lab5_rnn_for_pos_tagging

Harito ID

2025-11-11

Lab 5: Xây dựng mô hình RNN cho bài toán Part-of-Speech Tagging

1. Mục tiêu

Trong bài thực hành này, chúng ta sẽ áp dụng các kiến thức lý thuyết về Mạng Nơ-ron Hồi quy (RNN) đã học để xây dựng một mô hình hoàn chỉnh cho bài toán Part-of-Speech (POS) Tagging.

Sau khi hoàn thành bài lab, bạn có thể:

- Tải và tiền xử lý dữ liệu văn bản từ định dạng CoNLL-U.
- Xây dựng từ điển (vocabulary) cho từ và nhãn.
- Tạo một lớp `Dataset` tùy chỉnh trong PyTorch.
- Xây dựng một mô hình RNN đơn giản từ các khối `nn.Embedding`, `nn.RNN`, và `nn.Linear`.
- Huấn luyện và đánh giá hiệu năng của mô hình trên một bộ dữ liệu thực tế.

2. Bộ dữ liệu: Universal Dependencies (UD_English-EWT)

Chúng ta sẽ sử dụng bộ dữ liệu có sẵn trong thư mục `data/UD_English-EWT/`. Dữ liệu này ở định dạng CoNLL-U, một định dạng phổ biến cho các bài toán NLP có cấu trúc.

Một file `.conllu` bao gồm các câu, mỗi câu được ngăn cách bởi một dòng trống. Mỗi dòng trong một câu chứa thông tin về một token, các trường thông tin được ngăn cách bởi tab. Chúng ta chỉ cần quan tâm đến 2 cột:

- **Cột 2 (FORM):** Từ gốc.
- **Cột 4 (UPOS):** Nhãn Part-of-Speech theo chuẩn Universal.

Ví dụ:

```
# sent_id = weblog-juancole.com_juancole_20051126063000_ENG_20051126_063000-0003
# text = From the AP comes this story:
1 From from ADP IN _ 3 case _ _
2 the the DET DT Definite=Def|PronType=Art 3 det _ _
3 AP AP PROPN NNP Number=Sing 4 nsubj _ _
4 comes come VERB VBZ Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin 0 root _ _
5 this this DET DT Number=Sing|PronType=Dem 6 det _ _
6 story story NOUN NN Number=Sing 4 obj _ _
7 : : PUNCT : _ 4 punct _ _
```

3. Các bước thực hiện

Task 1: Tải và Tiền xử lý Dữ liệu

1. **Viết hàm đọc file `.conllu`:**
    - Viết một hàm `load_conllu(file_path)` để đọc dữ liệu từ các file `en_ewt-ud-train.conl` và `en_ewt-ud-dev.conllu`.
    - Hàm này cần trả về một danh sách các câu. Mỗi câu là một danh sách các cặp `(word, upos_tag)`.
    - Ví dụ: `[[('From', 'ADP'), ('the', 'DET'), ...], [('Another', 'DET'), ('sentence', 'NOUN'), ...]]`
2. **Xây dựng Từ điển (Vocabulary):**
    - Từ dữ liệu huấn luyện (train), tạo ra hai từ điển:
        - `word_to_ix`: Ánh xạ mỗi từ duy nhất sang một chỉ số (index) nguyên. Thêm một token đặc biệt là `<UNK>` cho các từ không có trong từ điển.
        - `tag_to_ix`: Ánh xạ mỗi nhãn UPOS duy nhất sang một chỉ số nguyên.
    - In ra kích thước của hai từ điển này.

