```
# Cell 0: Cài đặt thư viện cần thiết
!pip install datasets seqeval
```

```
Requirement already satisfied: datasets in /usr/local/lib/python3.12/dist-packages (4.0.0)
Collecting seqeval
  Downloading seqeval-1.2.2.tar.gz (43 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.6/43.6 kB 1.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from datasets) (3.20.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from datasets) (2.0.2)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (18.1.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.12/dist-packages (from datasets) (2.32.4)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.12/dist-packages (from datasets) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.12/dist-packages (from datasets) (3.6.0)
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2025.3.0,>=2023.1.0 in /usr/local/lib/python3.12/dist-packages (from fsspec[http]<=2025.3
Requirement already satisfied: huggingface-hub>=0.24.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.36.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from datasets) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from datasets) (6.0.3)
Requirement already satisfied: scikit-learn>=0.21.3 in /usr/local/lib/python3.12/dist-packages (from seqeval) (1.6.1)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.12/dist-packages (from fsspec[http]<=2025.3.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.24
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.24.0->da
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.32.2->datas
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests>=2.32.2->datasets) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.32.2->datasets) (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests>=2.32.2->datasets) (
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=0.21.3->seqeval) (1.1
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=0.21.3->seqeval) (1.
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=0.21.3->seqev
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2.9.0.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fss
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fs
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fss
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fs
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas->dataset
Building wheels for collected packages: seqeval
  Building wheel for seqeval (setup.py) ... done
  Created wheel for seqeval: filename=seqeval-1.2.2-py3-none-any.whl size=16162 sha256=103d1877d8949fd85682fc064287141b884f6e98c
  Stored in directory: /root/.cache/pip/wheels/5f/b8/73/0b2c1a76b701a677653dd79ece07cfabd7457989dbfbdcd8d7
Successfully built seqeval
Installing collected packages: seqeval
Successfully installed seqeval-1.2.2
```

```
# Cell 1: Import và tải dữ liệu CoNLL 2003 (bản mirror)

import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from torch.nn.utils.rnn import pad_sequence

from datasets import load_dataset

import numpy as np
import random

# Đặt seed cho reproducibility
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed_all(SEED)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device
```

```
device(type='cpu')
```

```
# Lấy tokens và ner_tags
train_sentences = dataset["train"]["tokens"]
train_tags_ids = dataset["train"]["ner_tags"]

val_sentences = dataset["validation"]["tokens"]
val_tags_ids = dataset["validation"]["ner_tags"]

# Vì bản parquet mới không lưu tên nhãn → tự định nghĩa nhãn CoNLL2003
tag_names = [
    "O",
    "B-MISC", "I-MISC",
    "B-PER", "I-PER",
    "B-ORG", "I-ORG",
    "B-LOC", "I-LOC"
]

print("Tag names:", tag_names)
```

```
Tag names: ['O', 'B-MISC', 'I-MISC', 'B-PER', 'I-PER', 'B-ORG', 'I-ORG', 'B-LOC', 'I-LOC']
```

```
# Cell 2: Chuyển ner_tags (id) -> string và xây vocabulary

# 1) Chuyển nhãn sang string
train_tags_str = [[tag_names[tag_id] for tag_id in seq] for seq in train_tags_ids]
val_tags_str   = [[tag_names[tag_id] for tag_id in seq] for seq in val_tags_ids]

# 2) Xây word_to_ix từ tập train
PAD_TOKEN = "<PAD>"
UNK_TOKEN = "<UNK>"

word_to_ix = {
    PAD_TOKEN: 0,
    UNK_TOKEN: 1,
}

for sent in train_sentences:
    for w in sent:
