CS6320E - Topics in Natural Language Processing

Assignment: Compact Language Model Challenge

Participants (individuals or teams) will build compact, sample-efficient language models that perform next-token prediction (causal language modelling) for **Telugu** and **Marathi**. Additional languages will be added one week after the assignment is released. Teams must design models that perform well with limited labeled data; we will evaluate submissions on a held-out test set.

Learning Objectives

By completing this assignment students will learn to:

- Design and implement compact language models suitable for low-resource languages.
- Measure and report both predictive performance and computational efficiency (parameters, latency, FLOPs, memory).
- Practice reproducible ML workflows: clear READMEs, scripts, and reporting of compute and random seeds.
- Compare and justify modeling choices under strict resource budgets.

Dataset & Splits (what we provide)

- We will provide the **train** and **validation** splits for each supported language at dataset release. Exact file format and schema consist many field, you should only extract text from two field i.e. “input” and “target”. You can use this text to train your model.

  E.g.: "input":"ీ రవరి 23, 2019 174 ఎన్నో భారీ అంచనాలతో వచ్చిన మహానుభావుడు ఎన్. టి. ఆర్ బయోపిక్ రెండు పార్టులు నిరాశపరచాయి.","target":"మహానాయకుడు ముంచేశాడుగా"

- A held-out **test** split will not be released to students. We will evaluate final submissions on this test split.
- **Important:** Students may only use the provided training split for supervised training. Use of any additional labeled data for supervised training is prohibited unless explicitly allowed by us.
- Use the following link to download the dataset- https://drive.google.com/drive/folders/1d2-Uf2yuCoJ2QLDd03pC-WyBR55JSE0-?usp=drive_link

Teams, Collaboration & Academic Integrity

- Teams: work may be submitted individually or in teams of up to **2** students. Each team must declare members at submission.

- Collaboration policy: Students may discuss high-level ideas with each other, but **code, model weights, and writeups must be original** to each team. Copying other students' code or reusing another team's submission without explicit permission is academic misconduct.

- Plagiarism: any submission found to use another group's work without attribution will receive a failing grade and be reported according to university policies.

Task & Evaluation Metrics

Task: next-token prediction (causal LM). Models should output a probability distribution for the next token given previous context.

Primary evaluation metric: Perplexity on the held-out test set (lower is better).

Required additional metrics to report:

- Cross-entropy / bits-per-token.

- Token-level top-1 accuracy.

- **Parameter counts:** total and trainable parameters (include embeddings), and exact method used to compute them.

- **Inference latency:** median and p95 time per token on the evaluation hardware

- **Throughput:** tokens/second under greedy decoding.

- **FLOPs / MACs estimate:** method used for estimation.

- **Quantized model size** (file size after INT8 or other quantization), if applicable.

- **Peak GPU/CPU memory usage** during inference.

- **Sample-efficiency curve:** validation performance vs. number of labeled examples used

Ranking for the assignment grade: we will consider test-set perplexity and the efficiency metrics when assigning grades. See the rubric below.

Grading Rubric (suggested; adapt as needed for course credit)

Total: **100 points**

- **Model performance (perplexity on test)** — 40 points (primary).

- **Efficiency & resource reporting** — 20 points (parameters, latency, throughput, FLOPs, memory). Correct measurement methodology is required.

- **Technical report (clarity & reproducibility)** — 20 points (concise 4–6 page PDF explaining architecture, training recipe, hyperparameters, preprocessing, and results; must include figures and tables).

- **Code quality & reproducibility** — 10 points (working inference script, README, environment spec, scripts to reproduce key results).

- **Viva** — 10 points

Deliverables (what to submit)

Submit a single Git repository or ZIP file with the following structure and files:

1. model/ — final model checkpoint(s) and tokenizer files (or link to an artifact store if files are too large). Include quantized artifacts if used.

2. infer.py — inference script that takes test inputs and writes next-token probabilities. This script must run on the standard evaluation environment (specs provided) and accept command-line args for batch size, device, and seed.

3. measure_performance.py — script or documented commands that compute parameter counts, inference latency (median & p95), throughput, peak memory, and FLOPs (or provide the profiling scripts used).

4. README.md — clear reproduction instructions (how to set up environment, commands to run inference and measurements, random seeds, hardware used).

5. report.pdf — 4–6 pages describing model architecture, training procedure, hyperparameters, preprocessing, external resources, results (include tables/plots), and ethical/license discussion.

6. requirements.txt or environment.yml — environment spec.

7. submission_metadata.json — JSON file listing team members and external resources.

Naming convention: COURSEID_ASSIGNMENT_COMPACTLM_<teamname>.zip or provide the Git repo URL.

Allowed & Disallowed Resources (course policy)

Allowed (default):

- Public open-source pretrained checkpoints (e.g., Hugging Face models under permissive licenses).

- Tokenizers and standard toolkits: Hugging Face Transformers, SentencePiece, KenLM, Fairseq, OpenNMT, etc.

- Standard libraries: NumPy, PyTorch, TensorFlow, JAX, Scikit-learn.

Disallowed:

- Fine-tuning or supervised training using proprietary closed-source LLMs accessed via APIs.

- Use of extra labeled datasets beyond the provided train split for supervised training.
- Leaking test data into training/validation.

Mandatory declarations: All external resources (model checkpoints, corpora, tokenizers, preprocessing scripts, libraries, seeds, compute/hardware) must be declared in submission_metadata.json, the README, and the report.

Submission & Late Policy

- Submission method: upload to the course LMS (or provide Git repo URL) before the deadline. Replace placeholders below with real dates.
    - **Deadline:** 3/11/2025 .
    - **Late policy:** Late submissions accepted up to 48 hours with a penalty (e.g., −10% per 24h window)
- No submissions will be accepted after 7 days without our approval.

Evaluation Process & Reproducibility Checks

- We will run infer.py and measure_performance.py on a standard evaluation VM to verify reported numbers.
- Provide Dockerfile or container image upon request for reproducibility.

Academic & Ethical Considerations

- Respect licenses for any external pretrained checkpoints and code. Do not redistribute datasets or checkpoints if their license forbids it.
- Discuss potential biases, harms, and limitations of your model in the report.