Task 2: Tạo PyTorch Dataset và DataLoader

1. **Tạo lớp `POSDataset`:**
    - Tạo một lớp kế thừa từ `torch.utils.data.Dataset`.
    - `__init__`: Nhận vào danh sách các câu đã xử lý và hai từ điển `word_to_ix`, `tag_to_ix`.
    - `__len__`: Trả về tổng số câu trong bộ dữ liệu.
    - `__getitem__`: Nhận vào một `index` và trả về một cặp tensor: `(sentence_indices, tag_indices)`. Tensor này chứa các chỉ số của từ/nhãn trong câu tương ứng.
2. **Tạo `DataLoader`:**
    - Khởi tạo `DataLoader` cho cả tập train và dev.
    - **Lưu ý quan trọng:** Các câu trong một batch có độ dài khác nhau. Bạn cần viết một hàm `collate_fn` để đệm (pad) các câu và nhãn trong cùng một batch về cùng một độ dài (độ dài của câu dài nhất batch đó). Sử dụng `torch.nn.utils.rnn.pad_sequence` với `batch_first=True`.

Task 3: Xây dựng Mô hình RNN

- Dựa trên đoạn code khái niệm trong bài giảng, hãy hoàn thiện lớp `SimpleRNNForTokenClas`
- Mô hình sẽ bao gồm 3 lớp chính:
    1. `nn.Embedding`: Chuyển đổi chỉ số của từ thành vector.
    2. `nn.RNN`: Xử lý chuỗi vector embedding.
    3. `nn.Linear`: Ánh xạ output của RNN sang không gian nhãn để dự đoán.
- Hãy chú ý đến kích thước (dimension) của các tensor ở đầu vào và đầu ra của mỗi lớp.

Task 4: Huấn luyện Mô hình

1. **Khởi tạo:**
    - Khởi tạo mô hình, optimizer (ví dụ: `torch.optim.Adam`), và loss function.

- Sử dụng `nn.CrossEntropyLoss` cho bài toán này. Lưu ý rằng `CrossEntropyLoss` yêu cầu đầu vào là raw scores (logits) và bỏ qua các giá trị đệm (padding) khi tính loss. Bạn có thể đặt `ignore_index` cho giá trị padding của nhãn.

2. **Viết vòng lặp huấn luyện:**
    - Lặp qua một số lượng `epoch` nhất định.
    - Trong mỗi epoch, lặp qua từng `batch` từ `DataLoader` huấn luyện.
    - Thực hiện 5 bước kinh điển: (1) Xóa gradient cũ, (2) Forward pass, (3) Tính loss, (4) Backward pass (lan truyền ngược), (5) Cập nhật trọng số.
    - In ra giá trị loss trung bình sau mỗi epoch hoặc sau một số lượng batch nhất định.

Task 5: Đánh giá Mô hình

1. **Viết hàm `evaluate`:**
    - Đặt mô hình ở chế độ đánh giá: `model.eval()`.
    - Tắt việc tính toán gradient: `with torch.no_grad(): ...`
    - Lặp qua từng batch trong `DataLoader` của tập dev.
    - Với mỗi batch, lấy dự đoán của mô hình bằng cách áp dụng `torch.argmax` trên chiều cuối cùng của output.
    - So sánh dự đoán với nhãn thật để tính toán độ chính xác (accuracy). **Lưu ý:** chỉ tính accuracy trên các token không phải là padding.
2. **Báo cáo kết quả:**
    - Show độ chính xác trên tập train và dev sau mỗi epoch huấn luyện. Lựa chọn mô hình tốt nhất dựa trên độ chính xác trên tập dev.
    - In ra độ chính xác cuối cùng trên tập dev.
    - (Nâng cao) Viết một hàm `predict_sentence(sentence)` nhận vào một câu mới (dạng chuỗi), xử lý nó và in ra các cặp `(từ, nhãn_dự_đoán)`.

4. Nộp bài

- Nộp link report cho bài lab.

KẾT QUẢ THỰC HIỆN

(Dành cho sinh viên điền vào)

- **Độ chính xác trên tập dev:** ...
- **Ví dụ dự đoán câu mới:**
    - **Câu:** “I love NLP”
    - **Dự đoán:** ...