HW3

Q1 Model (2%)

Model (1%)

I am using mt5 as the model architecture for this machine learning project.

This model is base on t5 but using cross-language data and training.

I will explain the architecture for t5 below briefly.

In simple terms, we can think of T5 as a complete Transformer model structure. What is a Transformer model? It is composed of two parts: the encoder (Encoder) and the decoder (Decoder).

T5 can convert all NLP tasks into the form of Text-to-Text (used to be called Seq-to-Seq, the text input is encoded by the Encoder, and then decoded by the Decoder to obtain the final text output)

On the other hand, T5 have a special mechanism called “prefix”.

T5 is pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task, e.g., for translation: *translate English to German: …*, for summarization: *summarize: ….*

As mentioned above, we can prefix the input with "summarize" because the summarization task is pre-trained for the model, and the model can do it well.

Preprocessing (1%)

I only do tokenization in my project. I will explain mt5 tokenization below briefly.

Because mt5 is base on t5, so I just check the document for t5.

We can find that the tokenization of it is base on **SentencePiece**.

SentencePiece is an unsupervised text tokenizer and de-tokenizer mainly for Neural Network-based text generation systems where the vocabulary size is predetermined prior to the neural model training. SentencePiece implements **subword units** (e.g., **byte-pair-encoding (BPE)** and **unigram language model** with the extension of direct training from raw sentences.

BPE principle is to express two common consecutive symbols with another symbol. For example, in abababab, a is often followed by b, we use c to express ab, and get a new sequence ccc, c=ab.

unigram language model is a special case for “n=1” of n-gram language model.

It refers to n words that appear consecutively in a text. The n-gram model is a probabilistic language model based on (n-1) order Markov chains, which infers the structure of sentences by the probability of occurrence of n words.

Q2 Training (2%)

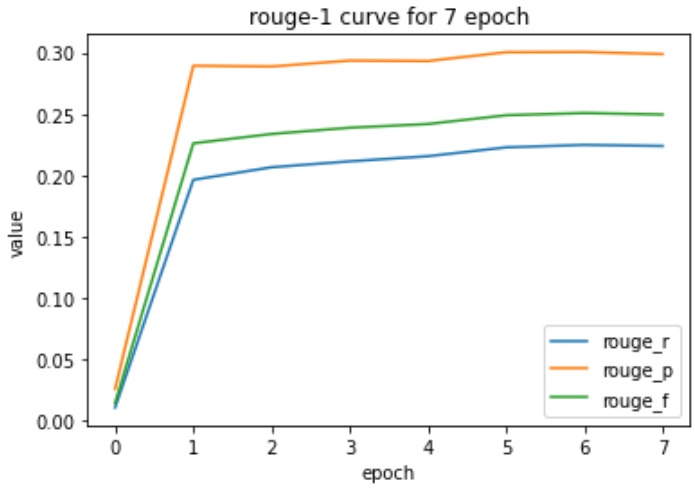
Hyperparameter (1%)

```
The following hyperparameters were used during training:
- learning_rate: 0.0005
- train_batch_size: 8
- eval_batch_size: 8
- seed: 42
- gradient_accumulation_steps: 2
- total_train_batch_size: 16
- optimizer: Adafactor
- lr_scheduler_type: linear
- num_epochs: 7.0
```

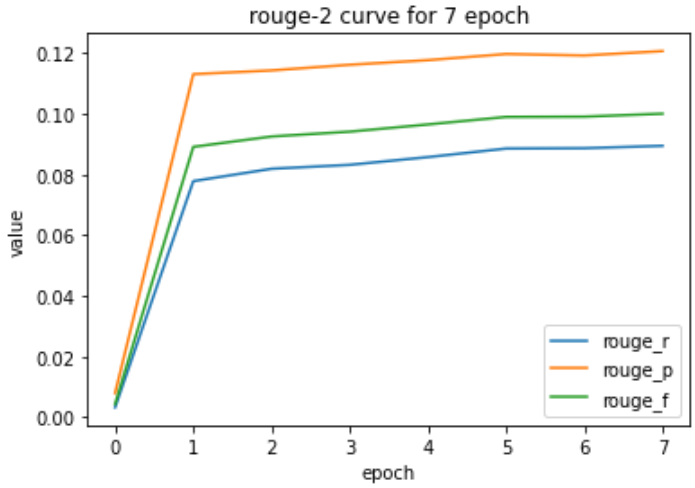
Learning Curves (1%)

