

# Applied Deep Learning HW3

## Natural Language Generation

Deadline: 2022/11/30 23:59:59

# Change Logs

---

- 11/4 Update deadline to 11/30
- package updated:
  - `protobuf == 3.20.x`

# Links

— — —

NTU COOL

[Data & Evaluation](#)

說明影片

[adl-ta@csie.ntu.edu.tw](mailto:adl-ta@csie.ntu.edu.tw)

TA Hours:

Tue. 14:00~15:30 @ [Google Meet](#)

Thu. 14:00~15:30 @ [Google Meet](#)

# Task Description

# Chinese News Summarization (Title Generation)

---

❖ input: news content

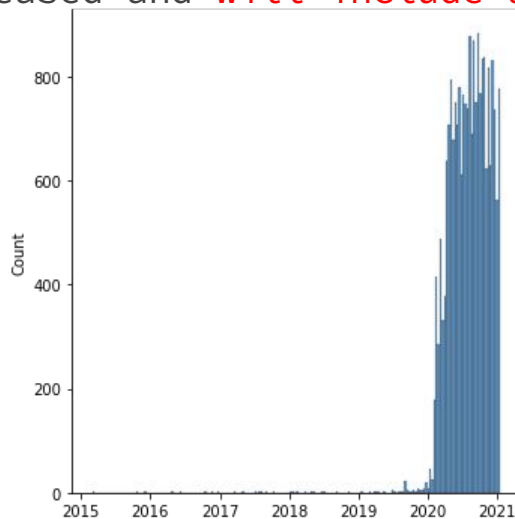
從小就很會念書的李悅寧，在眾人殷殷期盼下，以榜首之姿進入臺大醫學院，但始終忘不了對天文的熱情。大學四年級一場遠行後，她決心遠赴法國攻讀天文博士。從小沒想過當老師的她，再度跌破眾人眼鏡返台任教，

❖ output: news title

榜首進台大醫科卻休學、27歲拿到法國天文博士 李悅寧跌破眾人眼鏡返台任教

# Data

- ❖ Source: news articles scraped from udn.com
  - Train: 21710 articles from 2015-03-02 to 2021-01-13
  - Public: 5494 articles from 2021-01-14 to 2021-04-10
  - Private: Not released and will include articles after deadline



# Data (cont.)

— — —

## ❖ Example

```
1 {  
2   'date_publish': '2015-03-02 00:00:00',  
3   'title': '榜首進台大醫科卻休學 、27歲拿到法國天文博士 李悅寧跌破眾人眼鏡返台任教',  
4   'source_domain': 'udn.com',  
5   'maintext': '從小就很會念書的李悅寧， 在眾人殷殷期盼下，以榜首之姿進入臺大醫學院， 但始終忘不了對天文的熱情。...'  
6 }
```

# Metrics

— — —

## ❖ ROUGE score with chinese word segmentation

- [What is ROUGE score?](#)
- Chinese word segmentation: [ckiptagger\(github\)](#)

## ❖ Example

- candidate: 我 是 人
- reference: 我 是 一 個 人
- rouge-1: precision=1.0, recall=0.6, f1=0.75
- rouge-2: precision=0.5, recall=0.25, f1=0.33
- rouge-L: precision=1.0, recall=0.6, f1=0.75



