HW3: Headline Generation

Start Date : 2024/03/26 now

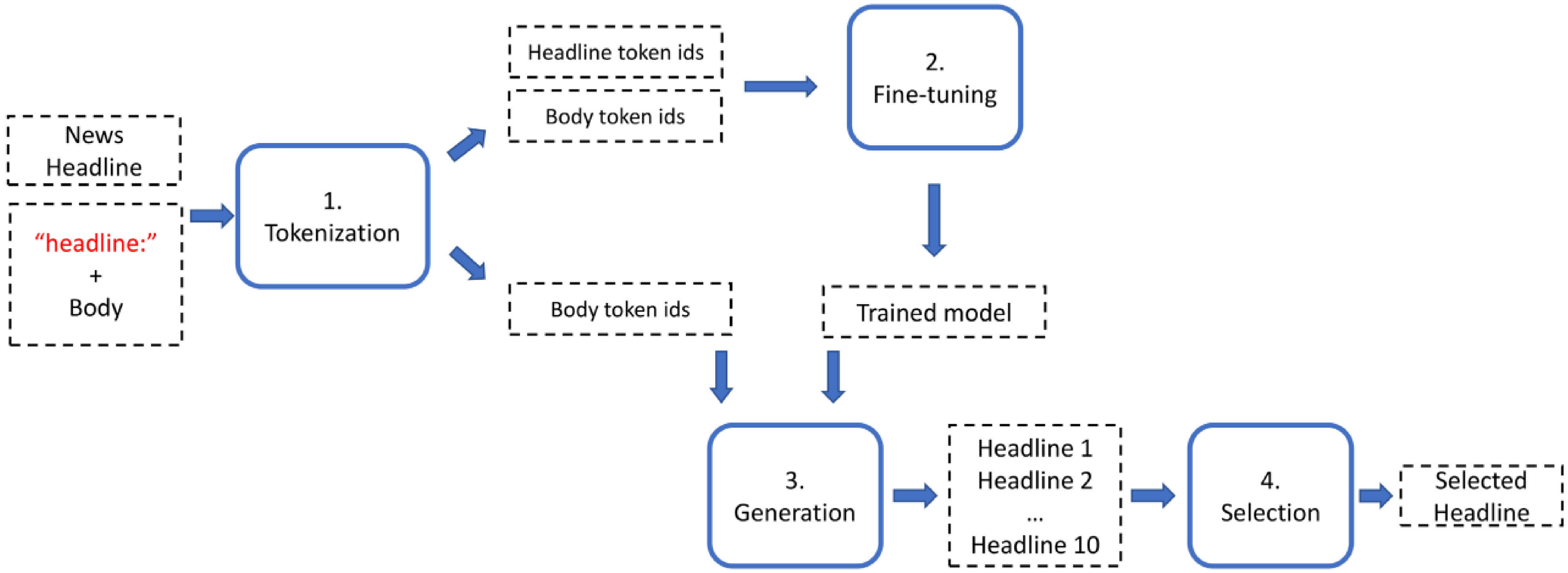
Checkpoint -1 : 2024/04/01 23:59

Checkpoint -2 : 2024/04/08 23:59

Deadline : 2024/04/15 23:59

[TA] Tzu-Ling Lin (林子淩)

Task Overview – headline generation



Ref: A. Omidvar and A. An, "Learning to Generate Popular Headlines," in IEEE Access, vol. 11, pp. 60904-60914, 2023, doi: 10.1109/ACCESS.2023.3286853.
keywords: {Predictive models;Transformers;Task analysis;Generators;Tokenization;Social networking (online);Media;Headline generation;headline popularity dataset;headline popularity prediction;summarization;transformer models},