- plot

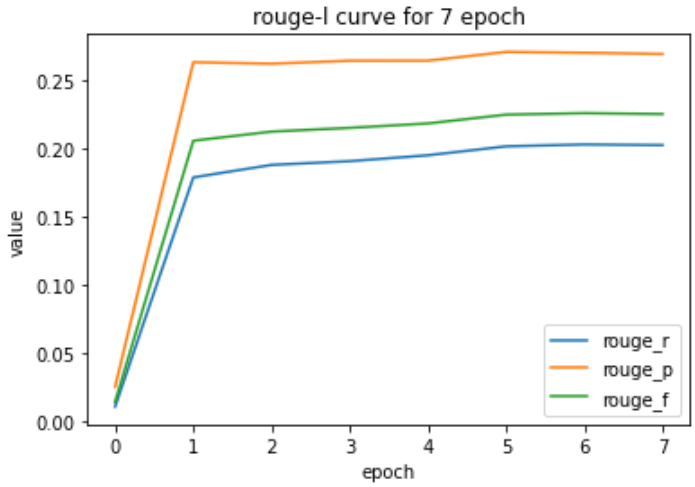
- rouge 1



- rouge 2



- rouge L



plot by the code in `./colab`

Q3 Generation Strategies(6%)

Stratgies (2%)

- Describe the detail of the following generation strategies:
  - Greedy
    - a simple algorithm
    - At each step, take the most probable word (i.e. argmax)
    - use it as the next word and provide it as input in the next step
    - keep going until formula is produced or some maximum length is reached
    - Output may be poor (eg, ungrammatical, unnatural, absurd) due to lack of backtracking
  - Beam Search
    - A search algorithm designed to find sequences with high probability (not necessarily the best) by tracking multiple possible sequences at once
    - **Core idea**: at each step of the decoder, keep track of the k most probable partial sequences (k is the beam size)
    - After reaching a certain stopping criterion, select the sequence with the highest probability (considering some length adjustments)
  - Top-k Sampling

    Top-k Sampling uses a strategy to sample from a shortlist of top k tokens. This method allows other high-scoring tokens to have a chance to be selected. The randomness introduced by this sampling helps to improve generation quality in many scenarios.
  - Top-p Sampling

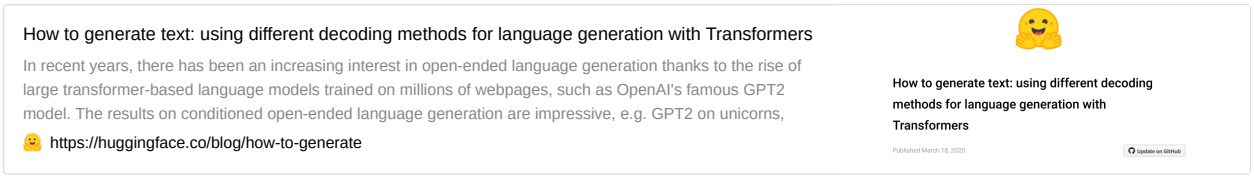
    top-p sampling sets the size of the token candidate list. This method shortlists top-level tokens whose likelihoods do not add up to a certain value
  - Temperature

    The temperature parameter t is introduced in the greedy calculation process to change the word probability distribution, making it more inclined to high probability words
    Just like the formula below, the value of t is between [0,1).

```
P(x|x1:t−1)=exp(ut/t)∑t'exp(ut'/t))P(x|x_{1:t-1})=\frac{exp(u_{t}/t)}{\sum {{t}'}exp(u{{t}'}/t))}P(x|x1:t−1)=∑t'exp(ut'/t))exp(ut/t
```

Hyperparameters (4%)

ref:



- Greedy
  - 實作

    ```
    from transformers import pipeline
    summarizer = pipeline(
            "summarization", model='../model', max_length=20)
    ```

  - 效果

    ```
    "rouge-1": {
        "r": 0.19553722196587202,
        "p": 0.253080303658285,
        "f": 0.21177408488498417
      },
    "rouge-2": {
        "r": 0.07577774193560375,
        "p": 0.10773741049595449,
        "f": 0.07654427177741884
    },
    "rouge-l": {
        "r": 0.15509865169969245,
        "p": 0.22464649153544884,
        "f": 0.2084953716141435
    }
    ```

