CSCI544: Homework Assignment №4

Due on Nov 09, 2023 (before class)

Introduction

This assignment gives you hands-on experience in building deep learning models on named entity recognition (NER).
Here is a table of models that will be covered in this assignment

Model	Expected F1 dev
BiLSTM	77
BiLSTM with Glove	88
Transformer Encoder	61

Table 1: Models

Dataset

We will use the CoNLL-2003 corpus to build a neural network for NER. link: https://huggingface.co/datasets/conll2003 **It is convenient to use the datasets library to get this dataset**

```
import datasets

dataset = datasets.load_dataset("conll2003")
```

Use the convenient *.map*() function to prepossess your dataset seamlessly. It can be used to add keys, update keys

Example:

```
def convert_word_to_id(sample):
    # Code to convert all tokens to their respective indexes
    return {
        'input_ids': [
            word2idx[token]
            for token in sample['tokens']
        ]
    }

dataset.map(convert_word_to_id)
```

We added the input_ids column to this dataset.
Since we do not permit you to use 'pos_tags', 'chunk_tags', remove the following columns from the dataset: ['pos_tags', 'chunk_tags']
Rename the ner_tags to labels
The string values of NER tags are https://huggingface.co/datasets/conll2003#:~:text=%3A%2022%7D-,ner_tags,-%3A%20a%20list%20of

Glove Embeddings

We provide you with a file named *glove.6B.100d.gz*, which is the GloVe word embeddings [1]. Alternatively, you can download it from https://nlp.stanford.edu/data/glove.6B.zip
Download script for ipynb files:

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip glove.6B.zip
```

Evaluation

We will also provide you with an evaluation function 'evaluate' from: https://github.com/sighsmile/conlleval **Whenever we ask you for accuracy, precision, recall, or f1 score, we are referring you to use this**

script. If this script is not used, that will lead to a penalty
Download script for ipynb

```
!wget https://raw.githubusercontent.com/sighsmile/conlleval/master/conlleval.py
```

Usage:

```
# This is an example and the code will fail
# Because the preds is not the required length
from conlleval import evaluate
import itertools

# labels = ner_tags
# Map the labels back to their corresponding tag strings
labels = [
    list(map(idx2tag.get, labels))
    for labels in dataset['validation']['labels']
]
# This is the prediction by your model
preds = [
    ['O', 'O', 'B-ORG', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O'],
    ['B-LOC', 'O'],
    ...
    ...
    ...
]

precision, recall, f1 = evaluate(
    itertools.chain(*labels),
    itertools.chain(*preds)
)
```

Important: GPUs

Use Google Colab For the GPUs You do not necessarily need to write an ipynb file, a py file works on Google Colab by using the following trick:

1. Upload the .py file to Google Colab. Let's name it task1.py

2. pip install all requirements:

```
!pip install datasets accelerate
```

3. run the task1.py in a colab cell using this line:

```
!python task1.py
```

Alternatively, you can use Github to *push* → *clone* to Colab.

If you cannot use Google Colab anymore, then Kaggle gives you 30 hours a week worth of free GPUs (more than enough to complete this homework 15 times in a week). The free tier of Kaggle is better than Colab, but you can only use the github *push* → *clone* trick on Kaggle.

Task 1: Bidirectional LSTM model (40 points)

The first task is to build a simple bidirectional LSTM model for NER. Implementing the bidirectional LSTM network with PyTorch. The architecture of the network is:

$$\text{Embedding} \rightarrow \text{BiLSTM} \rightarrow \text{Linear} \rightarrow \text{ELU} \rightarrow \text{classifier}$$

There is no flattening, the linear layer is run for every single hidden output of the BiLSTM layer.
The hyper-parameters of the network are listed in the following table:

Layer hyperparam	value
Embedding dim	100
Num LSTM layers	1
LSTM hidden dim	256
LSTM Dropout	0.33
Linear output dim	128

Table 2: Layer Specification

Train this BiLSTM model with the training data on NER with any optimizer you like. Tune other parameters that are not specified in the above table, such as batch size, learning rate, and learning rate scheduling.

Additionally, kindly provide answers to the following questions:
What are the precision, recall, and F1 score on the validation data?.
What are the precision, recall, and F1 score on the test data?

Task 2: Using GloVe word embeddings (60 points)

Use the GloVe word embeddings to improve the BLSTM in Task 1. Helpful link. The way we use the GloVe word embeddings is straightforward: we initialize the embeddings in our neural network with the corresponding vectors in GloVe. Freeze the embeddings. **Note that GloVe is case-insensitive, but our NER model should be case-sensitive because capitalization is important information for NER**. You are asked to find a way to deal with this conflict.
You may use the same solution to boost the score for Task 1.

Additionally, kindly provide answers to the following questions:
What is the precision, recall, and F1 score on the validation data?
What are the precision, recall, and F1 score on the test data?
BiLSTM with Glove Embeddings outperforms the model without. Can you provide a rationale for this?

Bonus: The Transformer Encoder (40 points)

Transformer [2] is currently the most dominant architecture in NLP research.
Let's apply the transformer to the CoNLL-2003 dataset.

Build a BERT-like model by stacking nn.TransformerEncoderLayer
Use Transformer Encoder to stack the transformer encoder layers: nn.TransformerEncoder
Define Positional Embedding for the transformer.
Define Token Embedding for the transformer.
Now define a class for your transformer model that uses:

1. Positional Embedding
2. Token Embedding
3. Transformer Encoder Stack
4. Linear Layer as a classifier

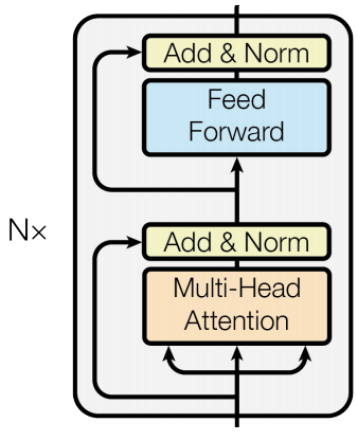

Figure 1: Transformer Encoder

Layer hyperparam	value
Embedding Size	128
num attention heads	8
sequence max length	128
feed-forward dimensions	128

Table 3: Transformer Specification

Do not forget to handle the *src_key_padding_mask*, it is very important for transformers.
The code from above links works when batch_first=False.
This means that the input to the transformer is of the dimension (sequence length, batch size, word vocab size)
This means that the output to the transformer is of the dimension (sequence length, batch size, tag vocab size)

Additionally, provide answers to the following questions:
What is the precision, recall, and F1 score on the validation data?
What are the precision, recall, and F1 score on the test data?
What is the reason behind the poor performance of the transformer?

Submission

Please follow the instructions and submit a zipped folder containing:

1. A PDF file that contains answers to the questions in the assignment along with a clear description of your solution, including all the hyper-parameters used in network architecture and model training.

2. Python code that only produces results on test data for task 1 (no training). You may need to save your models for this.

3. Python code that only produces results on test data for task 2 (no training). You may need to save your models for this.

4. README file to describe how to run your code to produce your results. In the README file, you need to provide the command line to produce the prediction files. (We will execute your cmd to reproduce your reported results on the test).

5. BONUS: The code you used.

References

[1] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.