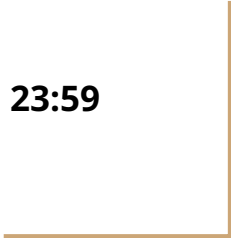# Applied Deep Learning Homework 2

Kaggle Due: **2022/11/09 23:59**

Code/Report Due: **2022/11/11 23:59**

Links

NTU COOL

討論區

說明影片

adl-ta@csie.ntu.edu.tw

TA Hours:

- 星期二 3:00 pm ~ 4:30pm @ Google Meet (Online)
- 星期四 4:00 pm ~ 5:30pm @ Google Meet (Online)

Change Log

如果有更新的資訊 會公布在此！

1. 降低Strong Baseline的分數，請同學在public盡量衝高一點(0.79以上較為保險)，避免private沒有過Strong Baseline

Change Log

如果有更新的資訊 會公布在此！

1. Huggingface上的pretrain model請不要使用訓練過QA或NLI data的model !!!
2. Transformer的tokenizer可以直接載嗎？還是要另外存在dropbox?因為助教有說會斷網，所以不是很確定，再麻煩助教回覆了，謝謝?
   a. 因為執行時會斷網，請另外存在 dropbox，可以使用 tokenizer.save_pretrained 來儲存你的 tokenizer !!!
3. 因為重新上傳檔案，NTU COOL自懂產生-1或-2的檔名是沒關係的

Change Log

如果有更新的資訊 會公布在此！

1. 剛好遇到期中考週，Deadline延後一週（請看NTU COOL公告確認最新繳交日期）
2. 如果遇到predict出1,234，建議可將1,234變成"1,234"，加上""就可以讓kaggle正確判斷
3. 建議如果使用huggingface github的script，可以使用no_trainer的sample code eg: run_swag_no_trainer.py(context selection) 和 run_qa_no_trainer.py(QA)
4. 如果遇到Huggingface transformers example有check_min_version("4.24.0.dev0")，註解掉這一行即可
5. 如果使用run_qa_no_trainer.py遇到error，建議可以在931行加上下面這一行程式碼
   a. predict_dataloader = accelerator.prepare(predict_dataloader)
6. 建議可以試試看將--max_length設為512(>384)

Outline

- Task Description
- Logistics
- Rules
- Report

Task Description

Chinese Question Answering

- Input: Context list

  "鼓是一種打擊樂器, ..., 最早的鼓出現於西元前六千年的兩河文明"

  "盧克萊修生於共和國末期, ..., 被古典主義文學視為經典"

  "視網膜又稱視衣, ..., 約3mm2大的橢圓。"

- Input: Question

  "最早的鼓可以追溯至什麼古文明?"

- Output: Answer

  "兩河文明"

Properties

- Model should predict the **answer** given **contexts** and a **question**.

- The answer is always a **span** in one of the contexts.

- This task can be decomposed into 2 tasks:

  - **Context selection**: determine which context is relevant.

  - **Span selection**: determine the start and end position of the answer span in the context.

Metrics

Exact Match (EM)

1 if prediction and answer are the same after preprocessing, 0 if they are different.

Data

- Context data
  - context.json
- Labeled Data
  - Training set: train.json
  - Validation set: valid.json (You can use this file for training model or only for validation)
- Unlabeled Data
  - Testing set: test.json
- Sample Submission
  - sample_submission.csv
- Check NTU COOL homework 2 page for download link.

Data Format - context.json

➢ list of short paragraphs

```
[
    "鼓是一種打擊樂器，也是一種通訊工具，非洲某些部落...的兩河文明。",
    "這次出售的贖罪券很特別，是全大赦贖罪券，可以贖買...購買，可見其盛況。",
    "處在千年古都的西安交大校園少不了和歷史千絲萬縷的...之一，可謂人傑地靈。",
    ...
]
```

Data Format - questions

➢ **id**: question ID

➢ **question:** question text

➢ **paragraphs:** list of 4 paragraph IDs (0-based)

➢ **relevant:** ID of the relevant context (0-based) (* absent in test.json)

➢ **answer**: answer to the question (* absent in test.json)

