CSE556 NLP, Winter 2026
Assignment 2

Total Marks: 100 **Due:** 19 February 2026, 11:59 PM

Instructions

1. Please see the course policy on extension days and bonus points. There will be no deviation from the policy for any late submissions.
2. You may use LLMs, but you must provide the final prompt you used in your submission.
3. Wherever required, each training run should conclude within 15 minutes on your own device or in an online runtime.
4. Only one member of the group should submit the assignment as a ZIP file on Classroom.
5. All members of the group will be expected to be familiar with all parts of the assignment for the demo.
6. Doubts will be resolved only through the Google Form shared on Classroom.
7. All questions are of equal marks.

Task

In this assignment, you will build models for **Named Entity Recognition (NER)** to identify location mentions in sentences. The task is formulated as sequence labeling, where each token is assigned one label from the BIO tag set.

Dataset

You will be provided a dataset with 2 files `train_data.jsonl` and `val_data.jsonl`.Each line is a JSON object with:

- **id**: sentence identifier
- **tokens**: list of tokens in the sentence
- **labels**: list of gold BIO labels (B-LOC, I-LOC, O)

The dataset already contains gold BIO tags. Your goal is to train deep learning models that predict these tags for each token. A sample annotated sentence from the dataset is shown below:

Accident	Azadpur	Delhi	Mai	Hua	Hai
O	B-LOC	I-LOC	O	O	O

Q1. Long Short-Term Memory (LSTM)

In this question, you will implement an LSTM-based sequence tagger for BIO prediction and analyze the effect of (i) different pretrained word embeddings and (ii) different numbers of stacked LSTM layers on performance. You may use a uni-directional LSTM or a BiLSTM, clearly state your choice.

You must train the model using two different pretrained word embeddings:

- GloVe
- fastText

For each embedding choice, you must experiment with different LSTM depths by training models with the number of stacked LSTM layers :

$$L \in \{1, 2, 3\} \quad \text{(or more if you wish, but at least three settings must be compared).}$$

This results in a total of **6 model variants** (2 embeddings $\times$ 3 layer settings).

You must report the hyperparameters used for training (e.g., hidden size, dropout, optimizer, learning rate, batch size, number of epochs, etc.) and include clear training and validation plots (loss curves and/or metric curves) for your experiments.

The models must be evaluated using the following metrics: FreeMatch-F1 and Strict EM (Exact Match) as defined in class. If you apply padding, padded positions must be excluded from all evaluation metrics. Summarize your results in a table (example format shown below) and provide an in-depth analysis in your report discussing how embedding choice and LSTM depth affect performance, and how/why the trends differ between FreeMatch-F1 and Strict EM.

Embedding	Layers (L)	FreeMatch-F1	Strict EM
GloVe	1		
GloVe	2		
GloVe	3		
fastText	1		
fastText	2		
fastText	3		

IMPORTANT (Output file for grading).

In your code, you must implement a function that reads inputs from a file named `test_data.jsonl` in the same format as the provided dataset but without the `labels` field. For each sample, the function must run your best-trained model, add a `labels` field containing the predicted BIO tags, and write the results to `output.jsonl`. Please ensure that the generated `output.jsonl` matches the provided annotated dataset format exactly; otherwise we will not be able to grade your assignment.

Deliverables (Q1).

1. **Implementation:** Provide your complete implementation for preprocessing, model training, evaluation, and inference in a single file named `part1.py` or `part1.ipynb`.

2. **Best-performing model:** Submit the saved checkpoint/weights of your best-performing configuration (clearly specify which embedding and number of layers it corresponds to). Name the file in a self-explanatory manner (e.g., `best_model_fasttext_L3.pt`) and ensure it can be loaded by your `part1.py/part1.ipynb` for inference.

3. **Detailed report:** Make sure that the report includes:

   (a) a short description of your final model architecture, training setup , any kind of preprocessing steps and full list of hyperparameters used.

   (b) training and validation plots (loss curves and/or metric curves) for all the experiments

   (c) a complete results table reporting FreeMatch-F1 and Strict EM for all 6 variants, also attach the screenshot of the EM and FreeMatch-F1 score for the best performing model.

   (d) an in-depth analysis discussing the effect of embedding choice and LSTM depth, and explaining any differences in trends between FreeMatch-F1 and Strict EM.

Q2. BiLSTM + CRF

A Conditional Random Field (CRF) is a probabilistic model for structured prediction that assigns labels to an entire sequence jointly, instead of labeling each token independently. In this question, you will train a BiLSTM + CRF model for BIO sequence labeling, where the BiLSTM encodes the input token sequence and the CRF produces the final predicted BIO tag sequence. You may use an existing CRF library for the CRF layer/decoding (e.g., `torchcrf`) or implement the CRF yourself.

