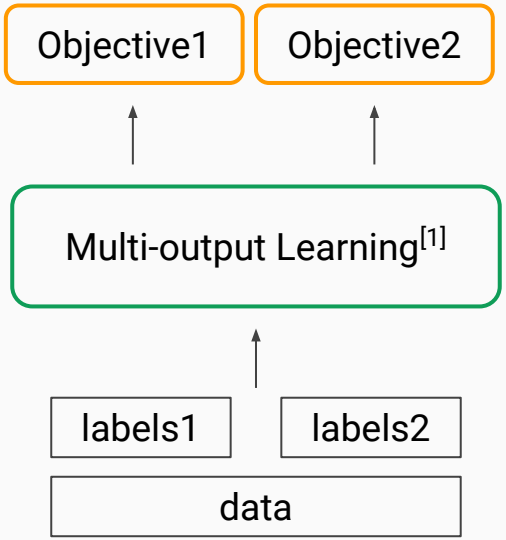
Assignment 3: Multi-output learning

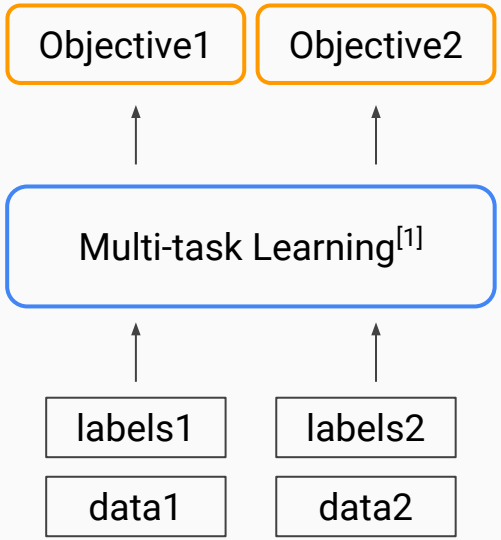
2024 NTHU Natural Language Processing

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

sub-task1
regression

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

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 (hints only)

Load the dataset from Hugging Face

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

    def __getitem__(self, index):
        d = self.data[index]
        # 把中文標點替換掉
        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
```

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

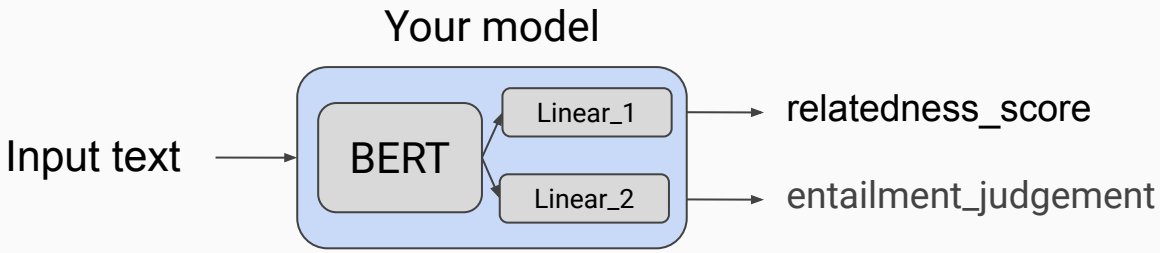
→

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
    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.



An example of model achitecture (BERT model)

TODO3: Define optimizer, loss fuction, and dataloader

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

# 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
spc = SpearmanCorrCoef()
acc = Accuracy(task="multiclass", num_classes=3)
f1 = F1Score(task="multiclass", num_classes=3, average='macro')
```

- 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.

```python
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

```
14      pbar = tqdm(dl_validation)
15      pbar.set_description(f"Validation epoch [{ep+1}/{epochs}]")
16      model.eval()
17      # TODO5: Write the evaluation loop
18      # Write your code here
19      # Evaluate your model
20      # Output all the evaluation scores (SpearmanCorrCoef, Accuracy, F1Score)
21      torch.save(model, f'./saved_models/ep{ep}.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 and Spearman correlation coefficients. For entailment_judgement, compute accuracy and macro F1 score.
  - The sample code use torchmetrics package to compute scores.

Submission

Scoring

Coding work : 40% (8% for each of the five TODOs)

Baseline: 10% (SpearmanCorrCoef 0.71, Accuracy 0.85, **test** set)

Bonus: 10% (Based on the score of the **test** set, only **BERT-base** can be used)

- **SpearmanCorrCoef** over 0.74: 3%, Over 0.77: 7%, Over 0.8: 10%

Report: 40%

- Which (pre-trained) model do you use? Why to choose the model? (5%)
- Compared with models trained separately on each of the sub-task, does multi-output learning improve the performance? (8%)
- Why does your model fail to correctly predict some data points? Please provide an error analysis. (8%)
- How do you improve your model performance? (9%)
- **… Anything that can strengthen your report.** (10%)

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.**

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**

Punishments

Rules	Name your code as NLP_HW3_school_student_ID.py (only .py is allowed!)	Name your code as NLP_HW3_school_student_ID.docx (only .docx is allowed!)	Include `requirements.txt` in the folder	Zip your files into NLP_HW3_school_student_ID.zip (only .zip is allowed!)	Include the running envrionment in your report
Punishment	-5	-5	-5	-5	-5
Rules	Include the running envrionment in your report	Include the Python version in your report	Include the GPU card(s) you use in your report	Usage with GAI or Internet sources with proper specifications	Do not copy and paste the code from your classmates
Punishment	-5	-5	-5	-100	-100 (for both)