NTU ADL 2024 Fall
HW1

Last updated on 9/28.

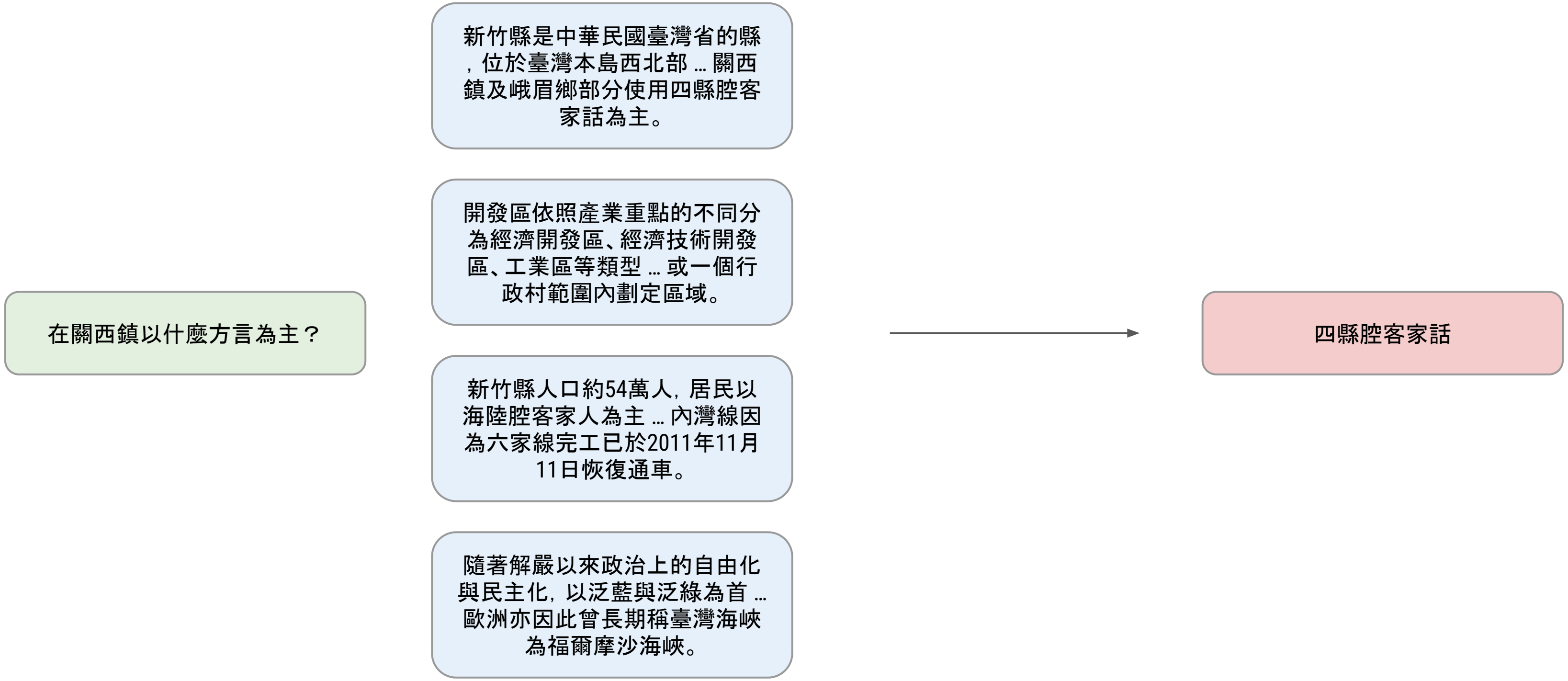
Due Date

- Kaggle leaderboard - **10/8 (Tue.) 23:59**
- Code and report (submit to NTU COOL) - **10/10 (Thu.) 23:59**
- **No late submission**.