Objective

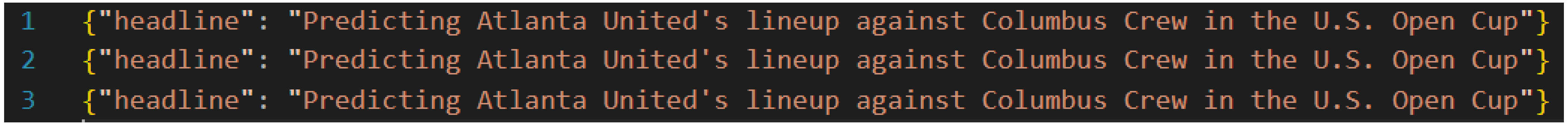
- In this homework, we aim to generate news headlines from news bodies. You should try to improve the headline quality to increase the performance.
- The data files can be found in the **E3** homework section.
- We will have **a two-stages submission**. The first stage is optional and is to check the current performance. The final result should be submitted to the second stage submission, which will be used to calculate your final score.

Dataset Files

- **train.json**: a text file with 100,000 json lines, where each line represents an individual row of data as follows:
  - **body**: news article in string format
  - **headline**: news title in string format
- **test.json**: a text file with 13,762 json lines, where each line represents an individual row of data as follows:
  - **body**: news article in string format
- **sample_submission.json**: a sample submission file with 13,762 json lines, where each line represents an individual row of data as follows:
  - **headline**: generated news title in string format

Submissions for 04/01 and 04/08 (Optional)

- Submit the prediction result named **‘{student_id}.json**’ to **E3**. Please ensure the format follows the **sample_submission.json.**
- It is recommended to submit the prediction within two checkpoint deadlines to check the current performance.



```
{"headline": "Predicting Atlanta United's lineup against Columbus Crew in the U.S. Open Cup"}
{"headline": "Predicting Atlanta United's lineup against Columbus Crew in the U.S. Open Cup"}
{"headline": "Predicting Atlanta United's lineup against Columbus Crew in the U.S. Open Cup"}
```

The first three lines of the sample submission

Submissions for 2024/04/15 (Deadline)

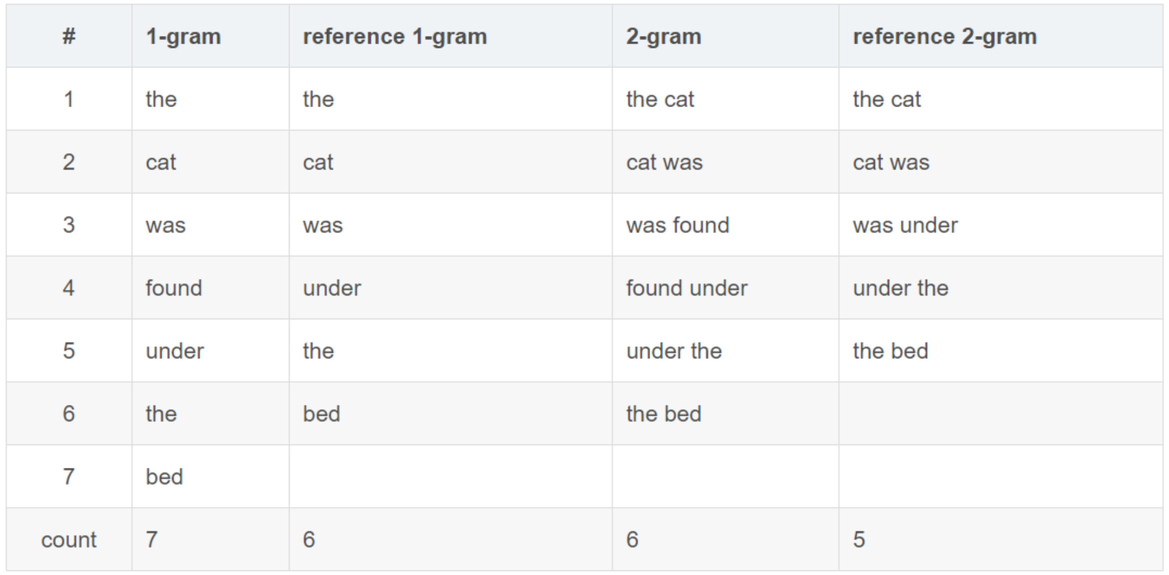
- Submit the prediction result named **‘{student_id}.json**’ to **E3**. Please ensure the format follows the **sample_submission.json** or you will get zero point**.**
- Submit the zipped source code named **‘{student_id}.zip**’ to **E3**. After unzipping, it should appear a folder {student_id} with the following structure:
  - {student_id}
    - {student_id}.sh: run this script should regenerate your final submission result.
    - requirements.txt: list the required libraries.
    - Other files.
    - No need to include the datasets. Therefore, please clearly specify the expected location of the datasets.

