2021-2 텍스트 및 자연어 빅데이터 분석방법론

Huggingface Transformers II

Text Generation, Named Entity Recognition, Summarization, Translation

산업공학과 남유리

Text Generation

- Text generation is the task of *creating coherent portion of text* that is a continuation from the given context.
- Simple text generation example with pipeline
  - Step 1: Install & Import Library

```
pip install transformers
```

```
from transformers import pipeline
```

  - Step 2: Build Text Generation Pipeline

```
generator = pipeline('text-generation', model='gpt2')
```

  - Step 3: Define the Text to Start Generating From

```
text  = "Today the weather is really nice and I am planning on"
```

  - Step 4: Start Generating

```
generated_text = generator(text, max_length=50, do_sample=False)
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
```

```
print(generated_text)
```

```
[{'generated_text': 'Today the weather is really nice and I am planning on going to the beach and going to the beach with my friends. I am going to go to the beach with my friends and I am going to go to the beach with my friends. I am'}]
```

text generation result

Text Generation: Decoding Methods

- Decoding methods for text generation with Transformers

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer

tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
# EOS토큰을 PAD토큰으로 지정하여 warning이 나오지 않도록 함
model = GPT2LMHeadModel.from_pretrained("gpt2", pad_token_id=tokenizer.eos_token_id)
```

GPT-2 example

```
# 생성할 텍스트에 대한 시작 문구를 지정
text  = "Today the weather is really nice and I am planning on "
input_ids = tokenizer.encode(text, return_tensors='pt')
```

  - Greedy Search: 타입 스텝 t에서 가장 높은 확률을 가지는 토큰을 다음 토큰으로 선택하는 전략

```
# 디코딩 최대 길이(50글자)까지 model.generate를 통해 디코딩 진행
greedy_output = model.generate(input_ids, max_length=50)
```

```
Output:
----------------------------------------------------------------------------------------------------
Today the weather is really nice and I am planning on  going to go to the beach and get some sun. I am going to go to the be
ach and get some sun. I am going to go to the beach and get some sun.
```

  - Beam Search: 각 타임스텝에서 가장 가능성 있는 num_beams개의 시퀀스를 유지하고, 최종적으로 가장 확률이 높은 가설을 선택하는 방법

```
# Beam-search를 적용하고 EOS가 나오면 생성을 멈춤
beam_output = model.generate(
    input_ids,
    max_length=50,
    num_beams=5, #1보다 큰 값을 지정
    early_stopping=True #EOS토큰이 나오면 생성을 중단
)
```

```
Output:
----------------------------------------------------------------------------------------------------
Today the weather is really nice and I am planning on  going to go out on a limb and go out on a limb and go out on a limb a
nd go out on a limb and go out on a limb and go out on a limb
```

Text Generation: Decoding Methods

- Decoding methods for text generation with Transformers
  - Top-k Sampling: 가장 확률이 높은 k개의 다음 단어들을 필터링하고, 확률 질량을 해당 k 개의 다음 단어들에 대해 재분배하는 전략

```
sample_output = model.generate(
    input_ids,
    do_sample=True, # 샘플링 사용
    max_length=50,
    top_k=10 #top_k=0으로 설정하면 타임스텝별로 하나의 토큰만 샘플링
)
```

```
Output:
----------------------------------------------------------------------------------------------------
Today the weather is really nice and I am planning on  going to go to the beach in the summer. I am planning on going to tak
e a break from doing the things I am doing and going back to doing the things I love doing.
```

  - Top-p Sampling: 가능성 있는 k 개의 단어로부터 샘플링하는 대신, 누적 확률이 확률 p에 다다르는 최소한의 단어 집합으로부터 샘플링하는 전략

```
# deactivate top_k sampling and sample only from 92% most likely words
sample_output = model.generate(
    input_ids,
    do_sample=True,
    max_length=50,
    top_p=0.92, #92%로 설정하고 샘플링하기
    top_k=0
)
```

```
Output:
----------------------------------------------------------------------------------------------------
Today the weather is really nice and I am planning on  moving to the Guadalupe ski area from Long Beach last year.
Geralp Maples
Photography by James L. Kazischner of Colyo.
```

Text Generation using Pre-trained Model

- Generating text with XLNet

```
from transformers import AutoModelForCausalLM, AutoTokenizer
```

```
model = AutoModelForCausalLM.from_pretrained("xlnet-base-cased")
tokenizer = AutoTokenizer.from_pretrained("xlnet-base-cased")
```