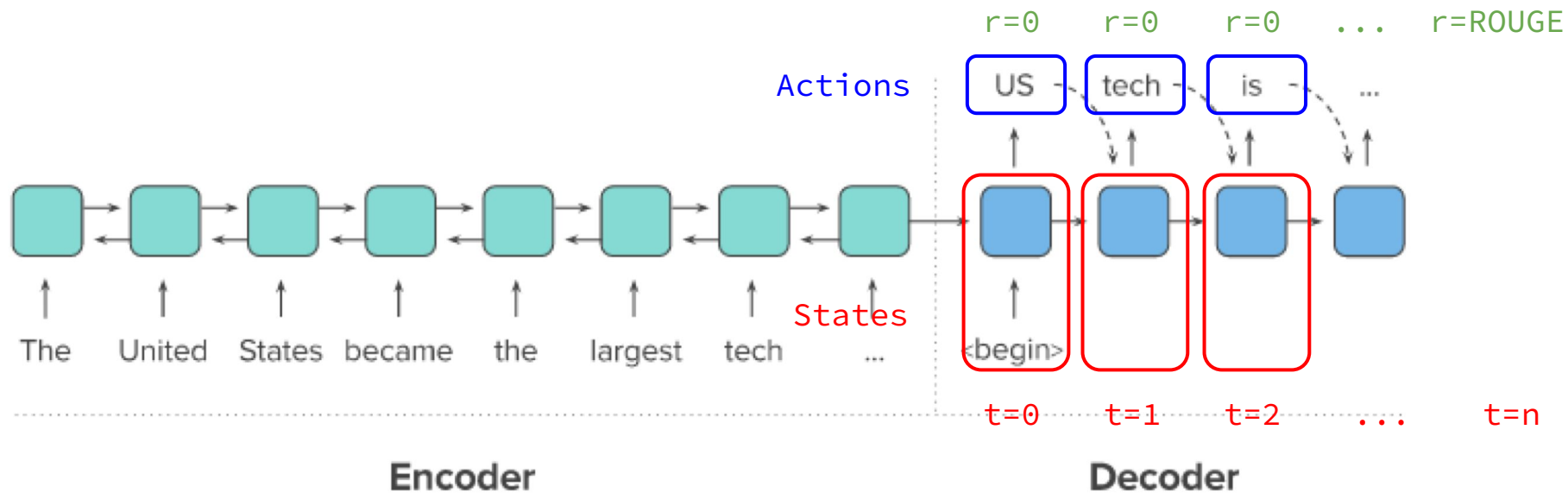
# Objective

— — —

- ❖ Fine-tune a pre-trained [small multilingual T5](#) model to pass the baselines
- ❖ Public baseline
  - rouge-1: 22.0, rouge-2: 8.5, rouge-L: 20.5 (f1-score \* 100)
- ❖ Private baseline
  - Will be announced after deadline

# Bonus: Applied RL on Summarization

---



## Bonus: Applied RL on Summarization (cont.)

— — —

- ❖ You can use any RL algorithms (policy gradient, DQN and etc.)
- ❖ You can design your own reward function
  - e.g. ROUGE-L, avg(ROUGE-N) and etc.
- ❖ You can either directly add RL loss while training or fine-tune from a supervised-learning checkpoint

# Report

# Q1: Model (2%)

— — —

## ❖ Model (1%)

- Describe the model architecture and how it works on text summarization.

## ❖ Preprocessing (1%)

- Describe your preprocessing (e.g. tokenization, data cleaning and etc.)

## Q2: Training (2%)

---

- ❖ Hyperparameter (1%)

- Describe your hyperparameter you use and how you decide it.

- ❖ Learning Curves (1%)

- Plot the learning curves (ROUGE versus training steps)

# Q3: Generation Strategies(6%)

— — —

## ❖ Strategies (2%)

- Describe the detail of the following generation strategies:
  - Greedy
  - Beam Search
  - Top-k Sampling
  - Top-p Sampling
  - Temperature

## ❖ Hyperparameters (4%)

- Try **at least 2 settings of each strategies** and compare the result.
- What is your final generation strategy? (you can combine any of them)

# Bonus: Applied RL on Summarization (2%)

---

## ❖ Algorithm (1%)

- Describe your RL algorithms, reward function, and hyperparameters.

## ❖ Compare to Supervised Learning (1%)

- Observe the loss, ROUGE score and output texts, what differences can you find?



# Rules

# What You Can Do

---

## ❖ Allowed packages/tools:

- Python 3.8 / 3.9 and Python Standard Library
- PyTorch 1.12.1, TensorFlow 2.10.0
- transformers, datasets, accelerate, sentencepiece
- rouge, spacy, nltk, ckiptagger, tqdm, pandas, jsonlines
- Dependencies of above packages/tools.
- No Network access after we used download.sh

## ❖ If you want to use other package, mail TA.

## ❖ You can use any package you want when writing report.

# What You Can **NOT** Do

— — —

- ❖ Use external training data
  - E.g. scrape news from the internet
- ❖ Any means of cheating or plagiarism, including but not limited to:
  - Use other classmates' published / unpublished code., including students who took previous ML / ADL / MLDS.
  - Just copy and past any public available code without modification
  - Use package or tools not allowed.
  - Give/get trained model to/from others.
  - Give/get report answers or plots to/from others.
  - Publish your code before deadline.
- ❖ Violation may cause zero/negative score and punishment from school.