Evaluation Metrics - ROUGE

- **ROUGE** counts the number of overlapping units such as n-gram, word sequences, and word pairs between the computer-generated text to be evaluated and the references created by humans.

prediction: `the cat was found under the bed`

reference: `the cat was under the bed`



#	1-gram	reference 1-gram	2-gram	reference 2-gram
1	the	the	the cat	the cat
2	cat	cat	cat was	cat was
3	was	was	was found	was under
4	found	under	found under	under the
5	under	the	under the	the bed
6	the	bed	the bed	
7	bed			
count	7	6	6	5

$$Rouge\_1(X1, Y) = \frac{6}{6} = 1.0 \qquad Rouge\_2(X1, Y) = \frac{4}{5} = 0.8$$

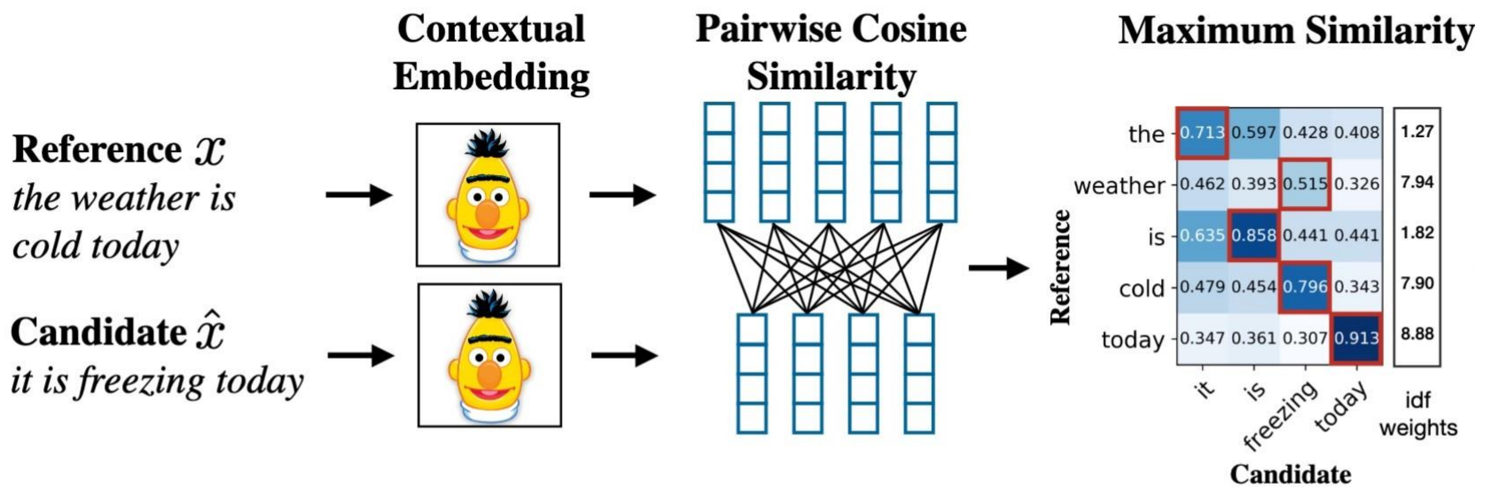
$$F_{\text{ROUGE}} = 2 * \frac{R_{\text{ROUGE}} P_{\text{ROUGE}}}{R_{\text{ROUGE}} + P_{\text{ROUGE}}}$$

$$R_{\text{ROUGE}} = \frac{\#\ of\ matched\ N - grams}{\#\ of\ N - grams\ in\ reference}$$

$$P_{\text{ROUGE}} = \frac{\#\ of\ matched\ N - grams}{\#\ of\ N - grams\ in\ prediction}$$

Evaluation Metrics - BERTScore

- **BERTScore** computes a similarity score for each token in the candidate sentence with each token in the reference sentence using contextual embeddings.