```
# Padding text helps XLNet with short prompts - proposed by Aman Rusia in https://github.com/rusiaaman/XLNet-gen#methodology
PADDING_TEXT = """In 1991, the remains of Russian Tsar Nicholas II and his family
... (except for Alexei and Maria) are discovered.
... The voice of Nicholas's young son, Tsarevich Alexei Nikolaevich, narrates the
... remainder of the story. 1883 Western Siberia,
... a young Grigori Rasputin is asked by his father and a group of men to perform magic.
... Rasputin has a vision and denounces one of the men as a horse thief. Although his
... father initially slaps him for making such an accusation, Rasputin watches as the
... man is chased outside and beaten. Twenty years later, Rasputin sees a vision of
... the Virgin Mary, prompting him to become a priest. Rasputin quickly becomes famous,
... with people, even a bishop, begging for his blessing. <eod> </s> <eos>"""
```

```
prompt = "Today the weather is really nice and I am planning on "
inputs = tokenizer(PADDING_TEXT + prompt, add_special_tokens=False, return_tensors="pt")["input_ids"]
```

```
prompt_length = len(tokenizer.decode(inputs[0]))
outputs = model.generate(inputs, max_length=250, do_sample=True, top_p=0.95, top_k=60)
generated = prompt + tokenizer.decode(outputs[0])[prompt_length+1:]
```

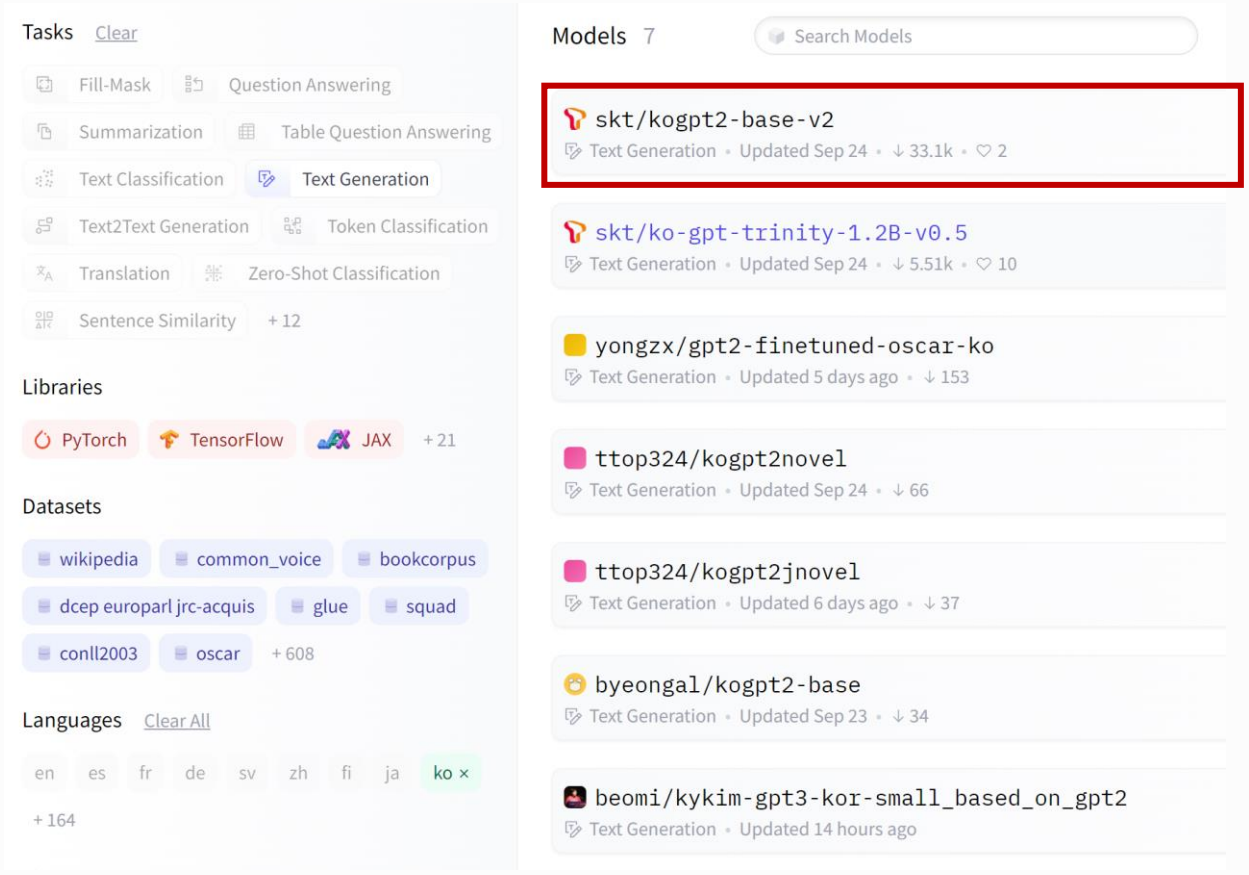
```
print(generated)
```