        if w not in word_to_ix:
            word_to_ix[w] = len(word_to_ix)

vocab_size = len(word_to_ix)

# 3) Xây tag_to_ix từ danh sách tag_names
tag_to_ix = {tag: i for i, tag in enumerate(tag_names)}
num_tags = len(tag_to_ix)

print("Số lượng từ (vocab_size):", vocab_size)
print("Số lượng nhãn NER:", num_tags)
print("Ví dụ word_to_ix (10 phần tử đầu):")
for i, (w, idx) in enumerate(word_to_ix.items()):
    print(w, "->", idx)
    if i >= 9:
        break

print("\nTag_to_ix:")
print(tag_to_ix)
```

```
Số lượng từ (vocab_size): 23625
Số lượng nhãn NER: 9
Ví dụ word_to_ix (10 phần tử đầu):
<PAD> -> 0
<UNK> -> 1
EU -> 2
rejects -> 3
German -> 4
call -> 5
to -> 6
boycott -> 7
British -> 8
lamb -> 9

Tag_to_ix:
```

```
{'O': 0, 'B-MISC': 1, 'I-MISC': 2, 'B-PER': 3, 'I-PER': 4, 'B-ORG': 5, 'I-ORG': 6, 'B-LOC': 7, 'I-LOC': 8}
```

```
# Cell 3: Định nghĩa Dataset cho NER

class NERDataset(Dataset):
    def __init__(self, sentences, tags, word_to_ix, tag_to_ix):
        """
        sentences: list[list[str]]
        tags: list[list[str]]  (nhãn dạng string)
        """
        self.sentences = sentences
        self.tags = tags
        self.word_to_ix = word_to_ix
        self.tag_to_ix = tag_to_ix

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

    def __getitem__(self, idx):
        words = self.sentences[idx]
        tags = self.tags[idx]

        word_ids = [
            self.word_to_ix.get(w, self.word_to_ix["<UNK>"])
            for w in words
        ]
        tag_ids = [
            self.tag_to_ix[t]
            for t in tags
        ]

        return word_ids, tag_ids
```

```
# Cell 3 (tiếp): collate_fn để pad

PAD_IDX = word_to_ix[PAD_TOKEN]
TAG_PAD_IDX = -1  # padding cho nhãn

def collate_fn(batch):
    """
    batch: list of (word_ids, tag_ids)
    """
    word_seqs, tag_seqs = zip(*batch)

    word_tensors = [torch.tensor(seq, dtype=torch.long) for seq in word_seqs]
    tag_tensors  = [torch.tensor(seq, dtype=torch.long) for seq in tag_seqs]

    padded_words = pad_sequence(word_tensors, batch_first=True, padding_value=PAD_IDX)
    padded_tags  = pad_sequence(tag_tensors,  batch_first=True, padding_value=TAG_PAD_IDX)

    lengths = torch.tensor([len(seq) for seq in word_seqs], dtype=torch.long)

    return padded_words, padded_tags, lengths
```

```
# Cell 3 (tiếp): DataLoader

batch_size = 32

train_dataset = NERDataset(train_sentences, train_tags_str, word_to_ix, tag_to_ix)
val_dataset   = NERDataset(val_sentences,   val_tags_str,   word_to_ix, tag_to_ix)

train_loader = DataLoader(
    train_dataset,
    batch_size=batch_size,
    shuffle=True,
    collate_fn=collate_fn
)

val_loader = DataLoader(
    val_dataset,
    batch_size=batch_size,
    shuffle=False,
    collate_fn=collate_fn
)

# Test 1 batch
batch_words, batch_tags, batch_lengths = next(iter(train_loader))
print("batch_words shape:", batch_words.shape)
print("batch_tags shape:", batch_tags.shape)
print("batch_lengths:", batch_lengths[:10])
```

```
batch_words shape: torch.Size([32, 47])
batch_tags shape: torch.Size([32, 47])
batch_lengths: tensor([10, 10, 15,  2,  5, 23,  4, 41,  2,  4])
```

```
# Cell 4: Định nghĩa mô hình LSTM cho NER

class SimpleRNNForTokenClassification(nn.Module):
    def __init__(
        self,
        vocab_size,
        tagset_size,
        embedding_dim=100,
        hidden_dim=128,
        num_layers=1,
        bidirectional=True,
        pad_idx=0
    ):
        super().__init__()

        self.embedding = nn.Embedding(
            num_embeddings=vocab_size,
            embedding_dim=embedding_dim,
            padding_idx=pad_idx
        )

        self.lstm = nn.LSTM(
            input_size=embedding_dim,
            hidden_size=hidden_dim,
            num_layers=num_layers,
            batch_first=True,
            bidirectional=bidirectional
        )

        direction_factor = 2 if bidirectional else 1
        self.fc = nn.Linear(hidden_dim * direction_factor, tagset_size)

    def forward(self, x):
        """
        x: (batch_size, seq_len)
        """
        embeddings = self.embedding(x)     # (B, L, E)
        outputs, _ = self.lstm(embeddings) # (B, L, H*dir)
        logits = self.fc(outputs)          # (B, L, num_tags)
