

Assignment 3: Cross-Lingual Relation Extraction with Decoder-Only Language Models

COL772

1 Goal

This goal of the assignment is to perform **relation extraction (RE)** using decoder-only language models in multilingual setting. You will implement three approaches: a discriminative classifier, an autoregressive generation model, and in-context learning. The core task is to teach the model to perform well on the given task in multi-lingual setting with little supervised data in these languages. Your models will be evaluated on four Indian languages.

The document specifies *what* you must build and *how it will be evaluated*; the design decisions are yours. Suggestions for things to try are provided in each section, but they are not requirements. Starter code is available [HERE](#)

2 Task Definition

Given a sentence containing one or more named entity pairs, predict from a fixed ontology the **relation label** for each queried pair. If no relation holds, predict **NA**. There is exactly one correct relationship for one pair of entities in the dataset.

Consider the following example:

```
{
  "sentText": "... Mr. Wagoner started talks ... with Carlos Ghosn ,
               the chief executive of Renault and Nissan .",
  "relationMentions": [
    {
      "em1Text": "Carlos Ghosn",
      "em2Text": "Renault",
      "label": "/business/person/company"
    }
  ],
  "sentId": "1"
}
```

Each data point is a **sentence** and a list of (**em1**, **em2**, **label**) tuples, **em1** and **em2** refer to the entity in the given sentence. A sentence may yield multiple data points if it contains multiple entity pairs. At test time the **label** field is withheld.

For each language the JSON keys remain in english, while the value of the fields is given in the respective language

3 Data

3.1 English — Labeled Training Corpus

The labelled data is constructed from New York Times articles with sentence, entity pairs, and relationships as explained in the section above. with sentence-level relation labels. You can find the complete dataset in the starter code.

```
labelled_corpus_en.zip
```

The dataset is a JSONL file, each entry in the file has a `sentText` field which contains the raw natural language sentence from which relationships are to be identified. The `relationMentions` field captures the semantic relationships expressed in the sentence as a list of structured annotations. Each relation mention specifies two entity mentions (`em1Text` and `em2Text`) along with a predefined relation label that characterizes the relationship between them. A single Jsonl entry is described below:

```
+-- sentText
\-- relationMentions
    +-- relation 1
      | +-- em1Text
      | +-- em2Text
      | \-- label
    +-- relation 2
      | +-- em1Text
      | +-- em2Text
      | \-- label
    \-- ...
```

3.2 Indian Language Task Datasets (Labelled)

Small labelled relation extraction datasets in the same format as NYT-10 are provided for all four Indian languages:

- **Hindi** (`train_hi.jsonl`) — a small set(200) of labelled (sentence, entity pair, relation) instances in Hindi.
- **Kannada** (`train_kn.jsonl`) — a small set(200) of labelled instances in Kannada.
- **Tulu** (`train_tcy.jsonl`) — a small set(100) of labelled instances in Tulu.
- **Oriya** (`train_or.jsonl`) — a small set(100) of labelled instances in Oriya.

3.3 Indian Language Pre-training Corpora (Unlabelled)

Large unlabelled monolingual text corpora are provided. You can access these test corpora from huggingface using the following code snippet.

Listing 1: Loading Wikimedia Wikipedia dataset

```
from datasets import load_dataset

ds = load_dataset("wikimedia/wikipedia", "20231101.or")
```

You can use the following codes to get the data for different languages.

- **Hindi** — hi
- **Kannada** — kn
- **Tulu** — tcy
- **Oriya** — or

This corpus, sourced from Wikipedia has no relational labels and is intended to "teach" the language to the models. You are not expected to utilize the entire dataset for training.

3.4 Test Languages

Models are evaluated on four Indian-language test sets:

Language	Code	Labelled task data?	Unlabelled corpus?	Script
Hindi	hi	Yes	Yes	Devanagari
Kannada	kn	Yes	Yes	Kannada
Oriya	or	Yes(Small)	Yes(Small)	Oriya
Tulu	tcy	Yes (Small)	Yes(Small)	Kannada

Table 1: Oriya and Tulu have no data of any kind and are completely unseen during training and adaptation.

4 Evaluation Metrics

All three tasks are evaluated on the same two metrics, computed at the relation-mention level. Evaluation code is provided in `eval.py`.

