```
!pip install seqeval --quiet
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.6/43.6 kB 1.3 MB/s eta
0:00:00
etadata (setup.py) ...
```

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
from torch.nn.utils.rnn import pad_sequence
from datasets import load_dataset
from tqdm import tqdm
from seqeval.metrics import classification_report, f1_score

BATCH_SIZE = 16
EMBEDDING_DIM = 100
HIDDEN_DIM = 256
OUTPUT_SIZE = 9
NUM_LAYERS = 1
EPOCHS = 10
LEARNING_RATE = 0.001
PAD_TAG = -1
PATIENCE = 3

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"Using device: {device}")
```

```
Using device: cpu
```

Task 1

```
dataset = load_dataset("lhoestq/conll2003")
train_data = dataset["train"]
val_data = dataset["validation"]
test_data = dataset["test"]
```


```
{"model_id":"62b4f9e67b6d4cfaa7bd62303580b283","version_major":2,"vers
ion_minor":0}

{"model_id":"b98aaab4a0cc48bf8e8bc0e381c9fc8f","version_major":2,"vers
ion_minor":0}

{"model_id":"635c0bb31fbd4833bc9d0f77c1da0c12","version_major":2,"vers
ion_minor":0}

{"model_id":"d74d2adc9d944f9f888358917bc44d78","version_major":2,"vers
ion_minor":0}
```

{"model_id":"9623e26074824c4c864e888c0ab38973","version_major":2,"version_minor":0}

{"model_id":"e4d8c0c751dc4112817e70cad06818cd","version_major":2,"version_minor":0}

{"model_id":"e5568ac48e4e496784d1812ae53fa8be","version_major":2,"version_minor":0}

```
word_to_ix = {"<PAD>": 0, "<UNK>": 1}
idx = 2
for sent in train_data["tokens"]:
    for tok in sent:
        if tok not in word_to_ix:
            word_to_ix[tok] = idx
            idx += 1

print(f"Vocab size: {len(word_to_ix)}")

id2tag = [
    "O", "B-PER", "I-PER", "B-ORG", "I-ORG",
    "B-LOC", "I-LOC", "B-MISC", "I-MISC"
]
```

```
Vocab size: 23625
```

Task 2

```
class NERDataset(Dataset):
    def __init__(self, sentences, tags_ids, word_to_ix):
        self.sentences = sentences
        self.tags_ids = tags_ids
        self.word_to_ix = word_to_ix
        self.unk_id = word_to_ix.get("<UNK>", 1)

    def __len__(self):
        return len(self.sentences)

    def __getitem__(self, idx):
        sent = self.sentences[idx]
        tag_ids = self.tags_ids[idx]

        sent_ids = [self.word_to_ix.get(tok, self.unk_id) for tok in
sent]

        return torch.tensor(sent_ids, dtype=torch.long),
torch.tensor(tag_ids, dtype=torch.long)

def collate_fn(batch):
```

```
    sentences, tags = zip(*batch)
    padded_sentences = pad_sequence(sentences, batch_first=True,
padding_value=0)
    padded_tags = pad_sequence(tags, batch_first=True,
padding_value=PAD_TAG)
    return padded_sentences, padded_tags

train_loader = DataLoader(
    NERDataset(train_data["tokens"], train_data["ner_tags"],
word_to_ix),
    batch_size=BATCH_SIZE, shuffle=True, collate_fn=collate_fn
)

val_loader = DataLoader(
    NERDataset(val_data["tokens"], val_data["ner_tags"], word_to_ix),
    batch_size=BATCH_SIZE, shuffle=False, collate_fn=collate_fn
)

test_loader = DataLoader(
    NERDataset(test_data["tokens"], test_data["ner_tags"],
word_to_ix),
    batch_size=BATCH_SIZE, shuffle=False, collate_fn=collate_fn
)
```

Task 3

```
class NERLSTMModel(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim,
output_size, num_layers=1):
        super(NERLSTMModel, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.lstm = nn.LSTM(
            input_size=embedding_dim,
            hidden_size=hidden_dim,
            num_layers=num_layers,
            batch_first=True,
            bidirectional=True)
        self.fc = nn.Linear(hidden_dim * 2, output_size)

    def forward(self, x):
        embedded = self.embedding(x)
        lstm_out, _ = self.lstm(embedded)
        logits = self.fc(lstm_out)
        return logits
```

Task 4

