokenizer(prompt, return_tensors="pt").to("cuda")

    # Generate the model's response
    outputs = model.generate(
        **inputs,
        max_new_tokens=10,  # Reduced max_new_tokens to force shorter response
        do_sample=False,  # Disabled randomness
    )

    # Extract the predicted label
    response = tokenizer.decode(outputs[0], skip_special_tokens=True).strip().lower()
    print("This is response: ", response)

    label_start_index = response.find("the claim is") + len("the claim is")
    predicted_label = response[label_start_index:].strip().split()[0].lower().replace('"', '').replace('.', '')
    print('This is the label:', predicted_label)
    # Ensure the predicted label is valid
    if predicted_label in ["true", "false", "misleading"]:
        return predicted_label
    else:
        return "unknown"  # Handle bad outputs
```

```python
from sklearn.metrics import accuracy_score
import torch
# Predict the labels for the claims in the dataset
predicted_labels = []
for claim in healthfact_test_sample["claim"]:
    predicted_label = predict_claim(claim, tokenizer, model)
    print(f"Claim: {claim}\nPredicted Label: {predicted_label}\n")
    predicted_labels.append(predicted_label)

# Evaluate the model's performance
accuracy = accuracy_score(healthfact_test_sample["label"], predicted_labels)
print(f"Accuracy: {accuracy:.4f}")
```

Step 2: Model fine-tuning for 300 samples of Healthfact

```python
# Set up LoRA config
lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

# Apply LoRA to the inner LLaMA model (before wrapping)
base_model = get_peft_model(model, lora_config)

# Wrap again with classification head
model_classify = LLaMAForSequenceClassification(base_model, num_labels)
```

```python
#Set up Training configuration
from transformers import Trainer, TrainingArguments
import os

os.environ["WANDB_DISABLED"] = "true"

training_args = TrainingArguments(
    output_dir="./llama2_lora_healthfact",
    per_device_train_batch_size=1,
    per_device_eval_batch_size=1,
    gradient_accumulation_steps=16,
    eval_strategy="epoch",
    save_strategy="no",

    logging_dir="./logs",
    logging_steps=10,
    num_train_epochs=3,
    fp16=True,
    optim="adamw_bnb_8bit",  # Optimizer for quantized training

)

trainer = Trainer(
    model=model_classify,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)
```

```
Using the `WANDB_DISABLED` environment variable is deprecated and will be removed in v5. Use the --report_to flag to control the integrations used for logging resu
```

Step 3: Model fine-tuning for 300 samples of Healthfact + 300 samples of Scifi

```python
from peft import get_peft_model, LoraConfig

lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

model_with_lora = get_peft_model(model, lora_config)

# Load previous adapter
model_with_lora.load_adapter("./llama2_lora_healthfact", adapter_name="healthfact")
model_with_lora.set_adapter("healthfact")
model_classify = LLaMAForSequenceClassification(model_with_lora, num_labels=3)
```

```
/usr/local/lib/python3.11/dist-packages/peft/mapping_func.py:73: UserWarning: You are trying to modify a model with PEFT for a second time. If you want to reload t
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/peft/tuners/tuners_utils.py:167: UserWarning: Already found a `peft_config` attribute in the model. This will lead to havir
  warnings.warn(
```

```python
# TEST MODEL FINE-TUNED ON HEALTHFACT ON SCIFI TEST DATA
import os
os.environ["WANDB_MODE"] = "disabled"

os.environ["PYTORCH_CUDA_ALLOC_CONF"] = "expandable_segments:True"

# Convert pandas DataFrames to datasets.Dataset
import datasets
from transformers import TrainingArguments
train_dataset = datasets.Dataset.from_pandas(scifi_train_sample)
val_dataset = datasets.Dataset.from_pandas(scifi_val_sample)
test_dataset_scifi = datasets.Dataset.from_pandas(scifi_test)

# Apply preprocessing using datasets.Dataset.map
train_dataset = train_dataset.map(preprocess_function, batched=True)
val_dataset = val_dataset.map(preprocess_function, batched=True)
test_dataset_scifi = test_dataset_scifi.map(preprocess_function, batched=True)

# ***REMOVE THESE LINES:***
# train_dataset = train_dataset.remove_columns(["label"])
# val_dataset = val_dataset.remove_columns(["label"])
# test_dataset_scifi =test_dataset_scifi.remove_columns(["label"])