# Logistics

# Grading

— — —

- ❖ Model performance (10%)

- Public baseline (5%)
- Private baseline (5%)

- ❖ Report (10% + 2%)

- In PDF format!
- Score of each problem is shown in the [Report section](#).

- ❖ Format

- You may lose (some or all) of your model performance score if your script is at wrong location, causes any error, etc.

# Submission - Format

sample\_submission.jsonl

```
1 {'title': 'Anker新款真無線藍牙耳機Liberty Air 2 Pro 引進台灣市場', 'id': '21710'}
2 {'title': '藍染、客家美食、舊山線自行車 「苗栗一日遊」超人氣美食美景', 'id': '21711'}
3 {'title': '華碩打造對應軍規防護與2 in 1設計的15.6吋Chromebook', 'id': '21712'}
4 {'title': '產業發展變革 台灣的優勢與機會', 'id': '21713'}
5 {'title': '全球Windows 7裝置粗估至少還有1億台以上 市佔率穩穩卡在20%', 'id': '21714'}
6 {'title': '強勢台幣理財攻略', 'id': '21715'}
7 {'title': '「不需治療，只需到台灣！」 美國「哈台馬克杯」賣到缺貨', 'id': '21716'}
```

# Submission - File Layout

— — —

- ❖ You are required to submit **.zip** file to NTU Cool
- ❖ File structure for the **.zip** file (case-sensitive):
  - `/[student id (lower-cased)]/` (Brackets not included.)
    - `download.sh`
    - `run.sh`
    - `README.md`
    - **`report.pdf`**
    - `code/all other files you need`

# Submission - Scripts

— — —

## ❖ **download.sh**

- Do not modify your file after deadline, or it will be seen as cheating.
  - Keep the URLs in **download.sh** valid for at least 2 weeks after deadline.
  - Do not do things more than downloading. Otherwise, your **download.sh** may be killed.
  - You can download at most 4G, and **download.sh** should finish within 1 hour. (At csie dept with maximum 10MB/s bandwidth)
- ❖ You can upload your model to [Dropbox](#). (see [tutorial](#))
- ❖ We will execute **download.sh** before predicting scripts.



# Submission - Scripts

— — —

- ❖ **run.sh**
- ❖ Arguments:
  - `${1}`: path to the input file
  - `${2}`: path to the output file
- ❖ TA will predict testing data as follow:
  - `bash ./download.sh`
  - `bash ./run.sh /path/to/input.jsonl /path/to/output.jsonl`
- ❖ Specify the Python version (3.8 or 3.9) in the .sh file.
  - Default python version would be 3.8
  - Ex. `python3.8 predict.py ... / python3.9 predict.py ...`  
“python” would be python3.8
- ❖ **Make sure your code works!**

# Submission - Reproducibility

---

- ❖ All the code you used to train, predict, plot figures for the report should be upload.
- ❖ We will remove the answers in `public.jsonl` when we reproduce your submission.
- ❖ README.md
  - Write down how to train your model with your code/script specifically.
  - If necessary, you will be required to reproduce your results based on the README.md.
  - If you cannot reproduce your result, you may lose points.
- ❖ You will get at least - 2 penalty if you have no or empty README.md.

# Execution Environment

---

- ❖ Will be run on computer with
  - Ubuntu 20.04
  - 32 GB RAM, GTX 3070 **8G** VRAM, 20G disk space available.
  - the packages we allow only.
  - python 3.8 / 3.9
- ❖ Do NOT train with very large model (e.g. mt5-xl) or you will get an out of memory error on 8G VRAM.
- ❖ Time limit: 1 hours for **run.sh** in total
- ❖ No network access when predicting.
- ❖ You will lose (some or all) your model performance score if your script is at wrong location, or cause any error.