  ○ **text:** the answer text

  ○ **start**: answer span start position in the relevant context

```
[
  {
    "id": "593f14f960d971e294af884f0194b3a7",
    "question": "舍本和誰的數據能推算出連星的恆星的質量？",
    "paragraphs": [2018, 6952, 8264, 836],
    "relevant": 836,
    "answer": { "text": "斯特魯維", "start": 108 }
  },
  {
    "id": "acd5d763ec4c250f9a11eac1412d6814",
    "question": "在關西鎮以什麼方言為主？",
    "paragraphs": [1716, 8318, 4070, 7571],
    "relevant": 8318,
    "answer": { "text": "四縣腔客家話", "start": 306 }
  }
```

Data Format - questions

Submission Format

- A CSV file whose header is **id,answer**
- See sample_submission.csv
- Example

```
id,answer
593f14f960d971e294af884f0194b3a7,斯特魯維
acd5d763ec4c250f9a11eac1412d6814,四縣腔客家話
```

Objective

Fine-tune a pre-trained Chinese BERT-based model on the dataset and pass the baselines.

Simple Baseline - (bert-base-chinese would be enough)

   EM: Public 0.73508 (on Kaggle)

Strong Baseline - (You might need to explore other pretraining models)

   EM: Public 0.77396 (on Kaggle)

For this homework, you can utilize the huggingface transformers package and pretrained model. You can adapt their example scripts to our task

Logistics

Grading

- Model Performance (10%)
  - Your model passes the simple baseline on the public test set (2%) and the private test set (3%) on kaggle
  - Your model passes the strong baseline on the public test set (2%) and the private test set (3%) on kaggle
  - 0 point if we can't reproduce your submission using `run.sh`
- Format (1%)
  - TA can run the grading script without human intervention.
- Report (9% + 2% Bonus)
  - In PDF format!

Code/Scripts/Report/Result Submission

- Zip your folder into a single **.zip** file.
- Submit to NTU Cool.

File Layout

Your zip must contain files (case sensitive):

- `/[student id (lower-cased)]/`, ex. `/r12922000/`, no brackets
  - `run.sh`
  - `README.md`
  - `report.pdf`
  - `download.sh`
  - Any other code/script.
- Do not upload training, validation, testing data and model to COOL.

Submission Files – download.sh

- `download.sh` to download your model.
  - 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)
- You can use **gdown** to download you model from Google Drive (please make sure we have the access to download)
- We will execute download.sh before predicting scripts.

Submission Files - Scripts

- `run.sh`
- Arguments:
  - `"${1}"`: path to the context file.
  - `"${2}"`: path to the testing file.
  - `"${3}"`: path to the output predictions.
- TA will predict testing data as follow:
  - `bash ./download.sh`
  - `bash ./run.sh /path/to/context.json /path/to/test.json  /path/to/pred/prediction.csv`

- 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 Files - Reproducibility

- All the code you used to train, predict, plot figures for the report should should be uploaded.
- README.md
  - Write down <u>how to train your model</u> 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
- Time limit: **2** hours for `run.sh` in total
- No network access when predicting (after we run download.sh).
- You will lose (some or all) your model performance score if
  - your script is at wrong location, or cause any error.

Rules

What You Can Do

- Train with the data we give you.
- Use publicly available pre-trained BERTs and their variants.
- Use the packages/tools we allow:
  - Python 3.8 / 3.9 and Python Standard Library
  - PyTorch 1.12.1, TensorFlow 2.10.0
  - Tqdm, numpy, pandas, scikit-learn 1.1.2, nltk 3.7
  - transformers==4.22.2, datasets==2.5.2, accelerate==0.13.0
  - Dependencies of above packages/tools.
- If you want to use other package, COOL/mail TA.

What You Can NOT Do

- Any means of cheating or plagiarism, including but not limited to:
  - Use others' code from anywhere (e.g. web, github, classmate, etc.).
  - Use the labels of the test data directly or indirectly. (Do not try to find them.)
  - Use package or tools not allowed.
  - Use model trained with other QA/NLI data (If not sure, ask TA first).
  - Give/get model prediction to/from others.
  - Give/get trained model to/from others.
  - Publish your code before deadline.
  - Submit to past Kaggle competitions.
- Violation may cause zero/negative score and punishment from school.

Submission Policy

- Submit to NTU Cool.
- No Late submission.

