Natural Language Processing with Transformers
6章 要約

2023/06/03

趙 将司（川口 将司）

要約タスクの特徴

系列変換タスク（seq2seq）で、Encoder-Decoder向きのタスク。
抽出的ではなく、抽象的であることが要求される。
定量的かつ汎用的なベンチマークの設定が困難。

▼ 教師データのサンプル：

article (string)	highlights (string)
"LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on Monday, but he insists the money won't cast a spell on him. Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the Phoenix" To the disappointment of gossip columnists around the world, the young actor says he has no plans to fritter his cash away on fast cars, drink and celebrity parties. "I don't plan to be one of those people who, as soon as they turn 18, suddenly buy themselves a massive sports car collection or something similar," he told an Australian interviewer earlier this month. "I don't think I'll be particularly extravagant. "The things I like buying are	"Harry Potter star Daniel Radcliffe gets £20M fortune as he turns 18 Monday . Young actor says he has no plans to fritter his cash away . Radcliffe's earnings from first five Potter films have been held in trust fund ."

出所：CNN/DailyMail 3.0.0 (https://huggingface.co/datasets/cnn_dailymail/viewer/3.0.0/train?row=0)

本章でベンチマークが比較されるモデル

中規模LLMのパフォーマンスを比較する。このうちT5, BART, PEGASUSはCNN/DailyMailに基づく要約タスクにファインチューニング済みの版。

モデル	開発元	Encoder / Decoder	パラメータ数	リリース
冒頭3文を抽出	-	-	-	-
GPT-2 (xl)	OpenAI	Decoderのみ	15.58億	Feb, 2019
T5 (Large)	Google	Encoder-Decoder	7.38億	Feb, 2020
BART (large)	Meta (Facebook)	Decoderのみ	4.06億	Oct, 2019
PEGASUS	Zhang et al., Google	Encoder-Decoder	5.68億	Dec, 2019

【参考】ネットワーク構造の比較

モデル	層数	埋め込み/隠れ状態	ヘッド数	パラメータ数
GPT-2 (xl)	48	1600	25	15.58億
T5 (Large)	24	1024 / 4096(ff)	16	7.38億
BART (large)	12	1024	16	4.06億
PEGASUS	16	1024	16	5.68億

# parameters	Exact formula	Approximate formula
Multi-head attention	$4(d_{model}^2 + d_{model})$	$4d_{model}^2$
Feed-forward network	$2d_{model}d_{ff} + d_{model} + d_{ff}$	$2d_{model}d_{ff}$
Layer normalization	$2d_{model}$	0
Transformer Encoder	$4d_{model}^2 + 2d_{model}d_{ff} + 9d_{model} + d_{ff}$	$4d_{model}^2 + 2d_{model}d_{ff}$
		or
		$12d_{model}^2 \approx 10d_{model}^2$
Transformer Decoder	$8d_{model}^2 + 2d_{model}d_{ff} + 15d_{model} + d_{ff}$	$8d_{model}^2 + 2d_{model}d_{ff}$
		or
		$16d_{model}^2 \approx 10d_{model}^2$

出所）
https://discuss.huggingface.co/t/summarization-on-long-documents/920

出所）
https://towardsdatascience.com/how-to-estimate-the-number-of-parameters-in-transformer-models-ca0f57d8dff0

要約の対象が長すぎる場合への対処

多くのTransformerのモデルでは入力サイズの上限が1,000トークン程度。この制約は長い文章を扱う要約タスクの障害になりがち。入力サイズを超える部分のテキストは切り捨てるのが標準的（かつ雑）な対処法。
その他にも予め文章の一部を抽出してから要約する方法や、入力サイズの限界までの文章を要約するのを文末まで繰り返す方法が考えられる（以下参照）。

pratikbhavsar Sep '20

You can try extractive summarisation followed by abstractive. In the extractive step you choose top k sentences of which you choose top n allowed till model max length.

Another way is to use successive abstractive summarisation where you summarise in chunk of model max length and then again use it to summarise till the length you want. This method will be super expensive.

You can also combine first + second method.