# Late Submission Penalty

---

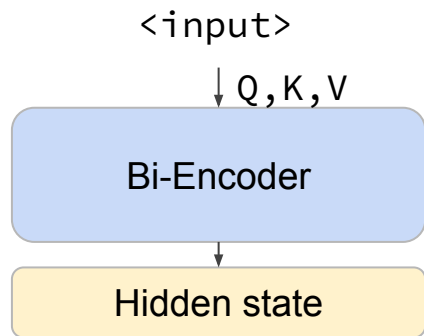
- ❖ Late submission of "code and report":
  - 0 day < late submission  $\leq$  1 day: original score \* 0.95
  - 1 day < late submission  $\leq$  3 day: original score \* 0.90
  - 3 day < late submission  $\leq$  4 day: original score \* 0.75
  - 4 day < late submission  $\leq$  5 day: original score \* 0.50
  - 5 day < late submission  $\leq$  6 day: original score \* 0.25
  - 6 day < late submission: original score \* 0.00
- ❖ Late submission is determined by the last submission.
  - Update your submission after deadline implies that you will get penalty.

# Guide

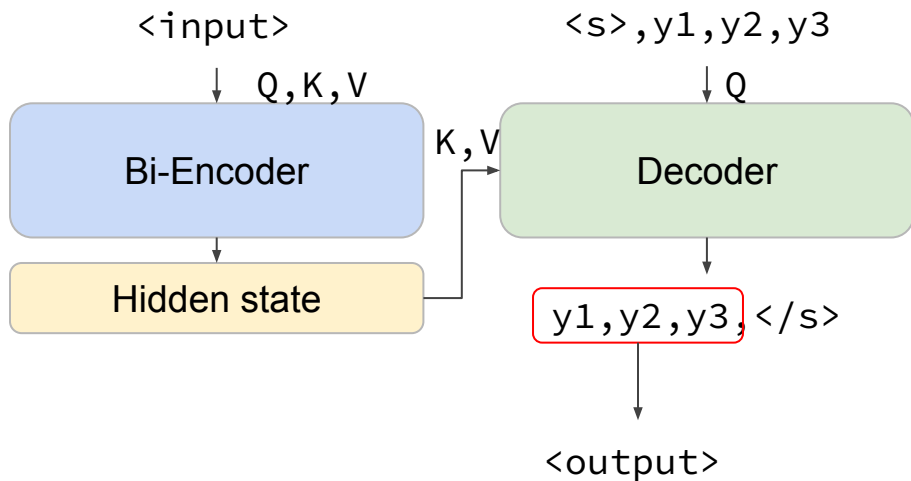
# Text-to-Text Transformer (T5)

— — —

HW2: BERT



HW3: T5



# Training

---

- ❖ Pre-trained mt5-small is very large. (300M parameters, 3x than BERT-base)
- ❖ Some tips to reduce GPU memory usage:
  - Reduce batch size + gradient accumulation
  - Truncate text length (256/64 for input/output can pass the baseline)
  - fp16 ([transformers==4.5.0 has a bug on T5 fp16 training](#))
  - adafactor (instead of Adam)
- ❖ For reference, you can pass the baseline within 4 hours training on single RTX 3070 8G if your code is correct.

# How to Fix T5 FP16 Training

— — —

- <https://github.com/huggingface/transformers/pull/10956>
- Install fixed version transformers library
  - `git clone https://github.com/huggingface/transformers.git`
  - `git checkout t5-fp16-no-nans`
  - `pip install -e .`



# Documents

— — —

## ❖ T5

- [https://huggingface.co/transformers/model\\_doc/t5.html](https://huggingface.co/transformers/model_doc/t5.html)
- [https://huggingface.co/transformers/model\\_doc/mt5.html](https://huggingface.co/transformers/model_doc/mt5.html)

## ❖ Generation:

- [https://huggingface.co/transformers/main\\_classes/model.html#generation](https://huggingface.co/transformers/main_classes/model.html#generation)

Q&A