```
Today the weather is really nice and I am planning on blogging about all my projects for my upcoming year with the other pro
jects on the blog as well as doing some fun projects for the upcoming project with the other blogger. I have also done many
projects
```

Text Generation using Pre-trained Model

- Generating text (in any language)
  - We need a language model previously trained on a corpus on that language

    https://huggingface.co/models?sort=downloads&p=1



Text Generation using Pre-trained Model

```
from transformers import PreTrainedTokenizerFast

tokenizer = PreTrainedTokenizerFast.from_pretrained("skt/kogpt2-base-v2",
bos_token='</s>', eos_token='</s>', unk_token='<unk>',
pad_token='<pad>', mask_token='<mask>')

tokenizer.tokenize("안녕하세요. 한국어 GPT-2 입니다.😪:)l^o")
#['_안녕', '하', '세', '요.', '_한국어', '_G', 'P', 'T', '-2', '_입', '니다.', '😪', ':)', 'l^o']
```

Korean GPT-2

```
['_안녕',
 '하',
 '세',
 '요.',
 '_한국어',
 '_G',
 'P',
 'T',
 '-2',
 '_입',
 '니다.',
 '😪',
 ':)',
 'l^o']
```

```
import torch
from transformers import GPT2LMHeadModel
```

```
model = GPT2LMHeadModel.from_pretrained('skt/kogpt2-base-v2')
```

```
Downloading:   0%|          | 0.00/490M [00:00<?, ?B/s]
```

```
text = '근육이 커지기 위해서는'
input_ids = tokenizer.encode(text)
```

```
input_ids
```

```
[33245, 10114, 12748, 11357]
```

```
gen_ids = model.generate(torch.tensor([input_ids]),
                         max_length=128,
                         repetition_penalty=2.0,
                         pad_token_id=tokenizer.pad_token_id,
                         eos_token_id=tokenizer.eos_token_id,
                         bos_token_id=tokenizer.bos_token_id,
                         use_cache=True)
```

```
generated = tokenizer.decode(gen_ids[0,:].tolist())
```

```
print(generated)
```

```
근육이 커지기 위해서는 무엇보다 규칙적인 생활습관이 중요하다.
특히, 아침식사는 단백질과 비타민이 풍부한 과일과 채소를 많이 섭취하는 것이 좋다.
또한 하루 30분 이상 충분한 수면을 취하는 것도 도움이 된다.
아침 식사를 거르지 않고 규칙적으로 운동을 하면 혈액순환에 도움을 줄 뿐만 아니라 신진대사를 촉진해 체내 노폐물을 배출하고 혈압을 낮춰준다.
운동은 하루에 10분 정도만 하는 게 좋으며 운동 후에는 반드시 스트레칭을 통해 근육량을 늘리고 유연성을 높여야 한다.
운동 후 바로 잠자리에 드는 것은 피해야 하며 특히 아침에 일어나면 몸이 피곤해지기 때문에 무리하게 움직이면 오히려 역효과가 날 수도 있다.
운동을
```

(FYI) Text Generation Model Comparison

- GPT-2 vs XLNet
  - GPT-2 uses a novel byte pair encoding which operates on utf-8 byte sequences themselves, but XLNet uses byte pair encoding of SentencePiece library which operates on Unicode strings. Because of this, GPT-2 can assign probability to any sequence of characters. XLNet has restricted vocabulary, doesn't handle multi-lingual characters or emojis. This is the reason we see <unk> being generated from time to time with XLNet-gen.
  - GPT-2 is trained on web scrapped text (reddit curated) which amounts to 40GB of data. XLNet is trained on multiple datasets which amount to 136 GB of data.
  - GPT-2 pre-trained model with 365M parameters has the same number of parameters as the largest released XLNet model.
  - GPT-2 models text left to right, but XLNet can model it in any permutation possible. However, during generation the current implementation of XLNet-gen uses only left-to-right decoding.

Named Entity Recognition

- Named entity recognition is the task of *classifying tokens according to a class*, for example, identifying a token as a person, an organization or a location.
- Simple named entity recognition example with pipeline