You must choose **any one** pretrained embedding of your choice (GloVe / fastText) and clearly mention your choice in the report. For your chosen embedding, you must experiment with different model depths by training BiLSTM+CRF models with the number of stacked BiLSTM layers:

$$L \in \{1, 2, 3\} \quad \text{(or more if you wish, but at least three settings must be compared).}$$

You must report the hyperparameters used for training (e.g., hidden size, dropout, optimizer, learning rate, batch size, number of epochs, etc.) and include clear training and validation plots (loss curves and/or metric curves) for your experiments.

The models must be evaluated using the following metrics: FreeMatch-F1 and Strict EM (Exact Match) as defined in class. If you apply padding, padded positions must be excluded from all evaluation metrics. Summarize your results in a table (example format shown below) and provide an in-depth analysis in your report discussing the effect of using a CRF layer for structured decoding.

Layers (L)	FreeMatch-F1	Strict EM
1		
2		
3		

Deliverables (Q2).

1. **Implementation:** Provide your complete implementation for preprocessing, model training, evaluation, and inference in a single file named `part2.py` or `part2.ipynb`.

2. **Best-performing model:** Submit the saved checkpoint/weights of your best-performing configuration (clearly specify the chosen embedding and number of layers it corresponds to). Name the file in a self-explanatory manner (e.g., `best_bilstmcrf_fasttext_L3.pt`) and ensure it can be loaded by your `part2.py/part2.ipynb` for inference.

3. **Detailed report:** Make sure that the report includes:

   (a) a short description of your BiLSTM+CRF architecture, training setup, any preprocessing steps, and the full list of hyperparameters used,

   (b) training and validation plots (loss curves and/or metric curves) for all experiments,

   (c) a complete results table reporting FreeMatch-F1 and Strict EM for all layer settings ($L \in \{1, 2, 3\}$) using your chosen embedding, also attach the screenshot of the EM and FreeMatch-F1 score for the best performing model.

   (d) an in-depth analysis discussing the effect of increasing depth and explaining any differences in trends between FreeMatch-F1 and Strict EM.

Q3. BiLSTM + Bahdanau Attention

In this question, you will train a sequence tagging model that uses the **Bahdanau attention** mechanism to improve context modeling for BIO prediction. You will use this mechanism to build an attention-based tagger for BIO sequence labeling. You must implement the Bahdanau attention mechanism yourself as a PyTorch module named `BahdanauAttention` and not import a ready-made Bahdanau attention module from external libraries.

You must choose **any one** pretrained embedding of your choice (GloVe / fastText) and clearly mention your choice in the report. Using your chosen embedding, you must train a BiLSTM + Bahdanau Attention model for BIO prediction. You must experiment with different model depths by training models with the number of stacked BiLSTM layers:

$$L \in \{1, 2, 3\} \quad \text{(or more if you wish, but at least three settings must be compared).}$$

You must report the hyperparameters used for training (e.g., hidden size, dropout, optimizer, learning rate, batch size, number of epochs, etc.) and include clear training and validation plots (loss curves and/or metric curves) for your experiments.

The models must be evaluated using the following metrics: FreeMatch-F1 and Strict EM (Exact Match) as defined in class. If you apply padding, padded positions must be excluded from all evaluation metrics. Summarize your results in a table (example format shown below) and provide an in-depth analysis in your report discussing how attention and model depth affect performance.

Layers (L)	FreeMatch-F1	Strict EM
1		
2		
3		

Deliverables (Q3).

1. **Implementation:** Provide your complete implementation for preprocessing, model training, evaluation, and inference in a single file named `part3.py` or `part3.ipynb`.
2. **Best-performing model:** Submit the saved checkpoint/weights of your best-performing configuration (clearly specify the chosen embedding and number of layers it corresponds to). Name the file in a self-explanatory manner (e.g., `best_attn_fasttext_L3.pt`) and ensure it can be loaded by your `part3.py/part3.ipynb` for inference.
3. **Detailed report:** Make sure that the report includes:
   (a) a short description of your BiLSTM+Bahdanau-Attention architecture, training setup, any preprocessing steps, and the full list of hyperparameters used,
   (b) training and validation plots (loss curves and/or metric curves) for all experiments,
   (c) a complete results table reporting FreeMatch-F1 and Strict EM for all layer settings ($L \in \{1, 2, 3\}$) using your chosen embedding, also attach the screenshot of the EM and FreeMatch-F1 score for the best performing model.
   (d) an in-depth analysis discussing the effect of attention and increasing depth, and explaining any differences in trends between FreeMatch-F1 and Strict EM.