# Remove unnecessary columns but keep 'labels' for training:
train_dataset = train_dataset.remove_columns(
    [col for col in train_dataset.column_names if col not in ['input_ids', 'attention_mask', 'labels']]
)
val_dataset = val_dataset.remove_columns(
    [col for col in val_dataset.column_names if col not in ['input_ids', 'attention_mask', 'labels']]
)
test_dataset_scifi = test_dataset_scifi.remove_columns(
    [col for col in test_dataset_scifi.column_names if col not in ['input_ids', 'attention_mask', 'labels']]
)
```

```
Map: 100% 300/300 [00:00<00:00, 6021.40 examples/s]
Map: 100% 100/100 [00:00<00:00, 3896.68 examples/s]
Map: 100% 100/100 [00:00<00:00, 3890.64 examples/s]
```

```python
from transformers import Trainer, TrainingArguments, DataCollatorWithPadding

# Setup trainer just for inference
training_args = TrainingArguments(
    output_dir="./temp_eval",
    per_device_eval_batch_size=1,
    dataloader_drop_last=False,
    report_to="none"
)

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

model_classify.classifier = model_classify.classifier.to(torch.float16)

trainer = Trainer(
    model=model_classify,
    args=training_args,
    data_collator=data_collator
)

# Run prediction
predictions = trainer.predict(test_dataset_scifi )
```

```
[30/100 00:07 < 00:17, 3.97 it/s]
```

Step 4: : Model fine-tuning for 300 samples of Healthfact + 300 samples of Scifi + + 300 samples of Covid-19

```python
from peft import get_peft_model, LoraConfig

lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

model_with_lora= get_peft_model(model, lora_config)

# Load previous adapter
model_with_lora.load_adapter("./llama2_lora_healthfact_scifi", adapter_name="healthfact_scifi")
model_with_lora.set_adapter("healthfact_scifi")
model_healthfact_scifi_classify = LLaMAForSequenceClassification(model_with_lora, num_labels=2)
```

```
/usr/local/lib/python3.11/dist-packages/peft/mapping_func.py:73: UserWarning: You are trying to modify a model with PEFT for a second time. If you want to reload t
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/peft/tuners/tuners_utils.py:167: UserWarning: Already found a `peft_config` attribute in the model. This will lead to havir
  warnings.warn(
```

---

8. Benchmarking & Evaluation

Required Components:

- **Metrics Used: Accuracy**
- **Why those metrics?**

The accuracy metric is good for the classification problem, as we want to know the percentage of true predictions of the model

- **Benchmark Dataset & Sample Size:**

I used 100 samples of Healthfact, Scifi and Covid-19 Fake News for benchmarking. After each step of fine-tuning the model on each training dataset, I tested the model on each benchmark dataset and made comparisons.

Code: Metric Calculation

Ex:

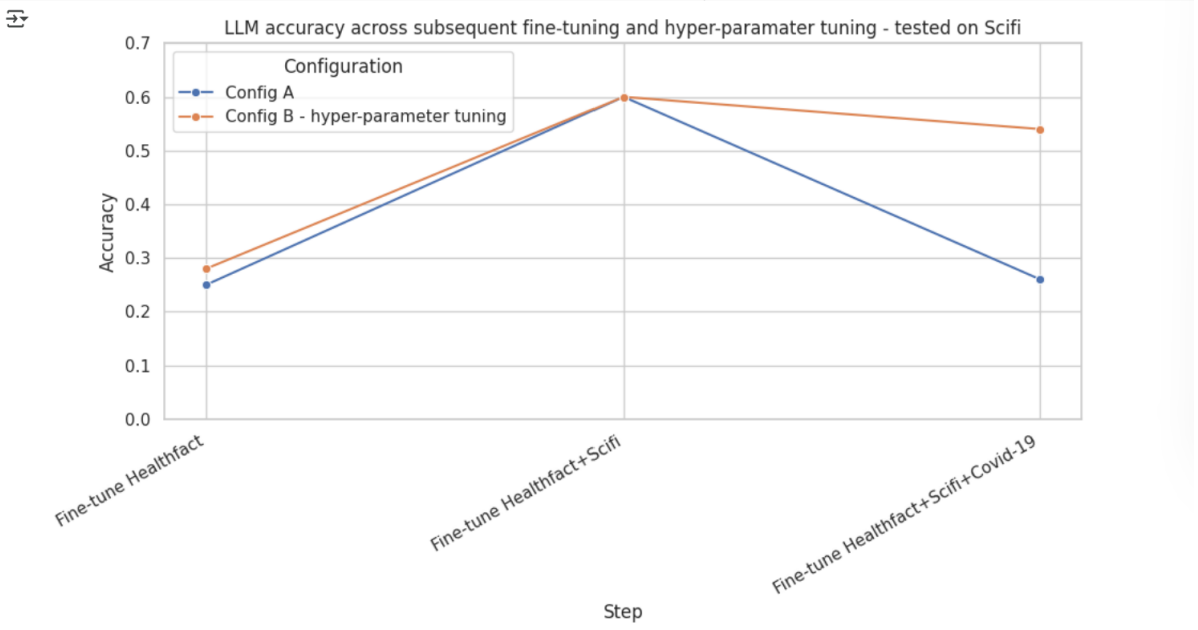
```python
import numpy as np