Contextual Embedding
Pairwise Cosine Similarity
Maximum Similarity
Reference x
the weather is cold today
Candidate $\hat{x}$
it is freezing today
Reference
Candidate
the 0.713 0.597 0.428 0.408
weather 0.462 0.393 0.515 0.326
is 0.635 0.858 0.441 0.441
cold 0.479 0.454 0.796 0.343
today 0.347 0.361 0.307 0.913
it
is
freezing
today
idf weights
1.27
7.94
1.82
7.90
8.88

$$F_{\text{BERT}} = 2\frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}$$

$$P_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^{\top} \hat{\mathbf{x}}_j$$

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^{\top} \hat{\mathbf{x}}_j$$

Grading Policy

- ROUGE (50 points) / BERTScore (50 points)
  - Top 10%: 100%
  - Top 25%: 90%
  - Top 50%: 80%
  - Top 75%: 70%
  - Other: 60%
  - Below baseline: 0%

Baseline

- Here is the baseline performance. Please try to beat it!

	Baseline
ROUGE 1	0.35
ROUGE 2	0.20
ROUGE L	0.30
BERTScore F1	0.85

Reference

- Huggingface: https://huggingface.co/docs
- Transformers: https://huggingface.co/docs/transformers/quicktour
- Tokenizers: https://huggingface.co/docs/tokenizers/quicktour
- Various pre-trained models: https://huggingface.co/models (NOTE: task-specified fine-tuned models are not allowed.)

```
>>> from datasets import load_dataset
>>> dataset = load_dataset(”json”, data_files=“train.json”)


>>> from transformers import AutoTokenizer
>>> tokenizer = AutoTokenizer.from_pretrained(“roberta-base”)


>>> def preprocess_func(examples):
...       return tokenizer(examples[“title”], padding=“max_length”, truncation=True)

>>> tokenized_dataset = dataset.map(preprocess_func, batched=True)
```