出所）https://discuss.huggingface.co/t/summarization-on-long-documents/920

【参考】GPT-2に要約させる方法

文末に”TL;DR:”を付与することで、要約を実施させることができる。
ただし、GPT-2に関しては要約タスクのファインチューニングが未実施のため、
出力の品質は決して高くない。

```
from transformers import pipeline, set_seed
set_seed(42)
pipe = pipeline("text-generation", model="gpt2-xl")
gpt2_query = sample_text + "\nTL;DR:\n"
pipe_out = pipe(gpt2_query, max_length=512, clean_up_tokenization_spaces=True)
summaries["gpt2"] = "\n".join(
    sent_tokenize(pipe_out[0]["generated_text"][len(gpt2_query) :]))
```

要約の出力結果の比較（1）

ground truth	冒頭3文	GPT-2
Mentally ill inmates in Miami are housed on the "forgotten floor" Judge Steven Leifman says most are there as a result of "avoidable felonies" While CNN tours facility, patient shouts: "I am the son of the president" Leifman says the system is unjust and he's fighting for change .	Editor's note: In our Behind the Scenes series, CNN correspondents share their experiences in covering news and analyze the stories behind the events. Here, Soledad O'Brien takes users inside a jail where many of the inmates are mentally ill. An inmate housed on the "forgotten floor," where many mentally ill inmates are housed in Miami before trial. MIAMI, Florida (CNN) -- The ninth floor of the Miami-Dade pretrial detention facility is dubbed the "forgotten floor."	Here's a more "interesting" story CNN.com was unable to access. This story in the Miami Herald is even more disturbing. A video report at 6:00 of Leifman's office shows a video of a mentally ill inmate breaking through the bars of a jail cell and making a break for it. The Miami Herald headline above tells you everything

要約の出力結果の比較（2）

T5 (Large)	BART (large)	PEGASUS
mentally ill inmates are housed on the ninth floor of a florida jail . most face drug charges or charges of assaulting an officer . judge says arrests often result from confrontations with police . one-third of all people in Miami-dade county jails are mental ill .	Mentally ill inmates are housed on the "forgotten floor" of Miami-Dade jail. Most often, they face drug charges or charges of assaulting an officer. Judge Steven Leifman says the arrests often result from confrontations with police. He says about one-third of all people in the county jails are mentally ill.	Mentally ill inmates in Miami are housed on the "forgotten floor"<n> The ninth floor is where they're held until they're ready to appear in court. Most often, they face drug charges or charges of assaulting an officer. They end up on the ninth floor severely mentally disturbed .

要約のベンチマーク

前述の通り、要約は客観的なベンチマークを示しにくい。合意形成の取れた普遍的な指標がなく、ヒューリスティックに手法が選択される。

手法	概要
BLEU	**生成文[*]中の**n-gramが参照文[*]に登場する数を集計する。精度ベース。
ROUGE	**参照文[*]中の**n-gramが生成文[*]に登場する数を集計する。再現率ベース。
NIST	BLEUの集計方法を踏襲しつつ、レアな語句を含む n-gramがマッチした際に加重する。
（目検）	文字通り、人間が生成文を読んで評価する。

* 生成結果に複数の文が含まれる場合、連結し1文と捉えて評価する。

BLEU（1）

生成文中の単語ならびにn-gramが参照文に登場する数を集計する。

【参照文】the cat is on the mat 【生成文】the the the the the the
→ 1-グラムのスコアは2/6。この要領で1,2,...,n-グラムまで繰り返す。

評価データセット（コーパス:C）に含まれる全サンプルに対するスコアを集計する。

$$p_n = \frac{\sum_{snt \in C} \sum_{n\text{-}gram \in snt} Count_{clip}(n\text{-}gram)}{\sum_{snt \in C} \sum_{n\text{-}gram \in snt} Count(n\text{-}gram)}$$

BLEU（2）

前述のスコア方法では、生成文が短いほど有利になる。この再現率の観点での不都合をカバーするため、Brevity Penalty（BP）を定義する。

$$BP = \min\left(1, e^{1-\ell_{ref}/\ell_{gen}}\right)$$

l_{ref}:参照文の長さ

l_{gen}:生成文の長さ

l_{gen}がl_{ref}よりも極端に小さな値をとる場合、指数項が極小値になる。
前述のスコア方法のペナルティ項としてBPを導入する。

$$\text{BLEU-}N = BP \times \left(\prod_{n=1}^{N} p_n\right)^{1/N} \quad \text{where} \quad p_n = \frac{\sum_{snt \in C} \sum_{n\text{-}gram \in snt} Count_{clip}(n\text{-}gram)}{\sum_{snt \in C} \sum_{n\text{-}gram \in snt} Count(n\text{-}gram)}$$

実用上ではBLUE-4がよく利用されるらしい。

SacreBLEU

BLEUは表記を重んじ、意味を考慮しない問題がある。つまり同義語や言い換え、語順の差異などに対処できない（結局のところ抽出的である）。加えてトークナイザの差異がモデル間の評価に影響を与えうる。

後者に関する善後策としてSacreBLEUの採用が考えられる。SacreBLEUでは、トークン化処理が組み込まれるため、トークナイザの差で有利不利が生じる状況を回避している。

```
reference=["the cat is on the mat"]
```

```
prediction="the the the the the the"
```

