CS 4063 - Natural Language Processing

Due Date: Friday, February 20th by 11:55pm.

Assignments are to be done in groups of **two or three**. No late assignments will be accepted.
Submissions that do not comply with the specifications given in this document will not be marked and a zero grade will be assigned.

You are allowed to use all the generative tools that you have access to. But you should generate the minimal possible code to implement the reqired functionality. You should also be able to explain the code and produce the prompts when asked. **Be careful, this is not a one shot assignment and requires multiple pieces to be executed separately before it all come together. Learn!**

Each group must submit a **single GitHub repository link** on Google Classroom, containing all code, documentation, models, deployment scripts, and test cases. Also submit the vercel link that you have deployed your app on. **Do not submit your code, models, docker files on Google Classroom**. A short loom video to demo your system is optional but recommended.

Urdu Children's Story Generation System

1 Introduction

In the pre-LLM era, language generation relied heavily on probabilistic modeling techniques such as n-gram models and corpus engineering. In this assignment, you will bridge the gap between classical foundations and modern software engineering by integrating state-of-the-art coding practices, microservices, containerization, and automated deployment.
Your task is to implement and deploy a **genAI system** that generates short Urdu stories for children. This involves data acquisition, preprocessing, subword tokenization, inference serving, and a web-based UI.

2 Overall Objective

Develop a fully functional **Urdu Story Generation AI App** that can be broken down into the following phases:

- Scrapes and processes real-world Urdu text.
- Trains a custom **Byte Pair Encoding (BPE)** tokenizer.
- Implements a **Tri-gram** probabilistic language model.
- Serves predictions via a containerized microservice.
- Provides a ChatGPT-like interface deployed on **Vercel**.

3 Phase-wise Assignment Tasks

3.1 Phase I: Dataset Collection and Preprocessing

1. Scrape a diverse corpus of Urdu stories (min. 200 stories).
2. **Preprocessing:** Remove HTML/ads, other language characters, normalize Unicode, and standardize punctuation.
3. **Special Tokens:** Add un-used unicode bytes for special tokens `<EOS>` (Sentence), `<EOP>` (Paragraph), and `<EOT>` (Story). These will be used for the tokenizer

3.2 Phase II: Tokenizer Training (BPE)

Train a BPE tokenizer on the training dataset that you have scraped. The tokenizer must handle the Urdu script efficiently, reducing vocabulary size while preserving subword semantics. The vocabulary size hyperparameter should be set to 250. Do not use pre-built tokenizers from any of the python libraries.

3.3 Phase III: Trigram Language Model

Train a **3-gram model** using Maximum Likelihood Estimation (MLE).

- **Interpolation:** Implement interpolation technique as discussed in class.
- **Generation:** Support variable-length generation until the `<EOT>` token is reached.

3.4 Phase IV: Microservice & Containerization

Expose the model for inference via a **FastAPI** (REST or gRPC) service.

- **Endpoint:** `POST /generate` (Input: prefix string, max-length).
- **DevOps:** Provide a `Dockerfile` and a GitHub Actions CI/CD pipeline.

3.5 Phase V: Web-based Frontend

Create a reactive UI (React, Next.js, or Vue) that:

- Accepts a starting phrase in Urdu.
- Displays streaming or step-wise story completion (just like chatGPT).

3.6 Phase VI: Cloud Deployment

Deploy the frontend on **Vercel**. The backend may be hosted as a Vercel Serverless Function or on a public container platform (e.g., Render or Railway) linked to the frontend.

Honor Policy

Cheating in code, datasets, or reports will result in an immediate **zero grade** and reporting to the academic disciplinary committee. All work must be original to the group.