DUE DATE: DEC/18/2024 23:59

BERT-based model fine-tuning (5 Points)

1 Coding: fine-tuning your own model

Instruction: Please submit an ipynb file named fine_tuning.ipynb via LMS.
BERT[1] is a transformer encoder based pre-trained model released by Google. Through the process of pre-training, BERT acquires general knowledge of language which can be transferred to specific downstream tasks such as natural language understanding, question answering, and many more. This makes BERT a great foundation model for building task-specific models that suits your own needs.
In this assignment, you will be using a variant of BERT to make your own fine-tuned model. By writing your own code to train and evaluate a model you will be able to gain experience with the underlying procedure involved in building NLP systems that can actually be applied to real-world scenarios.

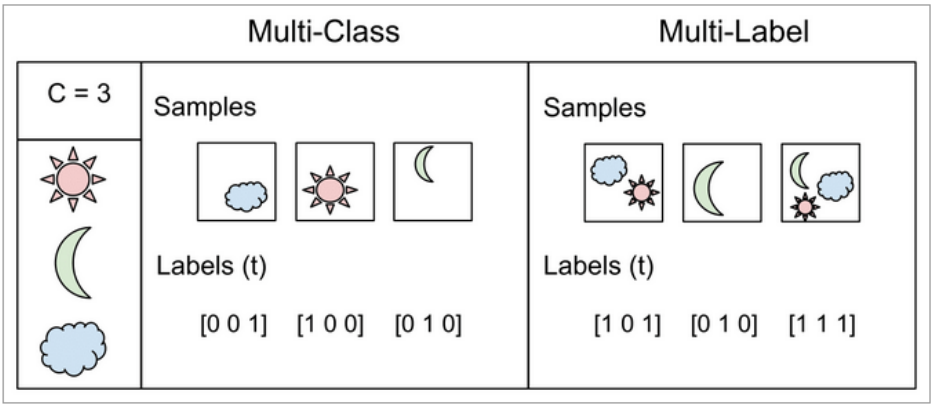

Figure 1: Multi-class (single-label) and multi-label classification

Task: The target task for this assignment is multi-intent detection(MID), which is a type of multi-label classification. Multi-label classification assumes a setting where multiple classes (or labels) can be assigned to a single input instance. Multi-label classification is much more challenging compared to single-label classification because the model is required to predict multiple labels and get them all correct at the same time. (In this assignment, you will be tackling a simplified version of MID where the number of labels is always two.)

[1] https://arxiv.org/pdf/1810.04805

Homework 4

Detailed Requirements: Make a multi-intent detection model based on BERT[2] (or any other BERT-like model of your choice) with the given dataset, HW4.json (check the "Dataset specifications" at the fifth page of this document for details).
Tip: You are probably going to need a GPU for this assignment. If you don't have your own GPU, use Colab!

- Choose a model to use as a foundation for fine-tuning. It should be a transformer encoder based pre-trained model best represented by BERT (Some other examples are DistilBERT, RoBERTa, DeBERTa). **Make sure the model you are using is not a fine-tuned model!** (You will **receive 0 points** for this assignment if you use a fine-tuned model.)

- Now let us remember what we learned from the Huggingface tutorial! First things first. You should import the necessary libraries such as transformers and datasets. Next, you should load your model and data from the Huggingface hub.

- Second, you are also going to need some pre-processing before you can use the given data for training. Make sure to use the "train" split for training and the "dev" split for evaluation. (You can make customizations to your training data if you want.)

- Before you can actually train a model, you are going to need a baseline. Try applying your untrained model to the "dev" split to compute the accuracy. Also, write a compute_metrics function to be used for training.

- Training can be done simply by setting some necessary training arguments (learning rate, number of epochs) and passing it to the trainer. However, the Huggingface trainer offers many more fascinating features and neat tricks to help you train a stronger model more efficiently. Give them a try to find out what works best for you.

- Once training is done you should evaluate your model. Compute the accuracy for your fine-tuned model the same way you did on the untrained model! And finally, **upload your model to the Huggingface hub.** (Hugging Face will automatically generate a "model card" for your model. Check if it includes details about the base model, hyperparameters, and training results.)

[**Note:** The grade for this assignment will be **heavily influenced by the performance** of your resulting model. We expect you to **do whatever you possibly can to make a better performing model.** Also, remember that you should be able to explain all the efforts and considerations you put into the training process in the written report.]