```
model = NERLSTMModel(len(word_to_ix), EMBEDDING_DIM, HIDDEN_DIM,
OUTPUT_SIZE, NUM_LAYERS).to(device)
criterion = nn.CrossEntropyLoss(ignore_index=PAD_TAG)
optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE)

trigger_times = 0
best_val_loss = float('inf')

for epoch in range(EPOCHS):
    model.train()
    total_loss = 0.0

    progress_bar = tqdm(train_loader, desc=f"Epoch
{epoch+1}/{EPOCHS}")

    for sentences, tags in progress_bar:
        sentences, tags = sentences.to(device), tags.to(device)

        optimizer.zero_grad()
        predictions = model(sentences)

        predictions = predictions.view(-1, OUTPUT_SIZE)
        tags = tags.view(-1)

        loss = criterion(predictions, tags)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()
        progress_bar.set_postfix(loss=loss.item())

    avg_train_loss = total_loss / len(train_loader)

    model.eval()
    total_val_loss = 0.0
    val_correct = 0
    val_total = 0

    with torch.no_grad():
        for sentences, tags in val_loader:
            sentences, tags = sentences.to(device), tags.to(device)

            outputs = model(sentences) # [Batch, Seq, Output]

            outputs_flat = outputs.view(-1, OUTPUT_SIZE)
            tags_flat = tags.view(-1)
            loss = criterion(outputs_flat, tags_flat)
            total_val_loss += loss.item()
```

```
            preds = torch.argmax(outputs, dim=2)
            mask = (tags != PAD_TAG) # Bỏ qua padding
            val_correct += (preds[mask] == tags[mask]).sum().item()
            val_total += tags[mask].numel()

    avg_val_loss = total_val_loss / len(val_loader)
    val_acc = val_correct / val_total if val_total > 0 else 0

    print(f"Epoch {epoch+1}: Train Loss: {avg_train_loss:.4f} | Val
Loss: {avg_val_loss:.4f} | Val Acc: {val_acc:.4f}")

    if avg_val_loss < best_val_loss:
        best_val_loss = avg_val_loss
        trigger_times = 0
    else:
        trigger_times += 1

        if trigger_times >= PATIENCE:
            print(f"Early Stopping! Stop training at epoch {epoch+1}")
            break
```

```
Epoch 1/10: 100%|██████████| 878/878 [00:59<00:00, 14.72it/s,
loss=0.126]

Epoch 1: Train Loss: 0.3705 | Val Loss: 0.2642 | Val Acc: 0.9262

Epoch 2/10: 100%|██████████| 878/878 [01:01<00:00, 14.36it/s,
loss=0.114]

Epoch 2: Train Loss: 0.1588 | Val Loss: 0.2051 | Val Acc: 0.9431

Epoch 3/10: 100%|██████████| 878/878 [01:01<00:00, 14.26it/s,
loss=0.0747]

Epoch 3: Train Loss: 0.0716 | Val Loss: 0.1841 | Val Acc: 0.9483

Epoch 4/10: 100%|██████████| 878/878 [01:01<00:00, 14.29it/s,
loss=0.0154]

Epoch 4: Train Loss: 0.0270 | Val Loss: 0.1942 | Val Acc: 0.9519

Epoch 5/10: 100%|██████████| 878/878 [01:02<00:00, 14.13it/s,
loss=0.00188]

Epoch 5: Train Loss: 0.0089 | Val Loss: 0.2128 | Val Acc: 0.9525

Epoch 6/10: 100%|██████████| 878/878 [01:02<00:00, 14.13it/s,
loss=0.00164]

Epoch 6: Train Loss: 0.0035 | Val Loss: 0.2415 | Val Acc: 0.9526
Early Stopping! Stop training at epoch 6
```

Task 5

```
def predict_sentence(sentence):
    model.eval()
    tokens = sentence.split()
    unk_id = word_to_ix.get("<UNK>", 1)
    input_ids = [word_to_ix.get(word, unk_id) for word in tokens]
    input_tensor = torch.tensor([input_ids],
dtype=torch.long).to(device)

    with torch.no_grad():
        outputs = model(input_tensor)
        prediction_indices = torch.argmax(outputs, dim=2)

    predicted_tags = [id2tag[idx] for idx in
prediction_indices[0].tolist()]

    print(f"\nResult for: '{sentence}'")
    print(f"{'Word':<15} {'Pred Tag':<15}")
    print("-" * 30)
    for word, tag in zip(tokens, predicted_tags):
        print(f"{word:<15} {tag:<15}")

test_sentence = "On Monday, October 14, 2024, Google CEO Sundar Pichai
traveled from Mountain View, California to New York City to meet with
Microsoft executives including Satya Nadella and Amy Hood at the
headquarters of the United Nations, where they announced a joint
initiative with NASA and SpaceX to develop advanced AI-powered
satellites capable of monitoring climate change, while also signing
agreements with the European Space Agency, Tesla, and Siemens to
collaborate on renewable energy projects that aim to reduce global
carbon emissions by 30 percent within the next decade, all before
attending a high-profile conference at Harvard University hosted by
the World Economic Forum, which included keynote speeches from former
U.S. President Barack Obama, Nobel laureate Malala Yousafzai, and
philanthropist Bill Gates."
predict_sentence(test_sentence)
```

```
Result for: 'On Monday, October 14, 2024, Google CEO Sundar Pichai
traveled from Mountain View, California to New York City to meet with
Microsoft executives including Satya Nadella and Amy Hood at the
headquarters of the United Nations, where they announced a joint
initiative with NASA and SpaceX to develop advanced AI-powered
satellites capable of monitoring climate change, while also signing
agreements with the European Space Agency, Tesla, and Siemens to
collaborate on renewable energy projects that aim to reduce global
carbon emissions by 30 percent within the next decade, all before
attending a high-profile conference at Harvard University hosted by
the World Economic Forum, which included keynote speeches from former
```

