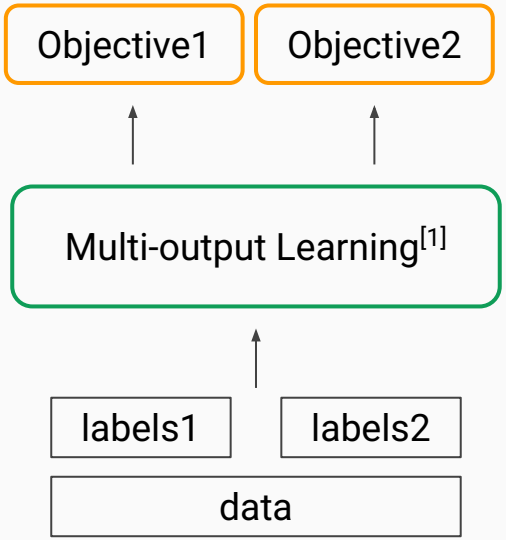
Assignment 3: Multi-output learning

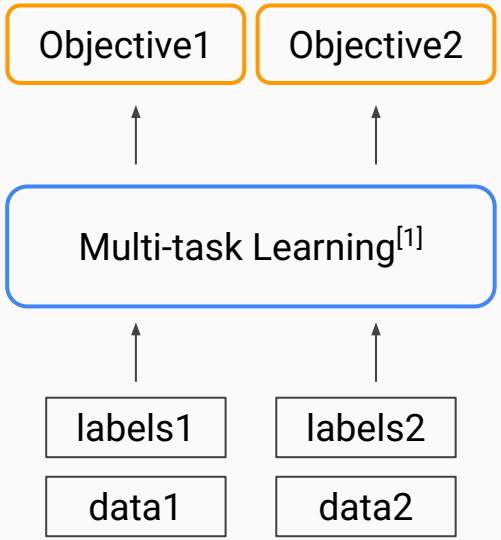
2025 NTHU Natural Language Processing

Hung-Yu Kao

IKM Lab TAs

Multi-output Learning





[1] Xu, Donna, et al. "Survey on multi-output learning." IEEE transactions on neural networks and learning systems 31.7 (2019): 2409-2429.

Assignment Description

- In Assignment 3, you will practice building a model for multi-output learning.
- You are going to use the **SemEval 2014 Task 1** dataset, where each data point is associated with multiple labels.

Dataset examples (SemEval 2014)

*Answer in red

Value: 1-5

'0': NEUTRAL
'1': ENTAILMENT
'2': CONTRADICTION

sub-task1
regression

sub-task2
3-class classification

premise	hypothesis	relatedness_score	entailment_judgement
A group of kids is playing in a yard and an old man is standing in the background	A group of boys in a yard is playing and a man is standing in the background	4.5	0
A man, a woman and two girls are walking on the beach	A group of people is on a beach	4.300000190734863	1

[1] SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment, SemEval 2014

Dataset

*SemEval 2014 Task1 dataset

- Train split: 4500 pieces
- Validation split: 500 pieces
- Test split: 4927 pieces
- Each data piece: A premise, a hypothesis, a relatedness_score, an entailment_judgement

[1] SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment, SemEval 2014

Code

Import packages

```
from transformers import BertTokenizer, BertModel
from datasets import load_dataset
from evaluate import load
import torch
from torch.utils.data import Dataset, DataLoader
from torch.optim import AdamW
from tqdm import tqdm
device = "cuda" if torch.cuda.is_available() else "cpu"
#  You can install and import any other libraries if needed
```

You can install and import any other libraries if needed

Load the dataset from Hugging Face

You may cache the dataset at any location you prefer.

```python
class SemevalDataset(Dataset):
    def __init__(self, split="train") -> None:
        super().__init__()
        assert split in ["train", "validation", "test"]
        self.data = load_dataset(
            "sem_eval_2014_task_1", split=split, trust_remote_code=True, cache_dir="./cache/"
        ).to_list()

    def __getitem__(self, index):
        d = self.data[index]
        # Replace Chinese punctuations with English ones
        for k in ["premise", "hypothesis"]:
            for tok in token_replacement:
                d[k] = d[k].replace(tok[0], tok[1])
        return d

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

data_sample = SemevalDataset(split="train").data[:3]
print(f"Dataset example: \n{data_sample[0]} \n{data_sample[1]} \n{data_sample[2]}")
```

You need datasets==2.21.0 to download **sem_eval_2014_task_1** from Hugging Face!

