QA: Question Answering System

A task in NLP & NLU, sub-tasks in AI

QA consists of three parts:

1. [Doc]uments
2. [Q]uestions
3. [A]nswers

Answering has mainly 2 types:

1. Extractive
2. Yes/No

Here: We will learn

1. Fine-tuning extractive QA
2. Fine-tuning Yes/No QA
3. Deploying finetuned model
4. A rule-based method for Hackathon proposal

Using tools:

* Colab (iPython Notebook)
* Transformers (for manual fine-tuning)
* Simpletransformers (Easy to use)
* Python Library for manipulating data
  + E.g. os, JSON, pandas, NumPy, etc.

1. QA Fintuning Extractive Answering using SimpleTranfromers

Pipeline:

Dataset(Q,A,D) → SQUAD2.0 -> SimpleTransformers

* SimpleTransformers require feeding data in SQUAD format (see https://simpletransformers.ai/docs/qa-data-formats/)

Dataset -> Train

-> Eval (or validate)

-> Test

Transformers: Pre-trained + Fine-tune

* Pre-trained: can be deployed from Hugging Face
* Fine-tune: we will implement this

Files: Thai wiki

1. 'AIFORTHAI-ThaiQACorpus.zip': Q, A
2. 'Wiki-documents.zip': D

Common notice: sometimes they give D more than actually necessary in Q, A

What we need to do:

(prerequisite) install all necessary libraries

1. Preprocess
2. Explore Q, A
   1. Check format
   2. Check amount
   3. Check description
3. Finding Doc or Article that these QA needed
   1. Pick a unique list of used articles
   2. Read used documents (put to RAM)
4. (Recommended) separate [Train, Test, Eval] since doing later is more complex
5. Make SQuAD format for both of them
6. Save all slow-processed data
7. Model
   1. Select a model as Pretrained
   2. Qualify training arguments
   3. Train a model
   4. Evaluate results if not, repeat a-c
   5. Try predict on the test set
   6. Save the model
8. QA Fintuning Yes/No using Transformers

Pipeline:

Dataset(QA, yes/no) -> preprocessing -> modelling

We need to do:

1. Pre-process
   1. Labeling answers to either 0 or 1
   2. Join QA DataFrame with QA documents DataFrame
2. Modeling
   1. Split train eval test in the form of datasets
   2. Load Pretrained, Tokenizer from hugging face -> using the sequence classification model
   3. Tokenize question, content, labels of datasets
   4. Qualify training arguments and prepare data collater
   5. Train and examine the results
3. QA Deploying Fine-tuned Model

What we need to do:

1. Prepare dataset, rearrange into DataFrame, or else
2. Load fine-tuned model, tokenizer from Hugging Face
3. Implement an iterative program for predicting each tuple of question, content, labels
   1. First, tokenize question and context
   2. Put it into the loaded model
   3. Locate an answer from the model’s output
   4. (optional) using pipeline will be useful for reducing processing
4. Examine the results

4. A rule-based method for Hackathon proposal

We tend to leverage a rule-based method for the special condition to increase the accuracy.

1. Implement a QA program (if we have times)

Dataset(Wiki, SCG) –find-> Extractive or YesNo

* What is the word that differentiates both?

If Extractive then use a pretrained to answer

If Yes/No the use sequence classifier model to answer

When finished, pack both into submission.csv [article\_id, answer] and try submitting to Kaggle