Report

You may lose score if TA has difficulty understanding it.

Please write in a human-readable way.

When Describing Model

- Please limit the use of imprecise words.
- Use equation whenever possible.
- Descriptions which is imprecise or hard to understand may cause loss of points.
- Ex.
  - bad: Feed the embedding of the sentence into a LSTM.
  - good: h_t, c_t = LSTM(w_t, h_{t-1}, c_{t-1}), where w_t is the word embedding of the t-th token.

Q1: Data processing (2%)

1. Tokenizer (1%):
   a. Describe in detail about the tokenization algorithm you use. You need to explain what it does in your own ways.

2. Answer Span (1%):
   a. How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?
   b. After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position?

Q2: Modeling with BERTs and their variants (4%)

1. Describe (2%)
   a. your model (configuration of the transformer model)
   b. performance of your model.
   c. the loss function you used.
   d. The optimization algorithm (e.g. Adam), learning rate and batch size.

2. Try another type of pretrained model and describe (2%)
   a. your model
   b. performance of your model
   c. the difference between pretrained model (architecture, pretraining loss, etc.)
   d. For example, BERT -> xlnet, or BERT -> BERT-wwm-ext. You can find these models in huggingface's Model Hub.

Q3: Curves (1%)

1. Plot the learning curve of your QA model
   a. Learning curve of loss (0.5%)
   b. Learning curve of EM (0.5%)

*You don't need to calculate the metrics in every iteration. Please make sure there are at least **5 data points** in each curve.*

Q4: Pretrained vs Not Pretrained (2%)

- Train a transformer model from scratch (without pretrained weights) on the dataset (you can choose either MC or QA)

- Describe

  - The configuration of the model and how do you train this model

  - the performance of this model v.s. BERT

- Hint: you can use the same training script for this problem, just skip the part where you load the pretrained weights

- Hint: the model size configuration for BERT might be too large for this problem, if you find it hard to train a model of the same size, try to reduce model size (num_layers, hidden_dim, num_heads). Remember to report the model configuration.

Q5: Bonus: HW1 with BERTs (2%)

- Train a BERT-based model on HW1 dataset and describe
  a. your model
  b. performance of your model.
     i. Intent classification (1%)
     ii. Slot tagging (1%)
  c. the loss function you used.
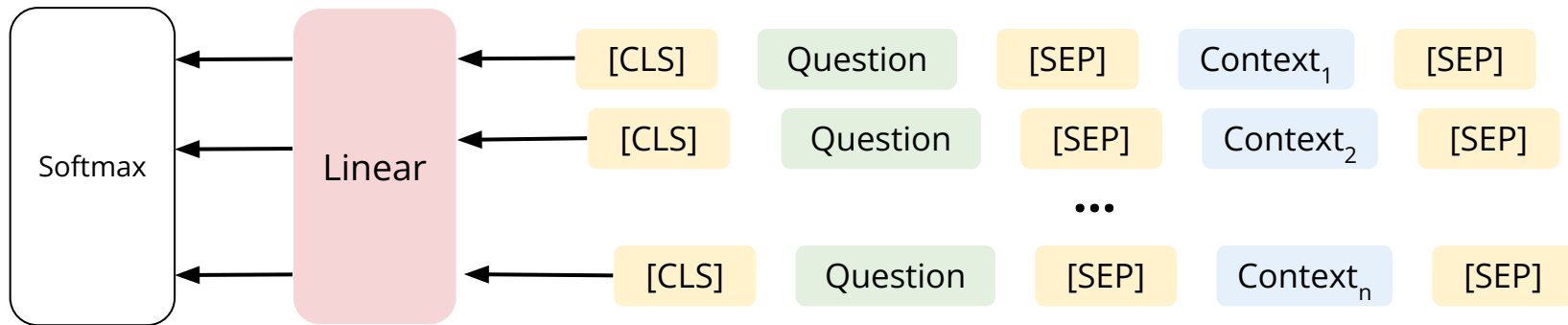 d. The optimization algorithm (e.g. Adam), learning rate and batch size.