# STEP 1: Extract logits correctly from the prediction output
logits = predictions.predictions
if isinstance(logits, tuple):  # sometimes wrapped in a tuple
    logits = logits[0]

# STEP 2: Confirm logits is a NumPy array
logits = np.array(logits)

# STEP 3: Confirm correct shape (should be 2D: (samples, classes))
print("Logits shape:", logits.shape)  # should be (50, 3)

# STEP 4: Get predictions
y_pred = np.argmax(logits, axis=1)

# STEP 5: Get true labels
y_true = predictions.label_ids
y_true = np.array(y_true)  # just in case it's still not a proper array

# STEP 6: Evaluate
from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_true, y_pred))
print("Confusion matrix:\n", confusion_matrix(y_true, y_pred))
```

```
Logits shape: (50, 3)
              precision    recall  f1-score   support

           0       0.55      0.78      0.65        27
           1       0.42      0.28      0.33        18
           2       0.00      0.00      0.00         5

    accuracy                           0.52        50
   macro avg       0.32      0.35      0.33        50
weighted avg       0.45      0.52      0.47        50

Confusion matrix:
 [[21  6  0]
 [13  5  0]
 [ 4  1  0]]
```

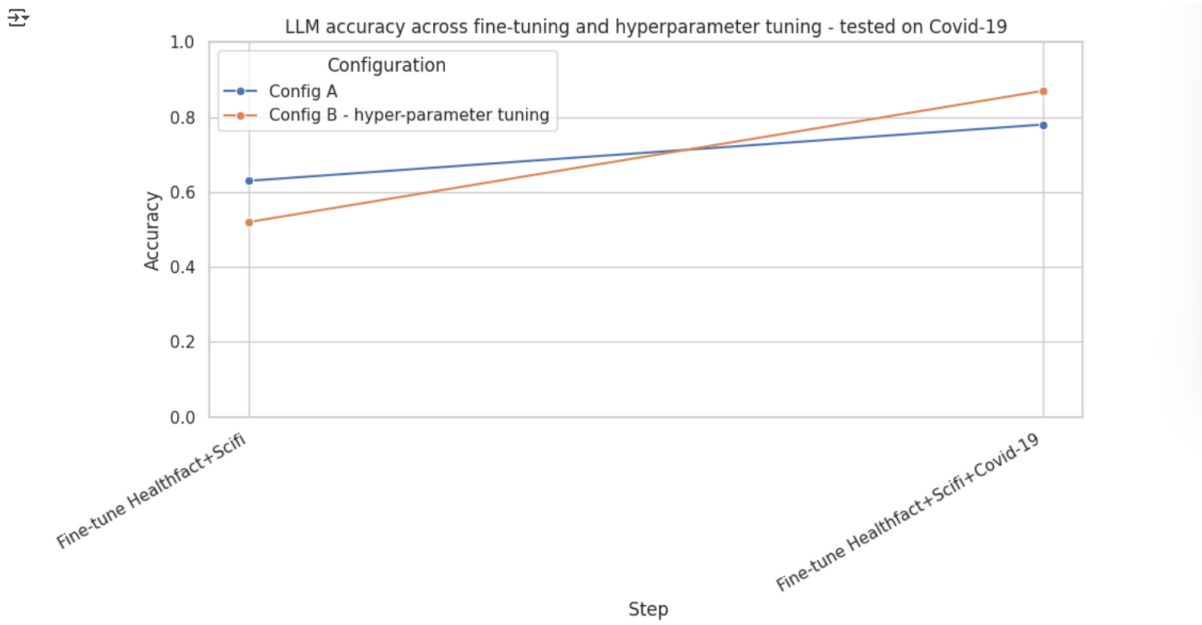
Plot Results