4.1 Micro-F1

$$\text{Micro-F1} = \frac{2 \cdot P_\mu \cdot R_\mu}{P_\mu + R_\mu}, \quad P_\mu = \frac{\sum_r \text{TP}_r}{\sum_r (\text{TP}_r + \text{FP}_r)}, \quad R_\mu = \frac{\sum_r \text{TP}_r}{\sum_r (\text{TP}_r + \text{FN}_r)}$$

TP, FP, and FN counts are pooled across all relation classes before computing precision and recall. Micro-F1 reflects overall accuracy and is dominated by frequent relation types.

4.2 Macro-F1

$$\text{Macro-F1} = \frac{1}{|R|} \sum_{r \in R} F1_r, \quad F1_r = \frac{2 \cdot P_r \cdot R_r}{P_r + R_r}$$

F1 is computed per relation class and averaged without frequency weighting. Macro-F1 captures performance on rare relation types. $|R|$ excludes NA.

5 Tasks

5.1 Task 1: Relation Extraction with a Classification Head [35 marks]

What is required. Train a decoder-only language model to perform relation extraction by attaching a **linear classification head** to the model’s hidden states. You must use **LoRA adapters** — the base model weights must remain frozen.

Your model must save a valid JSONL that can be parsed to extract predictions in the output format of Section 6.2.

Evaluated on. English, Hindi, Kannada NYT-10 test set.

Things you might consider:

- Which hidden state to pool for the classification input — the final token, the entity boundary tokens, layerwise pooling, attention or some combination?
- How to mark entities in the input sentence (e.g. with special tokens before feeding the sentence to the model)?
- How to handle multiple entity pairs in a single sentence?
- LoRA rank, target modules, and learning rate.
- How to handle the class imbalance between labelled relations?

5.2 Task 2: Relation Extraction via Autoregressive Generation [35 marks]

What is required. Train a decoder-only language model to perform relation extraction as a **natural language generation** task. Your pipeline should receive the sentence and one or more entity pair queries and must produce the answers in a structured JSON format. You are free to pre-process the input to suit the needs of the Language model.

Your model must save a valid JSONL that can be parsed to extract predictions in the output format of Section 6.2.

Evaluated on. English, Hindi, Kannada, Tulu, Oriya NYT-10 test set.

Things you might consider:

- How to use the Hindi and Kannada pre-training corpora to adapt the model before or after fine-tuning on English.
- Post-processing to enforce valid ontology labels.

5.3 Task 3: In-Context Learning

[30 marks]

What is required. Using a larger model(Meta-Llama-3.1-8B-Instruct) as your base, improve cross-lingual relation extraction on the four Indian-language test sets using **in-context learning** — without any additional gradient updates. You must use the same input/output format as Task 2. Beyond this, all design decisions are yours. We will provide the environment and script that runs efficient LLM inference through vllm out-of-the box.

Evaluated on. All four Indian-language test sets (Hindi, Kannada, Oriya, Tulu) of NYT-10.

Things you might consider:

- How many demonstrations to include and how to select them — random sampling, stratified by relation type, or retrieved by similarity(FAISS available in environment) to the test instance.
- Whether similarity-based retrieval (e.g. using sentence embeddings) improves over random selection(This is similar to how you retrieved author’s text in Assignment 1).
- How the number of demonstrations interacts with the model’s context window limit.
- Whether ICL helps more for Oriya and Tulu (no pre-training) than for Hindi and Kannada.

6 Input and Output Format

6.1 Input Format

Your final system should take as input data that is provided as JSON Lines (.jsonl) files — one object per line:

```
{
  "articleId": "id",
  "sentId": "sentId",
  "sentText": "<sentence>",
  "relationMentions": [
    {
      "em1Text": "<e1>",
      "em2Text": "<e2>"
    },
    {
      "em1Text": "<e1>",
      "em2Text": "<e3>"
    }
  ]
}
```

At test time the `label` field inside `relationMentions` is withheld. All other fields are provided. The Json keys would be in english. Sentence, Entity and relation labels would be provided in the respective languages, a mapping for relation labels between english and indic languages are provided.