[2] https://huggingface.co/docs/transformers/model_doc/bert

2 Written: writing a report on your work

Instruction: Please submit a pdf file named report.pdf via LMS.

Write a report that describes all the hard work and effort you put into your fine-tuning. **Please include screenshots of your code or resulting outputs to enrich your explanations.**

Detailed Requirements: Your submission must include the following information.

- **Understanding the task:** Explain what intent detection is. Why do you think such task is necessary and in what kind of scenarios do you think it is going to be useful?
- **Model information:** Explain the model of your choice along with reasons why you decided to use that particular model.
- **Training tactics:** Describe all the possible details regarding the strategies and considerations put into the process of training. For example, if you decided to train your model for N epochs then make an explanation for that (e.g. "Epochs lower than N resulted in under-fitting and epochs higher than N resulted in over-fitting"). If there are any tricks you found from external sources that you applied to your own training, then cite the source and explain what that trick is.
- **Trial and error:** Show all of your efforts, failures and hard work within this report! If you ran into errors before you were able to finally get some training done, explain what you did to fix them. **Include all of your progress, not just the final results!!!**
- **Resulting model:** Include a link to your fine-tuned model uploaded on the Huggingface hub. **(Very important!!!** Don't miss this part out or you will get a 0.)

How to Submit: Please upload your code and report as two separate files through LMS.[3] Do **not** compress them into a single zip file.

[3] https://lms.hanyang.ac.kr

3 A few tips regarding multi-label classification models

3.0.1 How to load a model from the Huggingface hub as a multi-label classification model

```
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained(
    "bert-base-uncased", # your model
    problem_type="multi_label_classification",
    num_labels=2 # number of labels
)
```

3.0.2 Evaluating a multi-label classification model

When evaluating your multi-label classification model, you should also consider the order of your predictions. Simply comparing the prediction of your model with the labels based on exact-match may result in certain errors like the following:

```
predictions = ['B','A']
labels = ['A','B']

print(predictions == labels) # this is going to be False!
```

3.0.3 Loading and processing a json file using the datasets library

The given dataset for this assignment is a json file. This means we cannot load it using the load_dataset function the way do for datasets shared via the Huggingface hub. However, the datasets library also offers ways to load data from local files including json, csv and etc. This function will result in a "Dataset" object[4] that you can easily process and adjust using the built-in functions[5].

```
from datasets import Dataset

data = Dataset.from_json("HW4.json") # this will result in a "Dataset" object.
```

[4] https://huggingface.co/docs/datasets/v3.1.0/en/package_reference/main_classes#datasets.Dataset

[5] https://huggingface.co/docs/datasets/v3.1.0/process

4 Dataset specifications

The dataset HW4.json used for this assignment is based on **BlendX**[6], a result of a research conducted at our lab! **BlendX** mainly focuses on the task of multi-intent detection, an expanded version of intent detection where we assume there are multiple intents in our input utterance. While the actual **BlendX** dataset covers instances that can have varying number of intents between 1 to 3, the dataset for this assignment only includes instances where there are 2 intents for simplicity.

Here is an example with three actual rows from the data:


```
{"split":"train",
"utterance":"include this song in the kids playlist, help me find my luggage",
"intent":"['lost_luggage', 'update_playlist']"}
{"split":"train",
"utterance":"where were you manufactured , what's my pay for this week",
"intent":"['income', 'where_are_you_from']"}
{"split":"dev",
"utterance":"what's my credit limit, and can i increase it",
"intent":"['credit_limit', 'credit_limit_change']"}
```


- **Split:** Each row includes a "split" value that marks which split the row belongs to. If the value is "train", that row is meant to be used for training. If the value is "dev"(short for development), that means the particular row is to be used for validation. The dataset consists of 28,815 rows for training and 1,513 rows for validation.
- **Utterance:** The "utterance" will be the input to your model. It is a single sentence that includes two intentions at once. For example, the utterance from the first row above is a combination of "include this song in the kids playlist" and "help me find my luggage", which can be mapped to intents "update_playlist" and "lost_luggage."
- **Intent:** The "intent" value is the label for that instance, the two intents that can be found from the given "utterance."

[6] https://arxiv.org/pdf/2403.18277