ELL881 : Assignment 3

In this assignment, you will be building a named entity recognition (NER) model using a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model. NER is a subtask of information extraction that involves identifying and classifying named entities in text into predefined categories such as person names, organization names, locations, and more.

Broadly the steps involved will be as follows:

1. **Data Preparation**: You will process the dataset given to you and tokenize it.
2. **Fine-Tuning BERT**: You will fine-tune a pre-trained BERT model for sequence classification using the training set. You will use the Hugging Face Transformers library to load the pre-trained BERT model and customize the final layers for NER. You will also define the loss function, optimizer, and learning rate scheduler.
3. **Model Evaluation**: You will evaluate the performance of the trained model using the test set on the accuracy metric.

Dataset description

```
import pandas as pd
df = pd.read_csv("ner.csv")
df.head()
```

```
                                                text
0  Thousands of demonstrators have marched throug...  \
1  Iranian officials say they expect to get acces...
2  Helicopter gunships Saturday pounded militant ...
3  They left after a tense hour-long standoff wit...
4  U.N. relief coordinator Jan Egeland said Sunda...

                                              labels
0  O O O O O O B-geo O O O O O B-geo O O O O O B-...
1  B-gpe O O O O O O O O O O O O O B-tim O O O ...
2  O O B-tim O O O O O B-geo O O O O O B-org O O ...
3                            O O O O O O O O O O O
4  B-geo O O B-per I-per O B-tim O B-geo O B-gpe ...
```

The labels in this dataset are as follows:

- `geo` for geographical entity
- `org` for organization entity
- `per` for person entity
- `gpe` for geopolitical entity
- `tim` for time indicator entity
- `art` for artifact entity
- `eve` for event entity

- `nat` for natural phenomenon entity
- `O` is assigned if a word doesn't belong to any entity.

The labels have also been tagged using the BIO scheme. You can use the following code for getting the list of labels.

```
labels = [i.split() for i in df['labels'].values.tolist()]
unique_labels = set()
for lb in labels:
    [unique_labels.add(i) for i in lb if i not in unique_labels]

print(unique_labels)
```

```
{'O', 'B-eve', 'B-art', 'I-eve', 'B-geo', 'B-per', 'B-nat', 'I-geo',
'I-org', 'I-gpe', 'B-tim', 'I-art', 'I-per', 'B-org', 'I-nat', 'I-
tim', 'B-gpe'}
```

```
# We split the data into train, validation and test sets (80-10-10
split)
import numpy as np
df_train, df_val, df_test = np.split(df.sample(frac=1,
random_state=42),
                                     [int(.8 * len(df)), int(.9 * len(df))])
```

Model Building

In this assignment, you will be using a pretrained BERT model from HuggingFace (supplied in the `transformers` library). This is a classification task hence the model that you should make use of `BertForTokenClassification` model.

You can train the model using GPU. You should ideally get the script ready on your system by taking a small subset of data and then train it completely using an online service such as Google Colab or Kaggle.

Further, the model expects the inputs to supplied in a particular format which you should be able to read online in the documentations and other resources like medium articles (read up on using BERT for NLP tasks in Pytorch and you will find a lot of resources online). Additionally, for performing tokenization you should be using the tokenizer supplied in the transformers library. The imports for the same have been done in the code snippet below:

```
from transformers import BertTokenizerFast
from transformers import BertForTokenClassification

tokenizer = BertTokenizerFast.from_pretrained('bert-base-cased')
# Note how we are using the cased version of tokenizer here since the
labels leverage the case information


class BertModel(torch.nn.Module):
```

```
    def __init__(self):
        super(BertModel, self).__init__()
        self.bert = BertForTokenClassification.from_pretrained('bert-
base-cased', num_labels=len(unique_labels))

    def forward(self, input_id, mask, label):
        output = self.bert(input_ids=input_id, attention_mask=mask,
labels=label, return_dict=False)
        return output
```

Evaluation

For evaluating the performance of the model, you should make use of the `accuracy` metric. You should report the performance after removing the pads. Further, it would be better if you report the accuracy both with and without the "O" label.

Your final submission should include a report that describes your methodology, experimental results, analysis, and discussion, as well as the code used to train and test the model.

Good luck, and have fun exploring BERT!