Change Logs

- [09/11] Hw1 announced.
- [09/11] Kaggle competition launched.
- [09/23] Baselines announced
- [09/23] Add the following allowed models: hfl/chinese-bert-wwm, hfl/chinese-bert-wwm-ext, macbert, hfl/chinese-lert
- [09/28] Deadlines extended by 1 week

Task Description - Chinese Extractive Question Answering (QA)



在關西鎮以什麼方言為主？

新竹縣是中華民國臺灣省的縣，位於臺灣本島西北部 … 關西鎮及峨眉鄉部分使用四縣腔客家話為主。

開發區依照產業重點的不同分為經濟開發區、經濟技術開發區、工業區等類型 … 或一個行政村範圍內劃定區域。

新竹縣人口約54萬人，居民以海陸腔客家人為主 … 內灣線因為六家線完工已於2011年11月11日恢復通車。

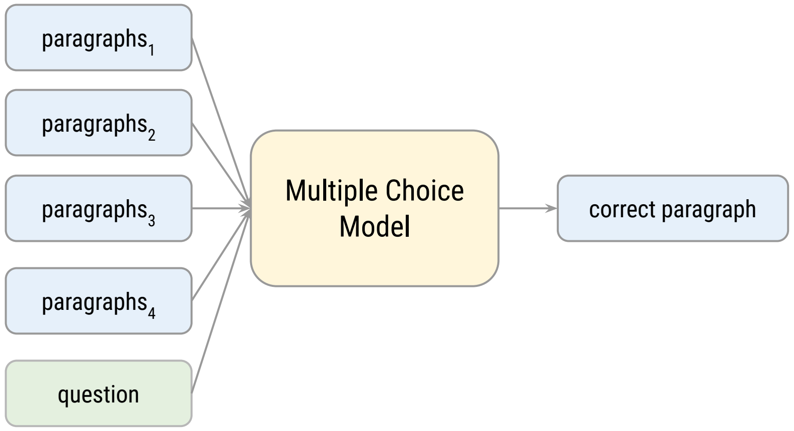
隨著解嚴以來政治上的自由化與民主化，以泛藍與泛綠為首 … 歐洲亦因此曾長期稱臺灣海峽為福爾摩沙海峽。

四縣腔客家話

Task Description - Chinese Extractive Question Answering (QA)

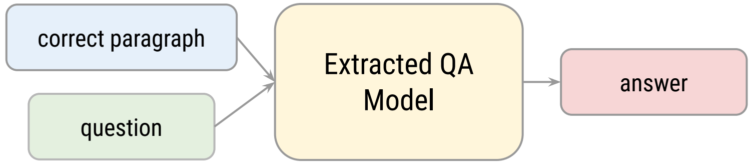
Paragraph Selection:

Determine which paragraph is relevant.



Span selection:

Determine the start and end position of the answer span.



The answer is always a **span in the correct paragraph.*

An Example

新竹縣是中華民國臺灣省的縣，位於臺灣本島西北部 … 關西鎮及峨眉鄉部分使用四縣腔客家話為主。

開發區依照產業重點的不同分為經濟開發區、經濟技術開發區、工業區等類型 … 或一個行政村範圍內劃定區域。

新竹縣人口約54萬人，居民以海陸腔客家人為主 … 內灣線因為六家線完工已於2011年11月11日恢復通車。

隨著解嚴以來政治上的自由化與民主化，以泛藍與泛綠為首 … 歐洲亦因此曾長期稱臺灣海峽為福爾摩沙海峽。

在關西鎮以什麼方言為主？

An Example (cont.)

新竹縣是中華民國臺灣省的縣，位於臺灣本島西北部 … 關西鎮及峨眉鄉部分使用四縣腔客家話為主。

開發區依照產業重點的不同分為經濟開發區、經濟技術開發區、工業區等類型 … 或一個行政村範圍內劃定區域。

新竹縣人口約54萬人，居民以海陸腔客家人為主 … 內灣線因為六家線完工已於2011年11月11日恢復通車。

隨著解嚴以來政治上的自由化與民主化，以泛藍與泛綠為首 … 歐洲亦因此曾長期稱臺灣海峽為福爾摩沙海峽。

在關西鎮以什麼方言為主？

An Example (cont.)

新竹縣是中華民國臺灣省的縣，位於臺灣本島西北部 … 關西鎮及峨眉鄉部分使用四縣腔客家話為主。

開發區依照產業重點的不同分為經濟開發區、經濟技術開發區、工業區等類型 … 或一個行政村範圍內劃定區域。

新竹縣人口約54萬人，居民以海陸腔客家人為主 … 內灣線因為六家線完工已於2011年11月11日恢復通車。

隨著解嚴以來政治上的自由化與民主化，以泛藍與泛綠為首 … 歐洲亦因此曾長期稱臺灣海峽為福爾摩沙海峽。

在關西鎮以什麼方言為主？

Data & Evaluation Metric

- Data can be downloaded on Kaggle
- We will use Exact Match (EM) for evaluation

Guides - Paragraph Selection

- For each question, you can view a paragraph-question pair as a choice, and then ask the model to predict the correct choice.
- You can modify this huggingface example code for multiple choice (highly recommended!!!).
- By making the format of our dataset the same as the expected format of the example code, you can use the example code out-of-the-box to train the desired paragraph selection model.