```python
[6] import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Data
steps= ["Base model","Fine-tune Healthfact","Fine-tune Healthfact+Scifi","Fine-tune Healthfact+Scifi+Covid-19"]
accuracy_configA=[0.21,0.48,0.54,0.32]
accuracy_configB=[0.21,0.6,0.52,0.46]

# To use seaborn easily, let's format the data into a pandas DataFrame
data = {
    'Step': steps * 2,
    'Accuracy': accuracy_configA + accuracy_configB,
    'Configuration': ['Config A'] * len(steps) + ['Config B - hyper-parameter tuning'] * len(steps)
}
df = pd.DataFrame(data)

# Set the style for seaborn plots
sns.set_theme(style="whitegrid")

# Create the plot using seaborn
plt.figure(figsize=(10, 6)) # Optional: Adjust figure size for better readability
line_plot = sns.lineplot(data=df, x="Step", y="Accuracy", hue="Configuration", marker="o")

# Increase the length of the y-axis (set limits)
# You can adjust the values (0.0 and 0.7 in this case) to fit your data range
plt.ylim(0.0, 0.7) # Set the lower and upper limits of the y-axis

# Rotate the x-axis labels
plt.xticks(rotation=30, ha='right') # Rotate labels by 30 degrees, align to the right

# Set title and labels (optional, seaborn lineplot often sets them from column names)
plt.title("LLM accuracy across subsequent fine-tuning and hyper-paramater tuning - tested on Healthfact")
plt.xlabel("Step")
plt.ylabel("Accuracy")


# Ensure layout is tight to prevent labels from overlapping
plt.tight_layout()

# Show the plot
plt.show()
```


```
[7] import matplotlib.pyplot as plt
    import seaborn as sns
    import pandas as pd

    # Data
    steps= ["Fine-tune Healthfact","Fine-tune Healthfact+Scifi","Fine-tune Healthfact+Scifi+Covid-19"]
    accuracy_configA=[0.25,0.6,0.26]
    accuracy_configB=[0.28,0.6,0.54]

    # To use seaborn easily, let's format the data into a pandas DataFrame
    data = {
        'Step': steps * 2,
        'Accuracy': accuracy_configA + accuracy_configB,
        'Configuration': ['Config A'] * len(steps) + ['Config B - hyper-parameter tuning'] * len(steps)
    }
    df = pd.DataFrame(data)

    # Set the style for seaborn plots
    sns.set_theme(style="whitegrid")

    # Create the plot using seaborn
    plt.figure(figsize=(10, 6)) # Optional: Adjust figure size for better readability
    line_plot = sns.lineplot(data=df, x="Step", y="Accuracy", hue="Configuration", marker="o")

    # Increase the length of the y-axis (set limits)
    # You can adjust the values (0.0 and 0.7 in this case) to fit your data range
    plt.ylim(0.0, 0.7) # Set the lower and upper limits of the y-axis

    # Rotate the x-axis labels
    plt.xticks(rotation=30, ha='right') # Rotate labels by 30 degrees, align to the right

    # Set title and labels (optional, seaborn lineplot often sets them from column names)
    plt.title("LLM accuracy across subsequent fine-tuning and hyper-paramater tuning - tested on Scifi")
    plt.xlabel("Step")
    plt.ylabel("Accuracy")


    # Ensure layout is tight to prevent labels from overlapping
    plt.tight_layout()

    # Show the plot
    plt.show()
```




```
[11] import matplotlib.pyplot as plt
     import seaborn as sns
     import pandas as pd

     # Data
     steps = ["Fine-tune Healthfact+Scifi", "Fine-tune Healthfact+Scifi+Covid-19"]
     accuracy_configA = [0.63, 0.78]
     accuracy_configB = [0.52, 0.87]