        return logits
```

```
# Cell 4 (tiếp): Khởi tạo model, optimizer, loss

embedding_dim = 100
hidden_dim = 128
num_layers = 1
bidirectional = True

model = SimpleRNNForTokenClassification(
    vocab_size=vocab_size,
    tagset_size=num_tags,
    embedding_dim=embedding_dim,
    hidden_dim=hidden_dim,
    num_layers=num_layers,
    bidirectional=bidirectional,
    pad_idx=PAD_IDX
).to(device)

optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

criterion = nn.CrossEntropyLoss(ignore_index=TAG_PAD_IDX)
```

```
model
```

```
SimpleRNNForTokenClassification(
  (embedding): Embedding(23625, 100, padding_idx=0)
  (lstm): LSTM(100, 128, batch_first=True, bidirectional=True)
  (fc): Linear(in_features=256, out_features=9, bias=True)
)
```

```
# Cell 5: train_one_epoch

def train_one_epoch(model, dataloader, optimizer, criterion, device):
    model.train()
    total_loss = 0.0
    total_tokens = 0

    for batch_words, batch_tags, batch_lengths in dataloader:
        batch_words = batch_words.to(device)
        batch_tags  = batch_tags.to(device)

        optimizer.zero_grad()

        logits = model(batch_words)  # (B, L, C)

        B, L, C = logits.shape
        logits_flat = logits.view(B * L, C)
        targets_flat = batch_tags.view(B * L)

        loss = criterion(logits_flat, targets_flat)
        loss.backward()
        optimizer.step()

        with torch.no_grad():
            mask = (targets_flat != TAG_PAD_IDX)
            num_valid = mask.sum().item()
            total_loss += loss.item() * num_valid
            total_tokens += num_valid

    avg_loss = total_loss / max(total_tokens, 1)
    return avg_loss
```

```
# Cell 5 (tiếp): evaluate accuracy trên dev

def evaluate(model, dataloader, device):
    model.eval()
    total_correct = 0
    total_tokens = 0

    with torch.no_grad():
        for batch_words, batch_tags, batch_lengths in dataloader:
            batch_words = batch_words.to(device)
            batch_tags  = batch_tags.to(device)

            logits = model(batch_words)       # (B, L, C)
            preds = logits.argmax(dim=-1)     # (B, L)

            mask = (batch_tags != TAG_PAD_IDX)

            correct = (preds == batch_tags) & mask
            total_correct += correct.sum().item()
            total_tokens  += mask.sum().item()

    accuracy = total_correct / max(total_tokens, 1)
    return accuracy
```

```
# Cell 5 (tiếp): Chạy huấn luyện

num_epochs = 3

for epoch in range(1, num_epochs + 1):
    train_loss = train_one_epoch(model, train_loader, optimizer, criterion, device)
    val_acc = evaluate(model, val_loader, device)

    print(f"Epoch {epoch}/{num_epochs} - Train loss: {train_loss:.4f} - Val accuracy: {val_acc:.4f}")
```

```
final_val_acc = evaluate(model, val_loader, device)
print("\nĐộ chính xác cuối cùng trên tập validation:", final_val_acc)
```

```
Epoch 1/3 - Train loss: 0.5909 - Val accuracy: 0.8828
Epoch 2/3 - Train loss: 0.2670 - Val accuracy: 0.9069
Epoch 3/3 - Train loss: 0.1523 - Val accuracy: 0.9197

Độ chính xác cuối cùng trên tập validation: 0.9196877068649975
```

```
# Cell 6: Hàm predict_sentence(sentence)

ix_to_tag = {idx: tag for tag, idx in tag_to_ix.items()}

def predict_sentence(model, sentence, word_to_ix, ix_to_tag, device):
    """
    sentence: string
    """
    model.eval()
    tokens = sentence.split()

    word_ids = [word_to_ix.get(w, word_to_ix["<UNK>"]) for w in tokens]
    input_tensor = torch.tensor([word_ids], dtype=torch.long).to(device)  # (1, L)

    with torch.no_grad():
        logits = model(input_tensor)          # (1, L, C)
        preds = logits.argmax(dim=-1).squeeze(0).tolist()  # (L,)

    pred_tags = [ix_to_tag[p] for p in preds]

    print("Sentence:", sentence)
    print("Predictions:")
    for w, t in zip(tokens, pred_tags):
        print(f"{w:15s} -> {t}")
```

```
# Cell 6 (tiếp): Test câu ví dụ

test_sentence = "VNU University is located in Hanoi"
predict_sentence(model, test_sentence, word_to_ix, ix_to_tag, device)
```

```
Sentence: VNU University is located in Hanoi
Predictions:
VNU             -> B-PER
University      -> I-PER
is              -> O
located         -> O
in              -> O
Hanoi           -> B-LOC
```