	Value
score	0.0
counts	[2, 0, 0, 0]
totals	[6, 5, 4, 3]
precisions	[33.33, 0.0, 0.0, 0.0]
bp	1.0
sys_len	6
ref_len	6

```
prediction="the cat is on mat"
```

	Value
score	57.893007
counts	[5, 3, 2, 1]
totals	[5, 4, 3, 2]
precisions	[100.0, 75.0, 66.67, 50.0]
bp	0.818731
sys_len	5
ref_len	6

ROUGE (ROUGE-N)

参照文中のn-gramが生成文に登場する数を集計する。
生成文中のn-gramが参照文に登場する数を評価したBLEUとは逆のアプローチである。

$$\text{ROUGE-}N = \frac{\sum_{\text{snt'} \in C} \sum_{n\text{-}gram \in \text{snt'}} Count_{match}(n\text{-}gram)}{\sum_{\text{snt'} \in C} \sum_{n\text{-}gram \in \text{snt'}} Count(n\text{-}gram)}$$

ROUGE-NはBLEU-Nと異なりBP項を持たず、さらにn-gramのnは1つだけ集計する（BLEUはn=1,2,...,Nの合計から幾何平均を取る）。

参考）https://stackoverflow.com/questions/38045290/text-summarization-evaluation-bleu-vs-rouge

ROUGE-N precision / recall / F1-score

N=1の例を示す。参照文と生成文として、それぞれ以下を想定する。
【参照文(R)】The bird is on the chair. (length=6)
【生成文(C)】The bird and the frog. (length=5)

手法	スコア計算方法
ROUGE-1 precision	CのうちRでマッチした1-gram数 / Cの長さ = 3/5 =0.6
ROUGE-1 recall	RのうちCでマッチした1-gram数 / Rの長さ = 3/6 = 0.5
ROUGE-1 F1-score	2 * (precision * recall) / (precision + recall) = 2 * (0.6 * 0.5) / (0.6 + 0.5) = 0.55

ROUGE-N precision / recall / F1-score

N=2の例を示す。参照文と生成文として、それぞれ以下を想定する。
【参照文(R)】The bird is on the chair. (length=5)
【生成文(C)】The bird and the frog. (length=4)

手法	スコア計算方法
ROUGE-2 precision	CのうちRでマッチした2-gram数 / Cの長さ = 1/4 =0.25
ROUGE-2 recall	RのうちCでマッチした2-gram数 / Rの長さ = 1/5 = 0.2
ROUGE-2 F1-score	2 * (precision * recall) / (precision + recall) = 2 * (0.25 * 0.2) / (0.25 + 0.2) ≒ 0.22

ROUGE-L

参照文と生成文との間で、語順に従いながら共起している単語の個数（Longest Common Subsequence / LCS）で評価を行う。**語順さえ一致していれば、単語の間に余計な単語が挟まってもカウントされる点に注意。**

summaries X of length m and Y of length n, assuming X is a reference summary sentence and Y is a candidate summary sentence, as follows:

$$R_{lcs} = \frac{LCS(X,Y)}{m} \quad (2)$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \quad (3)$$

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \quad (4)$$

where $\beta = P_{lcs}/R_{lcs}$

出所）ROUGE: A Package for Automatic Evaluation of Summaries; Lin (2004)

ROUGE-Lsum

ROUGE-Lでは一文毎に算出し平均を取るのに対し、ROUGE-Lsumは要約全体（すべての生成文の連結）に適用して算出したもの。
本章では当指標に基づき比較評価している。

```
rouge_names = ["rouge1", "rouge2", "rougeL", "rougeLsum"]
for model_name in summaries:
    rouge_metric.add(prediction=summaries[model_name], reference=reference)
    score = rouge_metric.compute()
    rouge_dict = dict((rn, score[rn].mid.fmeasure) for rn in rouge_names)
    records.append(rouge_dict)
pd.DataFrame.from_records(records, index=summaries.keys())
```

	rouge1	rouge2	rougeL	rougeLsum
baseline	0.365079	0.145161	0.206349	0.285714
gpt2	0.288288	0.018349	0.162162	0.288288
t5	0.382979	0.130435	0.255319	0.382979
pegasus	0.323232	0.206186	0.282828	0.323232

要約モデルの追加学習（ファインチューニング）

SAMsum（対話ログの要約）のタスクはCNN/DailyMailと異なり、短い対話形式かつground truthも抽象的。当設定でPEGASUSに下流タスクを習得させる。

