ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÀI TẬP
MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN

KHOA: KHOA HỌC MÁY TÍNH

HOMEWORK: Conditional Random Fields

Nhóm thực hiện:

1. Trương Văn Khải– 21520274

Câu 3:

Bài làm:

Lý do 1: Việc sử dụng hàm tối ưu L-BFGS-B có thể khiến quá trình huấn luyện bị chậm khi huấn luyện với lượng dữ liệu lớn.

Trong bài báo **An Introduction to Conditional Random Fields (**https://arxiv.org/pdf/1011.4088.pdf), ở mục 4.4.1. Pseudolikelihood có nhắc đến:

Rằng pseudolikelihood có thể kết hợp chậm trong mô hình tuần tự, tương tự như quá trình trộn chậm của Gibbs sampler. Điều này ngụ ý rằng quá trình tối ưu hóa có thể hội tụ chậm, đặc biệt khi xử lý dữ liệu lớn. Từ đó khiến quá trình huấn luyện với tập dữ liệu lớn bị chậm đi.

Lý do 2: Trong bài báo **AIN: Fast and Accurate Sequence Labeling with Approximate Inference Network** (https://arxiv.org/pdf/2009.08229.pdf) có nhắc đến:

Quá trình huấn luyện và dự đoán của CRF bằng cách sử dụng các thuật toán suy diễn xác suất chính xác như forward-backward và Viterbi đòi hỏi phải tính toán tuần tự. Tức là CRF layer phải xét các trường thông tin từ các thành phần trước đó trong chuỗi để đưa ra dự đoán. Điều này làm tăng độ phức tạp tính toán với độ dài chuỗi, hay với một tập dữ liệu lớn, và việc này không thể được hiệu quả hóa bằng các sử dụng phương pháp song song hóa trên GPU.

Ngoài ra, CRF đòi hỏi thời gian tuyến tính O(n) đối với n từ đầu vào, điều này ngụ ý là nếu CRF đã được song song hóa trên GPU, thì vẫn phải đòi hỏi O(n) thời gian xử lý mỗi từ vựng.

Lý do 3: Khi mô hình CRF đang được huấn luyện, nó sẽ sử dụng CPU rất nhiều, điều này được nêu ra nổi bật ở đây:

Model	CPU(%)	GPU(%)
Bi-LSTM+CRF(AllenNLP)	1124	1
Bi-LSTM(AllenNLP)	57	6
Bi-LSTM+CRF(Keras)	235	13

Bảng so sánh việc sử dụng GPU, GPU trong quá trình huấn luyện
https://github.com/allenai/allennlp/issues/2884

Có thể thấy được, mô hình Bi-LSTM có sử dụng CRF và không sử dụng CRF có sự khác như rất lớn về việc tiêu thụ CPU.

Điều này được giải thích là vì CRF có sử dụng Viterbi làm decoder, tương tự như ở đây https://github.com/kmkurn/pytorch-crf/issues/72. Trong đường dẫn này, việc CRF layer sử dụng thuật toán Viterbi thường chậm hơn từ 8-10 lần so với BERT-Tagger, sử dụng thuật toán tham lam để chọn xác suất cao nhất tại mỗi thời điểm.

Ngoài ra, Viterbi cũng sẽ bị chậm đối với loại dữ liệu có tính tuần tự cao, chẳng hạn như dữ liệu dạng văn bản. Khi đó, thuật toán Viterbi sẽ phải thuật hiện nhiều phép toán tuần tự, từ đó làm giảm khả năng tận dụng tính toán song song.

Giải pháp:

- Sử dụng các hàm tối ưu khác như Stochastic Gradient Descent (SGD) với learning_rate được Grid Search nhằm đảm bảo tính tối ưu nhất của hàm tối ưu hóa.
- Tensor hóa tất cả dữ liệu đầu vào, đảm bảo được việc tối ưu hóa dữ liệu và tận dụng khả năng tính toán song song của phần cứng.
- Thuật toán Viterbi: Nếu có t quan sát, s trạng thái và mỗi trạng thái có xác suất phát xạ e thì lưới sẽ có các nút t*s và để đánh giá mỗi nút sẽ tốn e hoạt động, do đó độ phức tạp tổng thể của việc triển khai sẽ là O(t *s*e). Vì vậy, hãy thử giảm số lượng trạng thái ẩn trong mô hình hoặc kiểm tra bằng thuật toán ngẫu nhiên có thể chạy nhanh hơn nhiều hay không. Tuy nhiên điều này không đảm bảo luôn trả về kết quả hoàn toàn tốt nhất.