Reference

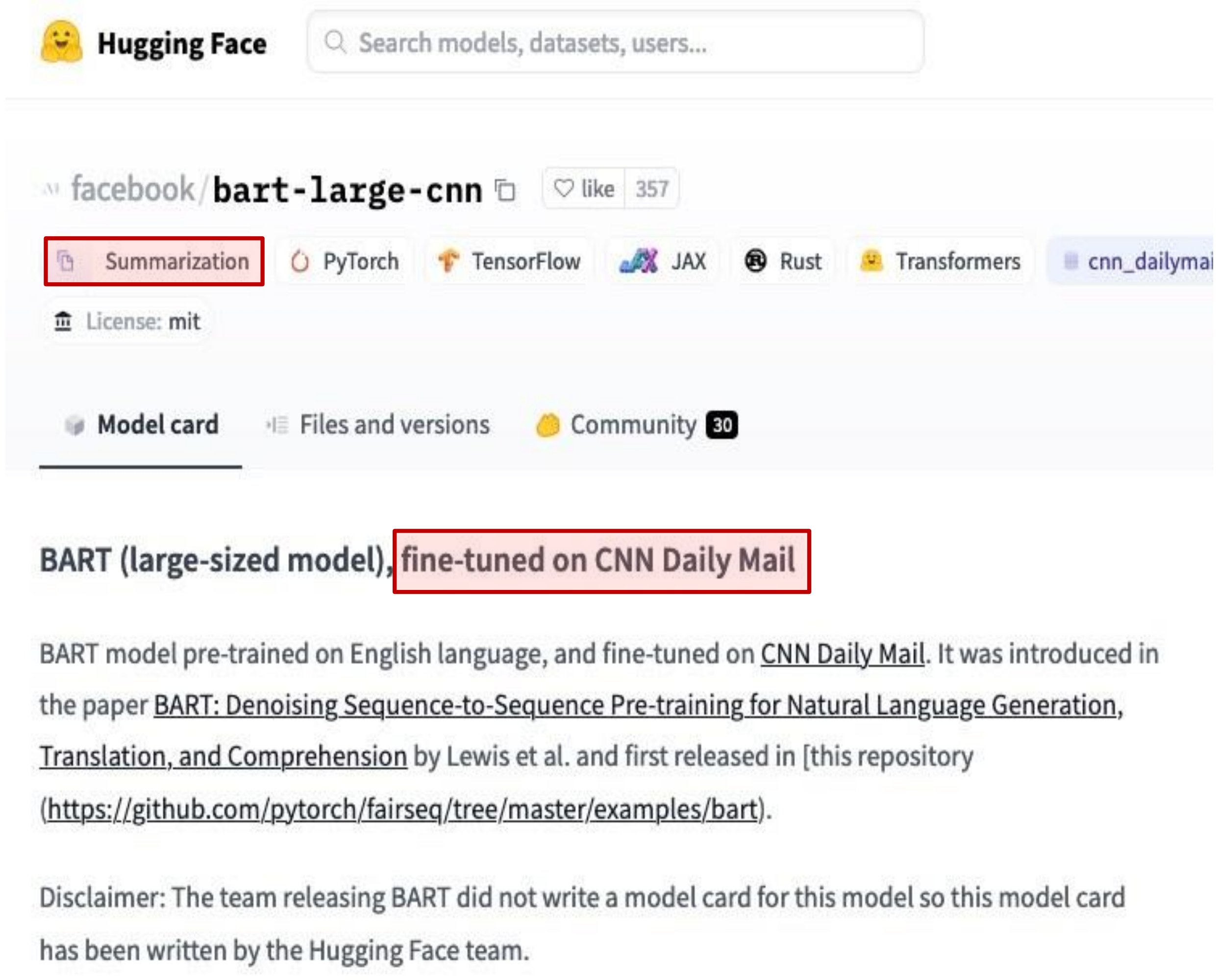
- Evaluate: https://huggingface.co/docs/evaluate/a_quick_tour

```
>>> import evaluate


>>> metric_rouge = evaluate.load("rouge", rouge_types=["rouge1", "rouge2", "rougeL"])
>>> metric_bertscore = evaluate.load("bertscore")

>>> rouge = metric_rouge.compute(predictions=preds, references=titles, use_stemmer=True)
>>> bertscore = metric_bertscore.compute(predictions=preds, references=titles, lang="en")
```