  - 評價

    未過 baseline , 很明顯效果不彰

    ```
    [{'summary_text': 'Anker推出真無線藍牙耳機 支援App操作'}]
    [{'summary_text': '苗栗「三義」新玩法！ 超狂「蔬皮肚皮」'}]
    [{'summary_text': '華碩換上Intel第11代Core處理器Chromebook Flip 將在4'}]
    [{'summary_text': '智慧化、智慧化 台灣為何不該跟上台廠?'}]
    [{'summary_text': '微軟:Windows 7市佔率落在20% 但仍有更多裝置'}]
    [{'summary_text': '台幣回流 這類股是什麼?'}]
    [{'summary_text': '美國網購平台上架「哈台馬克杯」 網友熱銷'}]
    [{'summary_text': '華碩更新雙螢幕筆電ZenBook Duo 14、Pro Duo 15 OLED'}]
    [{'summary_text': '週末炸雞俱樂部！ 台虎推「周末炸雞」 加'}]
    [{'summary_text': 'NBA/紐約記者爆料厄文 曾被曝不喜歡奈許'}]
    [{'summary_text': '健身教練:運動不斷 讓你變成破關遊戲'}]
    [{'summary_text': '華碩推出小型LED投影機 可透過遙控器、遙控器'}]
    [{'summary_text': '《發聲什麼事》 讓你聽出聲音的力量'}]
    [{'summary_text': '華碩推出新款14吋IP外接顯示器 加入客製化軟'}]
    [{'summary_text': '從「當心」看「適合自己」 如何讓自己成為「適合自己'}]
    [{'summary_text': '中國將在5G領域布局 加速車聯網應用'}]
    [{'summary_text': '金士頓推出首款PCIe Gen 4.0 SSD 與HyperX電競品牌'}]
    [{'summary_text': '台塑四寶股價飆漲 台股大漲逾20%'}]
    ```

    可以看出文具很順但沒抓到重點, 感覺像在自說自話

- Beam Search

- 設定 beams=5
  - 實作

```
from transformers import pipeline
summarizer = pipeline(
        "summarization", model='../model', max_length=20, num_beams=5)
```

  - 效果

```
"rouge-1": {
    "r": 0.2239020253726327,
    "p": 0.2987549567901063,
    "f": 0.24952843916581152
},
"rouge-2": {
    "r": 0.08949925556786237,
    "p": 0.12071952767604702,
    "f": 0.10008353530115667
},
"rouge-l": {
    "r": 0.20256687618597122,
    "p": 0.2695475725925197,
    "f": 0.22540043410896224
}
```

  - 評價

    效果極佳

- 設定 beams=50
  - 實作

```
from transformers import pipeline
summarizer = pipeline(
        "summarization", model='../model', max_length=20, num_beams=10)
```

  - 效果

```
[{'summary_text': 'Anker新款真無線藍牙耳機Liberty Air 2 Pro引進台灣市場'}]
[{'summary_text': '迷你縮小版!苗栗「七姊八弟山城小店」'}]
[{'summary_text': '華碩推出換上Intel第11代Core處理器Chromebook Flip Q5'}]
```

  - 評價

    基本上落差不大, 主要是運算時間太久, 可能因為機率分布上前幾個路徑就會取得最佳解

- Top-k Sampling
  - k=50
    - 實作

```
from transformers import pipeline
summarizer = pipeline(
        "summarization", model='../model', max_length=20, do_sample=True, top_k=50)
```

    - 效果

      基本上效果很差, 沒辦法抓到重點

    - 評價

可能因為機率分布上前幾個可能性特別高, 所以隨機在那麼多個取結果效果不好

- k=3
  - 實作

```
from transformers import pipeline
summarizer = pipeline(
        "summarization", model='../model', max_length=20, do_sample=True, top_k=3)
```

  - 效果

    和貪婪演算法的成果類似

  - 評價

    可能因為機率分布上前幾個可能性特別高, 所以 top-k 演算法數字設的太大效果反而很差

- Top-p Sampling
  - p=0.92
    - 實作

```
from transformers import pipeline
summarizer = pipeline(
        "summarization", model='../model', max_length=20, do_sample=True, top_p=0.92, top_k=0)
```

    - 效果

      成果極差

    - 評價

      可能前幾個選近來效果會比較好, 所以加入這種對大多數選項隨機挑選的機制後效果不彰

  - p=0.5
    - 實作

```
from transformers import pipeline
summarizer = pipeline(
        "summarization", model='../model', max_length=20, do_sample=True, top_p=0.5, top_k=0)
```

    - 效果

      近似貪婪演算法

    - 評價

      可能因為機率分布上前幾個可能性特別高, 所以 p 數字設的小就會在前幾個挑選, 結果類似貪婪

- Temperature
  - t=0.3
    - 實作

```
from transformers import pipeline
summarizer = pipeline(
        "summarization", model='../model', max_length=20, do_sample=True, top_k=0, temperature=0.3)
```

    - 效果

      近似貪婪演算法

    - 評價

在 t 小的時候, 對後續機率小的選項的可能性影響較低, 所以效果類似貪婪

- t=0.7
  - 實作

    ```
    from transformers import pipeline
    summarizer = pipeline(
            "summarization", model='../model', max_length=20, do_sample=True, top_k=0, temperature=0.7)
    ```

  - 效果

    成果極差

  - 評價

    可能因為機率分布上前幾個可能性特別高也效果特別好, 所以雖然語句更順了, 但卻抓不到重點

💡 What is your final generation strategy ? **Beam Search** for **num = 5**