← Observe the data instances

TODO1: Create batched data with PyTorch DataLoader

```python
# TODO1: Create batched data for DataLoader
# `collate_fn` is a function that defines how the data batch should be packed.
# This function will be called in the DataLoader to pack the data batch.

def collate_fn(batch):
    # TODO1-1: Implement the collate_fn function
    # Write your code here
    # The input parameter is a data batch (tuple), and this function packs it into tensors.
    # Use tokenizer to pack tokenize and pack the data and its corresponding labels.
    # Return the data batch and labels for each sub-task.

# TODO1-2: Define your DataLoader
dl_train = # Write your code here
dl_validation = # Write your code here
dl_test = # Write your code here
```

```
print(next(iter(dl_train))
```

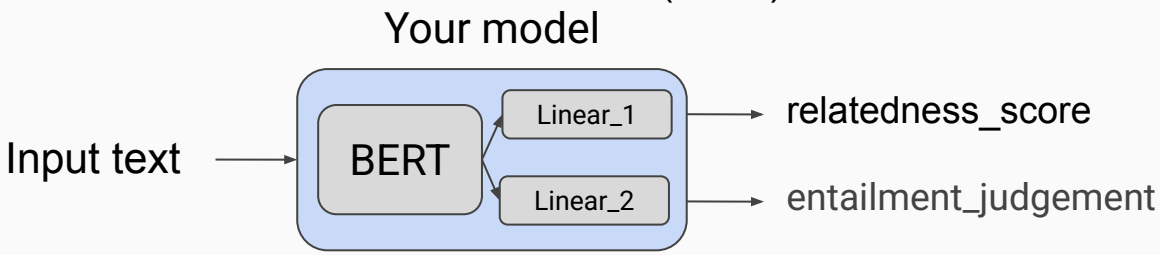
→

1. input_text (input_ids, token_type_ids, attention_mask)
2. labels1 (regression)
3. labels2 (classification)

TODO2: Construct your model

```python
# TODO2: Construct your model
class MultiLabelModel(torch.nn.Module):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        # Write your code here
        # Define what modules you will use in the model
        # Please use "google-bert/bert-base-uncased" model (https://huggingface.co/google-bert/bert-base-uncased)
        # Besides the base model, you may design additional architectures by incorporating linear layers, activation functions,
        # Remark: The use of any additional pretrained language models is not permitted.
    def forward(self, **kwargs):
        # Write your code here
        # Forward pass
```

You are required to build a model (using huggingface API) to handle the multi-output task.
Please use bert-base-uncased model (LINK).



An example of model achitecture (BERT model)

TODO3: Define optimizer, loss fuction, and dataloader

```
# TODO3: Define your optimizer and loss function

model = MultiLabelModel().to(device)
# TODO3-1: Define your Optimizer
optimizer = # Write your code here

# TODO3-2: Define your loss functions (you should have two)
# Write your code here

# scoring functions
psr = load("pearsonr")
acc = load("accuracy")
```

- We recommend using Adam or AdamW as the optimizer.
- For the loss functions, observe the type of each sub-task; use different loss functions for different types of tasks.

TODO4: Write the training loop

- Train your model using PyTorch gradient descend.
  - This time, you cannot use `HuggingFace Trainer`
- Compute gradient in each training step, and optimize the model
- The loss value is an aggregation of the losses from all sub-tasks.

```
best_score = 0.0
for ep in range(epochs):
    pbar = tqdm(dl_train)
    pbar.set_description(f"Training epoch [{ep+1}/{epochs}]")
    model.train()
    # TODO4: Write the training loop
    # Write your code here
    # train your model
    # clear gradient
    # forward pass
    # compute loss
    # back-propagation
    # model optimization
```

TODO5: Evaluate your model

```
15      pbar = tqdm(dl_validation)
16      pbar.set_description(f"Validation epoch [{ep+1}/{epochs}]")
17      model.eval()
18      # TODO5: Write the evaluation loop
19      # Write your code here
20      # Evaluate your model
21      # Output all the evaluation scores (PearsonCorr, Accuracy)
22      pearson_corr = # Write your code here
23      accuracy = # Write your code here
24      # print(f"F1 Score: {f1.compute()}")
25
26      if pearson_corr + accuracy > best:
27          best = pearson_corr + accuracy
28          torch.save(model.state_dict(), f'./saved_models/best_model.ckpt')
```

- Evaluate the model on Validation set.
- For each instance, there are multiple labels for each sub-tasks. The evaluation result includes scores of all sub-tasks.
  - For relatedness_score, compute the Pearson correlation coefficients. For entailment_judgement, compute accuracy.
  - The sample code use torchmetrics package to compute scores.