Guides - Span Selection (Extractive QA)

- You can modify this huggingface example code for extractive QA (highly recommended!!!).
- By making the format of our dataset the same as the expected format of the example code, you can use the example code out-of-the-box to train the desired span selection model.
- Use the start position to identify the answer span.
    - Do not use something like `context.index(“四縣腔客家話”)` as you might find another appearance of the answer text, which does not appear in the context to answer the question.

Guides - For Passing the Kaggle Simple Baseline

- Paragraph selection
  - Pretrained LM: 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
- Resource used: Nvidia RTX 3070 with 8GB memory

Guides - For Passing the Kaggle Simple Baseline (cont.)

- Span selection
  - Pre-trained LM: 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

Tips

- A public score of 0.78 will have a high probability to pass private strong baseline.
- You can safely comment `check_min_version("4.45.0.dev0")` if you encounter it in huggingface transformers example code.
- Setting a longer `max_length` (e.g. 512) is likely to bring a better performance.
- Some tricks to reduce memory usage:
  - Use gradient accumulation to reduce the memory usage without changing the effective batch size (simply reducing batch size might hurt the performance).
  - Effective batch size = batch_size * gradient_accumulation_steps

Rules - What You Can Do

- Only train with the data we give you.
- Use publicly available pre-trained LMs.
- Allowed packages:
  - Python 3.10 and Python Standard Library
  - PyTorch 2.1.0, scikit-learn 1.5.1, nltk 3.9.1
  - tqdm, numpy, pandas
  - transformers 4.44.2, datasets 2.21.0, accelerate 0.34.2
  - evaluate, matplotlib, gdown
- Dependencies of above packages.

Rules - What You Cannot Do

- Any means of cheating or plagiarism, e.g., use others' github code from previous years' ADL course.
- 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 or NLI datasets, including but not limited to
    - luhua/chinese_pretrain_mrc_macbert_large, uer/roberta-base-chinese-extractive-qa, NchuNLP/Chinese-Question-Answering
    - If not sure, ask TA first.
    - The following models are allowed:
        - hfl/chinese-bert-wwm, hfl/chinese-bert-wwm-ext, macbert, hfl/chinese-lert
- Give/get trained model/predictions to/from others.
- Publish your code before deadline.
- Submit to previous years's ADL Kaggle page.
- Violations may cause zero/negative score and punishment from school.

Submission

1. Kaggle leaderboard. Please set the team name as your student id (lower-cased) (ex. r11000000).
2. Zip your folder, which should be named as your student id (lower-cased) (ex. r11000000) and submit the `.zip` to NTU Cool. Your folder should contain
   - `README.md`
   - `run.sh`
   - `download.sh`
   - `report.pdf`
   - your code/script (all the code/script you used to train, predict, or plot report figures should be included).
   - Do not upload training, validation, testing data or models to COOL.

Submission - download.sh

- `download.sh` should download **your models, tokenizers** and **data**.
- You can upload your models/tokenizers/data to
  - Dropbox and use `wget` to download.
  - Google Drive and use `gdown` to download.
- Please make sure we have the access to download!
- Do not modify the files in download links after the deadline, this action is cosidered cheating.
- Keep the download links 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 **4GB**, and `download.sh` should finish within **1 hour**. (At csie dept. with maximum 10MB/s bandwidth)
- We will execute `download.sh` before running any other scripts.

Submission - run.sh

- `run.sh` should perform inference using your trained models and output predictions on `test.json`.
- 3 arguments are required:
  a. `"${1}"`: path to `context.json`.
  b. `"${2}"`: path to `test.json`.
  c. `"${3}"`: path to the output prediction file named `prediction.csv`.
- TA will predict testing data as follow:
  a. `bash ./download.sh`
  b. `bash ./run.sh /path/to/context.json /path/to/test.json /path/to/pred/prediction.csv`
- `run.sh` should finish within **2 hours**. (See next page for environment details)
- Make sure your code works!