6.2 Output Format

All three tasks share the same output format. Predictions must be saved as a JSON Lines (.jsonl) file with one line per sentence **in the same order as input JSONL**. The Json keys should be in english. Entity and relation labels must be in the respective languages, a mapping for relation labels between english and indic languages are provided. Output Schema:

```
{
  "articleId": "id",
  "sentId": "sentId",
  "sentText": "<sentence>",
  "relationMentions": [
    {
      "em1Text": "<e1>",
      "em2Text": "<e2>",
      "label": "label"
    },
    {
      "em1Text": "<e1>",
      "em2Text": "<e3>",
      "label": "label"
    }
  ]
}
```

Predicted labels must exactly match with strings in the training ontology.

7 Code Standard & Constraints

7.1 Models

- Qwen/Qwen2.5-1.5B for Q1 and Q2
- Meta-Llama-3.1-8B-Instruct for Q3

Python 3.12 will be used to test the code for Q1 and Q2.

For Q3, /scratch/cse/phd/csz208845/envs/vllm_server/ will be used to test your code.

7.2 Allowed Libraries

- PyTorch, HuggingFace Transformers, HuggingFace PEFT, vllm, FAISS.
- NumPy, scikit-learn, and any standard Python library included in the default installation/provided environment.
- Any additional packages must be requested over Piazza and are permitted only after verification.

7.3 Not Allowed

- Using Pre-trained Relation-Extraction models is not permitted.

- You are only allowed to use the given data.
- Translating prompts at inference time is not allowed.

7.4 Execution Commands

Training

For each question, executing `./train.sh`, from inside the folder `Q<#>` trains all components of your system for that question. All model weights and auxiliary files must be saved to the `Q<#>/output/`.

```
./train.sh <output_dir> # Task 1
```

Inference

The following command should load model weights from `<output_dir>` and save

"output_<en/hi/kn/or/tcy>.jsonl"

for each applicable language code.

```
./infer.sh <en/hi/kn/or/tcy> <test_file_path> <output_dir> # Task 1
```

Each command loads the relevant trained model, processes the test file, and writes a prediction JSONL to the output directory.

Evaluation

You can use `eval.py` to evaluate your generations.

```
python eval.py <file_path> <reference_jsonl> # Task 1
```

7.5 Time Limits

Phase	Limit
Training (all tasks combined)	480 minutes
Inference per test file(500 Samples)	30 minutes

Scripts that exceed these limits will be terminated.

8 What to Submit?

8.1 Submission Format

Submit a zip file named `<entry_number>.zip` through Moodle. Upon unzipping it should create a directory containing the scripts along with your code files:

- Q1
 - train.sh

- infer.sh
- Q2
 - train.sh
 - infer.sh
- Q3
 - infer.sh

You do not need to upload the data that is provided however any datapre-processing/augmenting scripts need to be submitted.

8.2 Writeup Format

The `readme.md` should be 2–4 pages and must include:

1. **Honor Code.** “Even though I have taken help from the following students and LLMs in terms of discussing ideas and coding practices, all my code is written by me.” List all students and LLMs you discussed with, or write “None”.
2. **Approach.** For each task, describe what you built and the key design choices you made. You do not need to justify every decision, but the writeup should make it clear what you tried and why.

9 Environment Creation Guidelines on V100 for Q1 and Q2

- `conda create -n skylake_env_2 python=3.12`
- `pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url https://download.pytorch.org/whl/cu124`
- `pip install transformers==4.57.1 peft==0.18.1`
- `pip install --only-binary=:all: pyarrow==20.0.0 datasets`
- `pip install --only-binary=:all: scikit-learn`
- `export WANDB_DISABLED="true"` (optional)

10 Submission Guidelines

- The assignment is to be done individually.
- Your code will be tested on Skylake Node with V100 GPU.
- Do not discuss this assignment with anyone outside the class.
- Plagiarism detection software will be run. Violations carry penalties per IIT rules.

- Submissions not conforming to the output format receive a minimum 20% penalty.
- You may use LLMs for debugging assistance only. List all LLMs used in your writeup.

11 Code Verification Before Submission

Before submitting, verify that:

- Your code runs end-to-end on the provided sample data without errors.
- Prediction files are generated in the correct location and format.
- All required files are present in the zip archive.

Details of the verification procedure will be shared on Piazza.

12 Deadline

The assignment deadline is 10th April. Late submissions: 10% penalty per day, up to one week.

13 Clarifications

Post all questions to the Piazza forum. It is your responsibility to seek clarification before the deadline.