TODO6: Test your model

```python
# Load the model
model = MultiLabelModel().to(device)
model.load_state_dict(torch.load(f"./saved_models/best_model.ckpt", weights_only=True))

# Test Loop
pbar = tqdm(dl_test, desc="Test")
model.eval()

# TODO6: Write the test loop
# Write your code here
# We have loaded the best model with the highest evaluation score for you
# Please implement the test loop to evaluate the model on the test dataset
# We will have 10% of the total score for the test accuracy and pearson correlation
```

Load the model checkpoint that achieves the best evaluation score and run it on test set. Then compute the test Pearson correlation coefficients and accuracy

Scoring

Coding work : 40%

TODOs	Scores
TODO1: Create batched data with PyTorch DataLoader	5%
TODO2: Construct your model	5% (Your submitted code must use the **bert-base-uncased** model; otherwise, you will not receive this 5%)
TODO3: Define optimizer, loss fuction, and dataloader	5%
TODO4: Write the training loop	5%
TODO5: Evaluate your model	10%
TODO6: Test your model	10%

Submission

Scoring

Coding work : 40%

Score: 10% (The higher the accuracy achieved, the higher the score awarded.)

- A screenshot of testing log and accuracy must be included at the end of the report.
- Baseline: (Pearson 0.8, Accuracy 0.8, **test** set, 4%)

Report: 50%

- Train a RoBERTa-base model on the same dataset and report the performance of both BERT-base and RoBERTa-base models. (10%) Discuss the main differences between BERT and RoBERTa, and explain how these differences may affect performance. (5%)
- Train a GPT-2 model on the same dataset and evaluate its performance. (10%) What are the main difference of BERT and GPT2? Discuss the main differences between BERT and GPT-2, and analyze how these differences influence their performance. (5%)
- Compare the performance of a multi-output learning model with BERT models trained separately on each sub-task. Does multi-output learning improve performance? Explain why or why not. (5%)
- Why does your model fail to correctly predict some data points? Please provide an error analysis and introduce how do you improve your performance. (10%)
- **… Anything that can strengthen your report.** (5%)

Delivery policies: File formats

- Coding work: Python file (.py)
  - Download your script via Colab.
- Package list: requirements.txt
  - E.g., numpy==1.26.3
- Report: Microsoft Word (.docx)
- No other formats are allowed.
- Zip the files above before uploading you assignment.

Delivery policies: Filenames

	Filename rule	**Filename example**
Coding work	NLP_HW3_school_student_ID.py	NLP_HW3_NTHU_12345678.py
Report	NLP_HW3_school_student_ID.docx	NLP_HW3_NTHU_12345678.docx
Package list	requirements.txt	
Zipped file	NLP_HW3_school_student_ID.zip	NLP_HW3_NTHU_12345678.zip

Delivery policies: Things You should include

- In your report:

	Example	
Environment types	**If Colab or Kaggle**	**If local**
Running environment	Colab	System: Ubuntu 22.04, CPU: Ryzen 7-7800X3D
Python version	Colab	Python 3.10.1
GPU(s) you used	Please check the info from Colab	NVIDIA RTX 3090 * 1

How to check the allocated GPU on Colab?

Type in a code block: `!nvidia-smi`

Delivery policies: Rules of coding

- If you use ChatGPT or Generative AI, please specify your usage **both** in:
  - **Code comments**
  - **Reports**
- **No plagiarism**. You should not copy and paste from your classmates. **Submit duplicate code or report will get 0 point !**
- Please provide links if you take the code from the Internet as reference.
- The following behaviors will cause loss in the score of the assignment: **(1) Usage with Generative AI without specifications (2) Internet sources without specifications (3) Plagiarism.**

Punishments

Rule	Name your code: NLP_HW3_school_student_ID.py (only .py is acceptable)	Name your report: NLP_HW3_school_student_ID.docx	Name your file: NLP_HW3_school_student_ID.zip	Include requirements.txt
Punishment	-5	-5	-5	-5
Rule	Include python version in your report	Do not modify the code template (only changes to data loading are allowed).	Do not modify the report template	Your code or report should not shows a high degree of similarity to another student's submission.
Punishment	-5	-5	-5	-100 for both

If you are using Colab, go to **File** → **Download** → **Download .py** to obtain the Python file.

Uploading the zipped file

- Please upload your file to NTU COOL.
- You will have three weeks to finish this assignment.
- If you have any question, please e-mail to **nthuikmlab@gmail.com**