Objective

The goal of this assignment is to design, implement, train, and evaluate a Conditional Random Field (CRF) model for Named Entity Recognition (NER). This involves both theoretical derivation and practical coding. By the end of this assignment, you should have a complete understanding of how CRFs work and how they can be applied to sequence labeling tasks like NER.

Detailed Instructions

1. Derive the Mathematical Formulation of a Linear-Chain CRF

Objective: Understand the theoretical foundation of CRFs. **What to Do:**

- Start by explaining what Conditional Random Fields are and why they are used for sequence labeling tasks.
- Derive the conditional probability distribution formula for a linear-chain CRF. Explain how it models the probability of a sequence of labels given a sequence of inputs.
- Define key components such as:
  - **Feature functions:** Functions that capture relationships between input data, labels, and transitions.
  - **Weights:** Parameters that determine the importance of each feature.
  - **Partition function:** A normalization factor that ensures probabilities sum to 1.
- Explain inference in CRFs using algorithms like the Viterbi algorithm for finding the most likely sequence of labels.

2. Implement the CRF Model Using Python

Objective: Build a linear-chain CRF model from scratch using Python libraries like NumPy and SciPy. **What to Do:**

- Write code to represent key components of a CRF:
  - Feature extraction: Extract features from input data (e.g., words, POS tags, capitalization).
  - Parameter initialization: Initialize weights for feature functions.
  - Log-likelihood computation: Implement the log-likelihood function and its gradient.
  - Inference: Use dynamic programming (e.g., Viterbi algorithm) to find the most probable label sequence.
- Avoid using pre-built CRF libraries. Instead, focus on implementing the core logic yourself.

3. Train the Model on a Provided Dataset

Objective: Train your CRF model using real-world labeled data. **What to Do:**

- Use the dataset provided on the piazza.
- Optimize model parameters using numerical optimization techniques such as gradient descent or L-BFGS. Use libraries like SciPy for optimization if needed.

4. Evaluate the Model's Performance

Objective: Measure how well your model performs on unseen data. **What to Do:**

- Evaluate your model on the test set using standard metrics:

  Precision: The proportion of correctly predicted entities out of all predicted entities.

  Recall: The proportion of correctly predicted entities out of all actual entities in the data.

  F1-score: The harmonic mean of precision and recall. This provides a balanced evaluation metric.

- Provide a detailed analysis of your results:

- Highlight which entity types (e.g., PERSON, LOCATION) perform well and which do not.
- Discuss possible reasons for errors or misclassifications.

5. Write a Report

Summarize your work in a report that includes:

1. An introduction explaining Named Entity Recognition and Conditional Random Fields.
2. A detailed explanation of your mathematical derivations.
3. A description of your implementation approach, including challenges faced and how you overcame them.
4. Results from training and evaluation, along with insights gained from analyzing your model's performance.

Expected Deliverables

You are expected to submit:

1. A Python script or notebook containing:
   (a) Your implementation of the CRF model from scratch.
   (b) Code for training and evaluating the model on the dataset.
2. A report summarizing your work as described above.

Additional Notes

1. Ensure that your code is well-documented with comments explaining each step.
2. If you use any external resources or libraries beyond NumPy/SciPy, cite them in your report.
3. Pay attention to computational efficiency when implementing algorithms like Viterbi or gradient computation.

Example Input/Output

To clarify expectations, here's an example:

Input: A sentence represented as tokens:

[“*Barack*”, “*Obama*”, “*visited*”, “*India*”, “*last*”, “*week*”]

Output: Predicted entity tags for each token:

[(“*Barack*”, “*B*−*PER*”), (“*Obama*”, “*I*−*PER*”), (“*visited*”, ”*O*”), (“*India*”, ”*B*−*LOC*”), (“*last*”, ”*O*”), (“*week*”, ”*O*”)]

Good luck!