     # Format into a DataFrame with numeric x-axis
     data = {
         'Step ID': [0, 1] * 2,
         'Step Label': steps * 2,
         'Accuracy': accuracy_configA + accuracy_configB,
         'Configuration': ['Config A'] * 2 + ['Config B - hyper-parameter tuning'] * 2
     }
     df = pd.DataFrame(data)

     # Set seaborn style
     sns.set_theme(style="whitegrid")

     # Plot using numeric x-axis for better control
     plt.figure(figsize=(10, 6))
     line_plot = sns.lineplot(data=df, x="Step ID", y="Accuracy", hue="Configuration", marker="o")

     # Fix x-ticks and labels
     plt.xticks([0, 1], steps, rotation=30, ha='right')

     # Fix y-axis range
     plt.ylim(0.0, 1.0)

     # Add title and labels
     plt.title("LLM accuracy across fine-tuning and hyperparameter tuning - tested on Covid-19")
     plt.xlabel(" Step")
     plt.ylabel("Accuracy")

     plt.tight_layout()
     plt.show()
```



Interpretation:

The graphs show that:

- For Healthfact datasets (3 labels True/False/Misleading): after fine-tuning Healthfact, the accuracy increases significantly from the base model (without fine-tuning) from 21% to 48%. This amount improves to 54% after continuing fine-tuning Scifi. However, it degrades after fine-tuning Covid-19. It is explained by the fact that when I fine-tuned Covid-19 the classifier layer only has two labels (True/False) and when test on Healthfact with 3 labels, the accuracy may obviously reduce.

- The trend is similar for Scifi datasets. The accuracy increases significantly after fine-tuning Scifi dataset and drops fairly after fine-tuning Covid-19

- For Covid-19 Fake News , the accuracy increases from 63% to 78%.

- Overall both three datasets experience significant improvements of accuracy after subsequent fine-tuning(especially after fine-tuning the same dataset source). Except for the case when fine-tuning the Covid-19 dataset and testing again on Healthfact and Scifi, the accuracy drops due to the mismatch of the number of labels. Furthermore, the hyperparameter tuning shows a good improvement (almost increases of about 10% for each step). This shows that when we increase the rank of the low-rank decomposition, expand the parts of the model that are LoRA-injected, the model performance improves clearly.

---

9. UI Integration

- **Tool Used : Gradio**
- **Key Features of the Interface: A chatbot that returns True/False/Misleading to a claim of an user**
- **Include 2+ Screenshots of Working UI:**

Code: UI Implementation (Gradio Example)

```python
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
# Make sure the classifier is on the correct device and the input size is correct
base_model_dtype = model_healthfact_scifi_covid.model.dtype

model_healthfact_scifi_covid.classifier = model_healthfact_scifi_covid.classifier.to(base_model_dtype)
# Ensure classifier is on the correct device
model_healthfact_scifi_covid.classifier.to(device)

label_map = {0: "true", 1: "false", 2: "misleading"}

def classify_claim(claim):
    # Tokenize
    inputs = tokenizer(claim, return_tensors="pt", padding=True, truncation=True)
    inputs = {k: v.to(device) for k, v in inputs.items()}

    # Ensure model is in eval mode and on the correct device
    model_healthfact_scifi_covid.eval()
    model_healthfact_scifi_covid.to(device)


    # Get logits
    with torch.no_grad():
        outputs = model_healthfact_scifi_covid(**inputs)
        logits = outputs['logits']   # Use the last token's logits if using a classification head

    # Classification from logits
    probs = F.softmax(logits, dim=-1)
    pred_class = torch.argmax(probs, dim=-1).item()

    return label_map[pred_class]

def respond_to_user(message, chat_history):
    try:
        response = classify_claim(message)

        return response
    except Exception as e:
        error_details = traceback.format_exc()
        print("[ERROR] Exception caught:\n", error_details)  # Print full traceback to console/log
        chat_history.append((message, f"[ERROR]: {str(e)}"))
        return "", chat_history



chatbot = gr.ChatInterface(
    fn=respond_to_user,
    title="Claim Verifier (LLaMA-2)",
    description="Enter a claim and get: true / false / misleading",
    examples=["COVID-19 vaccines cause infertility", "Water boils at 100°C", "The moon is made of cheese"],
)

chatbot.launch(share=True,debug=True)
```