Reference

Example

- Not allowed pre-trained model



BART (large-sized model), fine-tuned on CNN Daily Mail

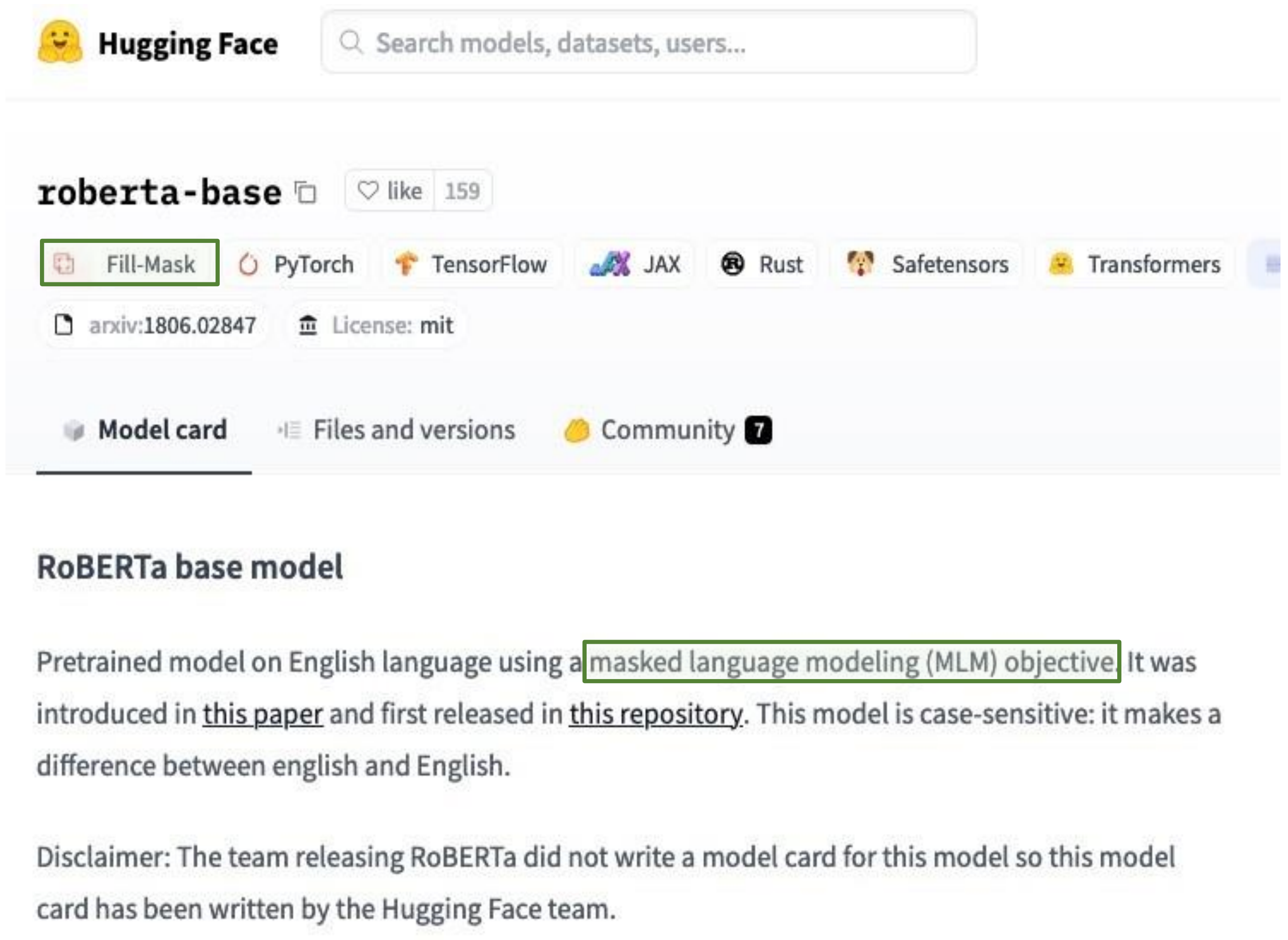
BART model pre-trained on English language, and fine-tuned on CNN Daily Mail. It was introduced in the paper BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension by Lewis et al. and first released in [this repository (https://github.com/pytorch/fairseq/tree/master/examples/bart).

Disclaimer: The team releasing BART did not write a model card for this model so this model card has been written by the Hugging Face team.

Reference

Example

- Allowed pre-trained model



Hugging Face Search models, datasets, users...

roberta-base like 159

Fill-Mask PyTorch TensorFlow JAX Rust Safetensors Transformers

arxiv:1806.02847 License: mit

Model card Files and versions Community 7

RoBERTa base model

Pretrained model on English language using a masked language modeling (MLM) objective. It was introduced in this paper and first released in this repository. This model is case-sensitive: it makes a difference between english and English.

Disclaimer: The team releasing RoBERTa did not write a model card for this model so this model card has been written by the Hugging Face team.

Rules

- Do not plagiarize. Write your own codes.
- Do not use additional datasets.
- You can use pre-trained language models as initialization. However, **fined-tuned models are not allowed**, i.e., using additional annotation for summarization or headline generation is not permitted.
- All kinds of library is allowed.

Information

- Checkpoint: 2024/04/01 and 2024/04/08 23:59 (Optional)
- Deadline: 2024/04/15 23:59
- Please post you question on the **E3 forum**
- [TA] Tzu-Ling Lin (林子淩): tzulinglin.11@nycu.edu.tw