Submission - Execution Environment

- Your code/script will be run on a server with
  a. Ubuntu 20.04
  b. 32GB RAM, RTX 2080 Ti 11GB VRAM, and 20GB disk space available.
- The packages we allow only.
- Python 3.10
- No network access after we run `download.sh`.

Submission - README.md

- README.md should contain step-by-step instruction on how to train your model with your codes/scripts.
- You will get a **-2** penalty if you have no or empty README.md.
- 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.

Grading

- Model Performance (11%)
  - Kaggle simple baseline: public (2%), private (3%).
  - Kaggle strong baseline: public (2%), private (3%).
  - TA can reproduce your results *without* human intervention. (1%)
  - 0 point if we cannot reproduce your submission after human intervention.
- Report (9% + 2% Bonus)
  - In PDF format.

Report - Q1: Data processing (2%)

- Tokenizer (1%):
  - Describe in detail about the tokenization algorithm you use. You need to explain what algorithm does in your own ways. Just answering with “I called the () function” is not allowed.
- Answer Span (1%):
  - How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?
  - After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position?

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

- Describe (2%)
  - Your model.
  - The performance of your model.
  - The loss function you used.
  - The optimization algorithm (e.g. Adam), learning rate and batch size.
- Try another type of pre-trained LMs and describe (2%)
  - The new model.
  - The performance of this model.
  - The difference between pre-trained LMs (architecture, pretraining loss, etc.)
  - For example, BERT -> xlnet, or BERT -> BERT-wwm-ext. You can find these models in the huggingface's Model Hub.

Report - Q3: Curves (1%)

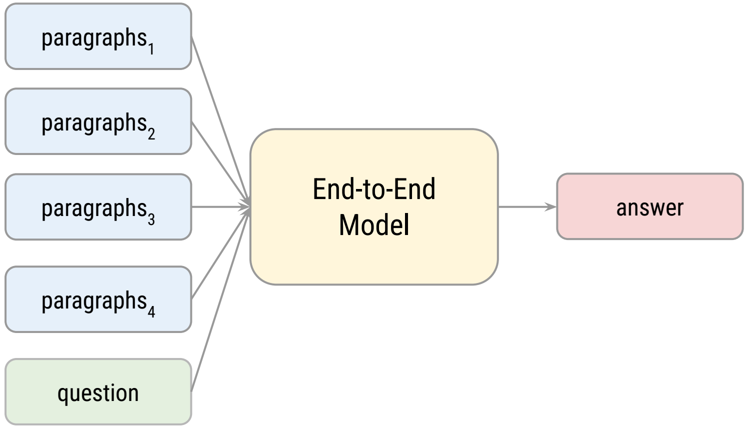
- Plot the learning curve of your span selection (extractive QA) model. **You can plot both the training and validation set or only one set.** Please make sure there are at least **5 data points** in each curve.
  - Learning curve of the loss value (0.5%)
  - Learning curve of the Exact Match metric value (0.5%)

Report - Q4: Pre-trained vs Not Pre-trained (2%)

- Train a transformer-based model (you can choose either paragraph selection or span selection) from scratch (i.e. without pretrained weights).
- Describe
  - The configuration of the model and how do you train this model (e.g., hyper-parameters).
  - The performance of this model v.s. BERT.
- Hint
  - You can use the same training code, just skip the part where you load the pretrained weights.
  - 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 (e.g. num_layers, hidden_dim, num_heads).

Report - Q5: Bonus (2%)

- Instead of the paragraph selection + span selection pipeline approach, train an **end-to-end** transformer-based model and describe
  - Your model.
  - The performance of your model.
  - The loss function you used.
  - The optimization algorithm (e.g. Adam), learning rate and batch size.
- Hint: Try models that can handle long input (e.g., models that have a larger context windows).



Any questions?

- COOL discussion board (Answered by both TAs and classmates, encouraged!!!)
- TAs
  - Email: adl-ta@csie.ntu.edu.tw (When sending emails, add “[ADL2024 HW1]” to the beginning of the title.)
  - Office hour: Every Monday 17:00~18:00
    - Sep. 16 & 23: Online @ https://meet.google.com/toj-dhac-wwf
    - Sep. 30 & Oct. 7: @ Lab 524