O-Health
Innovation for better health

O-Health NLP_ML Engineer Assessment Task

About O-Health

O-Health is an innovative digital health service provider that brings quality, accessible, and affordable healthcare to rural communities in India through it's robust and user-friendly platform that will connect communities with qualified healthcare professionals. O-Health is dedicated to revolutionizing patient care through advanced machine learning technologies, and committed to enhancing patient outcomes, streamlining clinical workflows, and empowering healthcare professionals with intelligent tools.

Introduction & Project Assessment Context

Task Introduction:

Given the complexities of our target user group (primarily rural areas speaking regional languages like Hindi/Dogri), we face challenges in acquiring datasets in suitable formats.

Challenge:

- Publicly available healthcare datasets are often incomplete, unstructured, or in a different format.
- We need custom scripts to **restructure, normalize, and potentially synthesize** missing data.

Thus, the initial assessment task involves data acquisition, preprocessing, restructuring, and then building a functional NLP pipeline addressing specific needs outlined below in the document. Your assessment task includes developing an **AI model for the Edge** with a requirement of minimum model size that addresses data preparation and symptom-extraction tasks, including severity sentiment detection and cause-reasoning. Below are the detailed tasks and required deliverables.

Edge Optimization & Model Choice

Since this AI system will run on the edge (smartphones and low-power devices), you cannot rely on large models like GPT-4 or other cloud-based LLMs that require multiple billion parameters and massive computational resources. Instead, you must develop a small, efficient model that performs well within the given constraints.

For this task, we expect you to:

1. Design a compact model that balances accuracy with efficiency, ensuring minimal memory and processing requirements.
2. Justify your architecture choice—you can choose between an RNN-based models, Transformer-based lightweight architecture
3. Explain why your chosen approach is suitable for symptom extraction, risk assessment, and reasoning
4. Optimize the model

Task 1 : Data Preparation & Normalization

Brief:
Finding publicly available datasets specifically formatted for our requirements (medical conversations, symptom intensity detection, symptom cause-reasoning) is challenging. Part of this task is to identify open datasets, restructure them into a uniform format, and develop Python scripts for data cleaning and formatting.

Necessity:
This forms the foundational step for accurate downstream ML tasks, ensuring models are trained on properly structured data.

Develop scripts (Python preferred) that take raw text data (e.g., different formats, messy structure) and convert it into a standardized format suitable for ML training/inference.

Possible transformations:

- Removing duplicates, normalizing cases, handling punctuation.
- Splitting transcripts into sentences or token-level structure.

Where real data is insufficient, propose methods to synthetically generate symptom text—e.g., controlled random insertion of synonyms, severity terms, cause phrases.

Outline a brief validation approach to ensure synthetic data does not deviate too far from real-world usage.

Deliverables:

- Dataset for training the model in required structure for the next tasks.
- Python script (Jupyter Notebook/colab notebook) to identify suitable datasets with the script to clean, preprocess, and normalize the dataset (converting various formats to a standardized CSV/JSON structure)

Task 2: Symptom Extraction Model

Brief:

- Propose and implement **any suitable NLP approach** for extracting symptoms from the normalized text. The model must intelligently handle multiple variants of symptom descriptions. In an input text (patient-doctor conversation) the symptoms must be accurately extracted from the input.
- Synonymous phrases and various Lexical variations must be considered.
  (e.g., "chest pain," "pain in chest," "aching chest" → unified to "chest pain").
- Healthcare text often includes negations that must be excluded from the final symptom list. For example: "He does not have a fever but has a mild headache."
  - Extracted symptom: "mild headache"
  - Fever should NOT be extracted because it's explicitly negated.

Questions to Answer:

- What is your chosen approach and why?
- How do you ensure accurate symptom normalization?
- Discuss how you would scale to more symptoms (e.g., if you add 100 new symptoms)

Deliverables:

- Provide a working prototype in a shared environment (e.g., Jupyter Notebook).
- Benchmark metrics (Accuracy, Precision, Recall).
- Estimate memory footprint (model size in MB) when running inference.

Task 3 : Severity & Sentiment Analysis

Brief:

Extend the symptom-extraction model to identify severity or sentiment cues (e.g., "severe," "mild," "moderate," "really bad") and assign a risk score or category (e.g., low, moderate, high).

We must combine multiple factors when determining the patient's risk category:

1. **Symptom Duration**
   - Short-term (e.g., 1–2 days, up to 1 week): Typically low or moderate risk, *unless* it involves critical symptoms (chest pain, severe bleeding, etc.).
   - Long-term (e.g., >2 weeks): If accompanied by significant symptoms (shortness of breath, persistent pain, chest pressure), it often escalates to high risk.
2. **Symptom Severity**
   - Mild/Moderate: Usually lower risk unless chronic.
   - Severe/Critical: Usually high risk, *especially* if persistent.