```
U.S. President Barack Obama, Nobel laureate Malala Yousafzai, and
philanthropist Bill Gates.'
Word            Pred Tag
------------------------------
On              O
Monday,         O
October         O
14,             B-MISC
2024,           O
Google          B-MISC
CEO             O
Sundar          O
Pichai          B-MISC
traveled        O
from            O
Mountain        B-LOC
View,           O
California      B-LOC
to              O
New             B-LOC
York            I-LOC
City            I-LOC
to              O
meet            O
with            O
Microsoft       B-ORG
executives      O
including       O
Satya           B-ORG
Nadella         I-PER
and             O
Amy             B-PER
Hood            O
at              O
the             O
headquarters    O
of              O
the             O
United          B-LOC
Nations,        O
where           O
they            O
announced       O
a               O
joint           O
initiative      O
with            O
NASA            O
and             O
```

```
SpaceX          O
to              O
develop         O
advanced        O
AI-powered      O
satellites      O
capable         O
of              O
monitoring      O
climate         O
change,         O
while           O
also            O
signing         O
agreements      O
with            O
the             O
European        B-MISC
Space           I-MISC
Agency,         I-MISC
Tesla,          I-MISC
and             O
Siemens         B-MISC
to              O
collaborate     O
on              O
renewable       O
energy          O
projects        O
that            O
aim             O
to              O
reduce          O
global          O
carbon          O
emissions       O
by              O
30              O
percent         O
within          O
the             O
next            O
decade,         O
all             O
before          O
attending       O
a               O
high-profile    O
conference      O
```

```
at              O
Harvard         B-ORG
University      I-ORG
hosted          O
by              O
the             O
World           B-MISC
Economic        I-MISC
Forum,          I-MISC
which           O
included        O
keynote         O
speeches        O
from            O
former          O
U.S.            B-LOC
President       O
Barack          B-PER
Obama,          I-PER
Nobel           I-PER
laureate        I-PER
Malala          O
Yousafzai,      I-PER
and             O
philanthropist  B-MISC
Bill            I-PER
Gates.          I-PER
```

```python
def align_predictions(predictions, label_ids, id2tag, pad_tag_id=-1):
    preds_list = []
    labels_list = []
    for i in range(len(label_ids)):
        sent_preds = []
        sent_labels = []
        for j in range(len(label_ids[i])):
            if label_ids[i][j] != pad_tag_id:
                sent_preds.append(id2tag[predictions[i][j]])
                sent_labels.append(id2tag[label_ids[i][j]])
        preds_list.append(sent_preds)
        labels_list.append(sent_labels)
    return preds_list, labels_list

def evaluate_ner_metrics(model, data_loader, id2tag, device,
pad_tag_id=-1):
    model.eval()
    all_preds = []
    all_labels = []
    with torch.no_grad():
        for sentences, tags in data_loader:
            sentences = sentences.to(device)
```

```
            tags = tags.to(device)
            outputs = model(sentences)
            predictions = torch.argmax(outputs, dim=2)
            predictions_np = predictions.cpu().numpy()
            tags_np = tags.cpu().numpy()
            batch_preds, batch_labels =
align_predictions(predictions_np, tags_np, id2tag, pad_tag_id)
            all_preds.extend(batch_preds)
            all_labels.extend(batch_labels)
    print("\n" + "="*55)
    print(classification_report(all_labels, all_preds))
    print(f"Macro F1-Score: {f1_score(all_labels, all_preds):.4f}")
    print("="*55 + "\n")

def evaluate(model, data_loader):
    model.eval()
    correct_preds = 0
    total_preds = 0
    with torch.no_grad():
        for sentences, tags in data_loader:
            sentences, tags = sentences.to(device), tags.to(device)
            outputs = model(sentences)
            predictions = torch.argmax(outputs, dim=2)

            mask = (tags != PAD_TAG)
            valid_predictions = predictions[mask]
            valid_tags = tags[mask]

            correct_preds += (valid_predictions ==
valid_tags).sum().item()
            total_preds += valid_tags.numel()

    return correct_preds / total_preds if total_preds > 0 else 0.0

evaluate(model, test_loader)
```

```
0.9307418972757618
```

```
evaluate_ner_metrics(model, test_loader, id2tag, device)
```

```
=======================================================
              precision    recall  f1-score   support

         LOC       0.86      0.68      0.76      1668
        MISC       0.45      0.64      0.53       702
         ORG       0.74      0.57      0.64      1661
         PER       0.65      0.67      0.66      1617

   micro avg       0.69      0.64      0.66      5648
   macro avg       0.68      0.64      0.65      5648
```

```
weighted avg       0.71      0.64      0.67      5648

Macro F1-Score: 0.6622
============================================================
```