SAMsumのデータ形式サンプル：

id (string)	dialogue (string)	summary (string)
"13818513"	"Amanda: I baked cookies. Do you want some? Jerry: Sure! Amanda: I'll bring you tomorrow…	"Amanda baked cookies and will bring Jerry some tomorrow."
"13728867"	"Olivia: Who are you voting for in this election? Oliver: Liberals as always. Olivia: M…	"Olivia and Olivier are voting for liberals in this election. "
"13681000"	"Tim: Hi, what's up? Kim: Bad mood tbh, I was going to do lots of stuff but ended up…	"Kim may try the pomodoro technique recommended by Tim to get more stuff done."
"13730747"	"Edward: Rachel, I think I'm in ove with Bella.. rachel: Dont say anything else.. Edward: What d…	"Edward thinks he is in love with Bella. Rachel wants Edward to open his door. Rachel is…
"13728094"	"Sam: hey overheard rick say something Sam: i don't know what to do :-/ Naomi: what did he	"Sam is confused, because he overheard Rick complaining about him as a roommate. Naomi

PEGASUS（ベースモデル）の要約結果：

```
Summary:
Amanda: Ask Larry Amanda: He called her last time we were at the park together.
Hannah: I'd rather you texted him.
Amanda: Just text him .
```

	rouge1	rouge2	rougeL	rougeLsum
pegasus	0.296038	0.087469	0.229174	0.229574

トークナイザ

対話文（要約前）と要約文の最大長をそれぞれ1024/128に設定。
モデル次第ではDecoderに対して特殊なトークンの挿入が生じる場合があり、処理対象をEncoderから峻別するためwith構文（context manager）が利用される。

```
model_ckpt = "google/pegasus-cnn_dailymail"
tokenizer = AutoTokenizer.from_pretrained(model_ckpt)
```

```
dataset_samsum = load_dataset("samsum")
```

```
def convert_examples_to_features(example_batch):
    input_encodings = tokenizer(example_batch["dialogue"], max_length=1024, truncation=True)

    with tokenizer.as_target_tokenizer():
        target_encodings = tokenizer(example_batch["summary"], max_length=128, truncation=True)

    return {"input_ids": input_encodings["input_ids"], "attention_mask": input_encodings["attention_mask"], "labels": target_encodings["input_ids"]}

dataset_samsum_pt = dataset_samsum.map(convert_examples_to_features, batched=True)
columns = ["input_ids", "labels", "attention_mask"]
dataset_samsum_pt.set_format(type="torch", columns=columns)
```

データコレーター（Data Collator）

モデルにデータを与える直前に呼び出される機構。トークンを1つずつ積み上げてモデルの入力（と出力）に与えていく。
要約タスクはseq2seqの構造なので、入力だけでなく出力のラベル（要約文側のトークン）もDecoder側へ1つずつ与える（PEGASUSはEncoder-Decoder）。

```
model = AutoModelForSeq2SeqLM.from_pretrained(model_ckpt).to(device)
```

```
from transformers import DataCollatorForSeq2Seq

seq2seq_data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)
```

```
from transformers import TrainingArguments, Trainer
training_args = TrainingArguments(
    output_dir='pegasus-samsum', num_train_epochs=1, warmup_steps=500,
    per_device_train_batch_size=1, per_device_eval_batch_size=1,
    weight_decay=0.01, logging_steps=10, push_to_hub=True, evaluation_strategy='steps',
    eval_steps=500, save_steps=1e6, gradient_accumulation_steps=16)
```

```
trainer = Trainer(model=model, args=training_args,
                  tokenizer=tokenizer, data_collator=seq2seq_data_collator,
                  train_dataset=dataset_samsum_pt["train"],
                  eval_dataset=dataset_samsum_pt["validation"])
```

	decoder_input	label
step		
1	[PAD]	Transformers
2	[PAD, Transformers]	are
3	[PAD, Transformers, are]	awesome
4	[PAD, Transformers, are, awesome]	for
5	[PAD, Transformers, are, awesome, for]	text
6	[PAD, Transformers, are, awesome, for, text]	summarization

学習結果

精度が向上したのはもちろん、要約の文体も対話を合成した抽象的なものに。

```
trainer.train()
score = evaluate_summaries_pegasus(
    dataset_samsum["test"], rouge_metric, trainer.model, tokenizer,
    batch_size=2, column_text="dialogue", column_summary="summary")
rouge_dict = dict((rn, score[rn].mid.fmeasure) for rn in rouge_names)
pd.DataFrame(rouge_dict, index=[f"pegasus"])
```

	rouge1	rouge2	rougeL	rougeLsum
pegasus	0.296038	0.087469	0.229174	0.229574

	rouge1	rouge2	rougeL	rougeLsum
pegasus	0.426183	0.196296	0.338714	0.338947

```
Summary:
Amanda: Ask Larry Amanda: He called her last time we were at the park together.
Hannah: I'd rather you texted him.
Amanda: Just text him .
```

```
Model Summary:
Amanda can't find Betty's number. Larry called Betty last time they were at the park together. Hannah wants Amanda to text Larry instead of calling Betty.
```