Finding the right blend of parameters and their intensity to determine the risk score is essential. More parameters to assess the risk score are welcome.

Examples

- ○ "I have had a chest pain for 3 weeks and difficulty breathing." → **High risk** (long duration + severe symptoms).
- ○ "I developed a slight rash on my arm for 2 days." → **Low or moderate risk** (short duration + mild symptom).

Deliverables:

- Demonstrate the risk categorization logic in your Jupyter Notebook.
- Provide example metrics (accuracy in detecting severity, confusion matrix if applicable).

Attached is an excel document containing sample patient inputs of varying lengths (short, medium, long). Each input simulates a patient's spoken/written description. After each input, we show the expected output the model should extract.

(Optional) Task 4 : Reasoning/Root Cause Extraction

Extended Model Functionality

Beyond just symptoms and severity, **identify possible cause or reason**.

Example:

- **Input:**
  "My lower back started aching right after I lifted a heavy box last week. I guess I twisted something."
- **Expected Output:**
  Symptoms: "lower back ache"
  Cause: "lifting a heavy box"

Deliverables:

- ○ Extend the existing pipeline to detect cause phrases in free-text.

Task 5 : Practical Assessment Questions

Context:

We would require a **speech-to-text (STT)** model specifically tuned for **Dogri**—a regional Indian language with similarities to Hindi. This model will be **embedded on smartphones** (edge deployment). Your task would be answering your approach on how you would develop such a model to run entirely offline on a smartphone, performing well under low computational constraints.

Question 1: Model Selection & Data Preparation

- **How would you approach designing a robust STT model specifically for Dogri?**
    - What existing pre-trained models or toolkits (e.g., Whisper, AI4Bharat, Vosk, Bhashini, Kaldi, etc.) would you initially consider and why?
    - Given Dogri's linguistic similarity to Hindi, how would you leverage Hindi datasets/models to bootstrap your Dogri model?

Question 2: Steps for Model Development (Provide a detailed step-wise approach)

- **Data Collection & Preparation:**
    - How would you source and collect Dogri speech datasets? If publicly available data is limited, how would you generate synthetic or semi-synthetic data?
    - Outline the steps clearly: e.g., data gathering → audio cleaning/noise filtering → transcription and validation → dataset normalization.

Question 3: Data Diversity

- How would you ensure your collected/synthetic dataset adequately represents variations in dialect, pronunciation, accents, background noise, and speaker diversity?

Question 4: Model Training & Evaluation Strategy

- Briefly describe the training approach you would adopt to fine-tune a speech recognition model on your Dogri dataset. What architecture would you use (Transformers, CNN-RNN hybrids, etc.) and why?

Question 5 : Training & GPU Requirements:

Training a speech-to-text model, especially multilingual fine-tuning, is computationally demanding. Based on your experience or knowledge answer the following:

- **What hardware (GPU, RAM, CPU) and infrastructure would you require to train your Dogri STT model effectively?**
- Clearly specify GPU type (4090/H100/A6000), approximate GPU hours or epochs, and if any special hardware or cloud infrastructure would be necessary.
- For running inference on a smartphone, state clearly the approximate **RAM and battery impact** you would anticipate. Suggest an inference framework (e.g., TFLite, PyTorch Mobile, ONNX Runtime) to deploy on edge and justify your choice.

Candidate Requirements & Final Deliverables

1. **Working Prototype (Jupyter Notebooks) with 1 or multiple scripts containing:**

    - **Task 1**: Data scripts for restructuring/normalizing any found dataset (plus potential synthetic data generation).
    - **Task 2**: Symptom extraction model with accuracy and memory footprint measurements.
    - **Task 3**: Extended model to detect severity or sentiment, assigning risk categories.
    - **Task 4**: Additional logic for extracting root cause if present.

2. **Documentation**
    Google Doc/PDF with:
        - Steps of how data was prepared.
        - Model architecture(s), libraries, techniques used (rule-based, NER, ML, etc.).
        - Why did you choose a particular NLP architecture?
        - **Scaling strategy** for additional symptoms
        - Observations on **accuracy** and **memory usage** for each model/task.
        - Edge efficiency strategies: How would you reduce computational overhead while maintaining accuracy?
        - Task 5 Answers in own words

3. **Shared Access**
    - Provide a **Google Colab link** or a **GitHub repository** with Jupyter Notebooks so we can run and test locally.

Next Steps

- **Implement** each task in phases, beginning with data preparation.
- **Test** with the provided example inputs to verify your model's correctness.
- **Iterate** on your approach if certain symptom synonyms or severity terms are missed.
- **Optimize** memory usage where possible, as this eventually needs to run on-edge.
- Completing the task at the earliest would be essential as the task completion time would be a main factor for evaluation.
- If any clarification is needed on the tasks or data specifics, do not hesitate to ask.