Guide

Pipeline
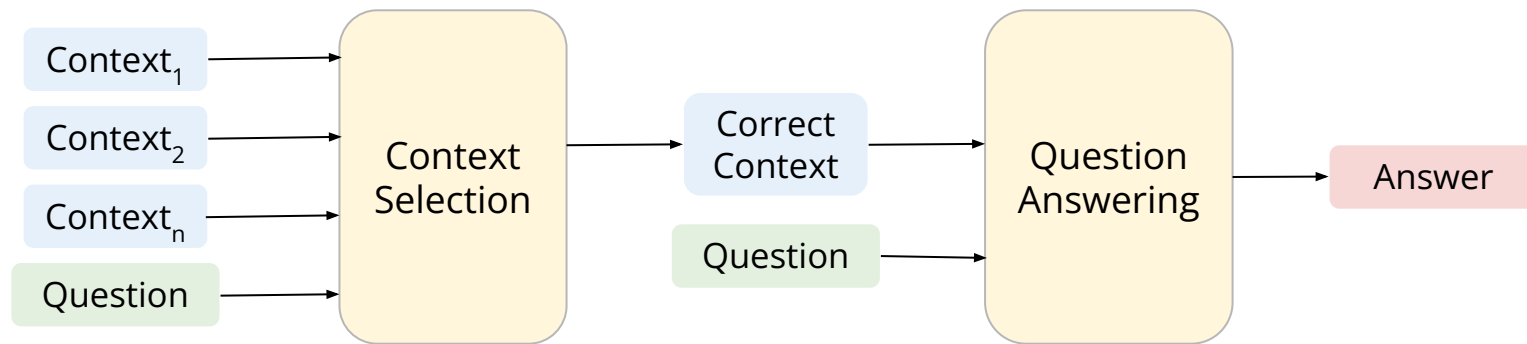
Context Selection

- In the first stage, you should train a model to select the relevant context

- A simple way to do this is by using the BertForMultipleChoice model in Huggingface transformers (example)

- For each question, you can view a (context, question) pair as a choice, and then ask the model to predict the correct choice

Softmax	Linear	[CLS]	Question	[SEP]	$Context_1$	[SEP]
		[CLS]	Question	[SEP]	$Context_2$	[SEP]
		...				
		[CLS]	Question	[SEP]	$Context_n$	[SEP]

Context Selection

- By making the format of our dataset the same as the expected format of the example script (SWAG dataset), you can use the script to train a context selection model out-of-the-box.

- The default ratio of the example script is 1 positive : 3 negative.

Context Selection

- Some tricks to reduce memory usage:
  - Use [gradient accumulation](#) to reduce memory usage without changing effective batch size, we do not encourage simply reducing batch size as it might hurt performance
  - Effective batch size = batch_size * gradient_accumulation_steps
- We recommend using max_length=512

Question Answering

- The easiest way to do this is by using/modifying the [script](#) from huggingface transformers. If you make the format of our dataset the same as their expected format (SQuAD dataset), you can use the script out-of-the-box.

- You can simply use the relevant context and the corresponding answer to train your model for this task, let the context selection model handle the selection part

Question Answering

- If you do the preprocessing yourself, be careful handling the position of **answer_span** after tokenization. BERT uses subword tokenization which split a word into subwords, thus changing the *start_position* of the correct span.

- Use the *start_position* to identify the correct span. Do not use something like `context.index("兩河流域")` as you might find another appearance of the answer text, which does not contain the clue to answer the question, such that the model could not learn how to do answer questions.

Question Answering

- If the input length is longer than 512, or you want to truncate the input text, be careful of the **answer_span**. <mark>You should keep the correct span in the truncated input text, otherwise there will be no answer to select.</mark>

- *If you use the example scripts in the transformers library out-of-the-box, they deal this part for you.*

- *We recommend using the postprocessing function in the transformers library.*

Simple Baseline Hyperparameters

- Context classification
    - Pretrained model: bert-base-chinese
    - Max_len: 512
    - Batch_size: 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
    - Num_train_epochs: 1
    - Learning_rate: 3e-5
    - Total running time: < 2 hours

- Question answering
    - Pretrained model: bert-base-chinese
    - Max_len: 512
    - Batch_size: 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
    - Num_train_epochs: 1~3
    - Learning_rate: 3e-5
    - Total running time: < 1 hour

- Resource used: Nvidia RTX 3070 with 8GB memory