```
>>> from transformers import pipeline

>>> ner_pipe = pipeline("ner")

>>> sequence = """Hugging Face Inc. is a company based in New York City. Its headquarters are in DUMBO,
... therefore very close to the Manhattan Bridge which is visible from the window."""
>>> for entity in ner_pipe(sequence):
...     print(entity)
{'entity': 'I-ORG', 'score': 0.9996, 'index': 1, 'word': 'Hu', 'start': 0, 'end': 2}
{'entity': 'I-ORG', 'score': 0.9910, 'index': 2, 'word': '##gging', 'start': 2, 'end': 7}
{'entity': 'I-ORG', 'score': 0.9982, 'index': 3, 'word': 'Face', 'start': 8, 'end': 12}
{'entity': 'I-ORG', 'score': 0.9995, 'index': 4, 'word': 'Inc', 'start': 13, 'end': 16}
{'entity': 'I-LOC', 'score': 0.9994, 'index': 11, 'word': 'New', 'start': 40, 'end': 43}
{'entity': 'I-LOC', 'score': 0.9993, 'index': 12, 'word': 'York', 'start': 44, 'end': 48}
{'entity': 'I-LOC', 'score': 0.9994, 'index': 13, 'word': 'City', 'start': 49, 'end': 53}
{'entity': 'I-LOC', 'score': 0.9863, 'index': 19, 'word': 'D', 'start': 79, 'end': 80}
{'entity': 'I-LOC', 'score': 0.9514, 'index': 20, 'word': '##UM', 'start': 80, 'end': 82}
{'entity': 'I-LOC', 'score': 0.9337, 'index': 21, 'word': '##BO', 'start': 82, 'end': 84}
{'entity': 'I-LOC', 'score': 0.9762, 'index': 28, 'word': 'Manhattan', 'start': 114, 'end': 123}
{'entity': 'I-LOC', 'score': 0.9915, 'index': 29, 'word': 'Bridge', 'start': 124, 'end': 130}
```

Named Entity Recognition

- Named entity recognition using pre-trained model
  - Step 1: Instantiate a tokenizer and a model

```
>>> from transformers import AutoModelForTokenClassification, AutoTokenizer
>>> import torch

>>> model = AutoModelForTokenClassification.from_pretrained("dbmdz/bert-large-cased-finetuned-conll03-english")
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
```

  - Step 2: Define a sequence with known entities

```
>>> sequence = "Hugging Face Inc. is a company based in New York City. Its headquarters are in DUMBO, " \
...            "therefore very close to the Manhattan Bridge."
```

  - Step 3: Split words into tokens and encode that sequence

```
>>> inputs = tokenizer(sequence, return_tensors="pt")
>>> tokens = inputs.tokens()
```

  - Step 4: Retrieve the predictions and get the first output

```
>>> outputs = model(**inputs).logits
>>> predictions = torch.argmax(outputs, dim=2)
```

  - Step 5: Zip each token with its prediction

```
>>> for token, prediction in zip(tokens, predictions[0].numpy()):
...     print((token, model.config.id2label[prediction]))
```

NER Result

```
('[CLS]', 'O')
('Hu', 'I-ORG')
('##gging', 'I-ORG')
('Face', 'I-ORG')
('Inc', 'I-ORG')
('.', 'O')
('is', 'O')
('a', 'O')
('company', 'O')
('based', 'O')
('in', 'O')
('New', 'I-LOC')
('York', 'I-LOC')
('City', 'I-LOC')
('.', 'O')
('Its', 'O')
('headquarters', 'O')
('are', 'O')
('in', 'O')
('D', 'I-LOC')
('##UM', 'I-LOC')
('##BO', 'I-LOC')
(',', 'O')
('therefore', 'O')
('very', 'O')
('close', 'O')
('to', 'O')
('the', 'O')
('Manhattan', 'I-LOC')
('Bridge', 'I-LOC')
('.', 'O')
('[SEP]', 'O')
```

Summarization

- Summarization is the task of *shortening long pieces of text into a concise summary* that preserves key information content and overall meaning
- Two different approaches
  - Extractive summarization: the model identifies the important sentences and phrases from the original text and only outputs those
  - Abstractive summarization: the model produces a completely different text that is shorter that the original, it generates new sentences in a new form.

Text Summarization using Pre-trained Model

- Text summarization using T5
  - Step 1: Instantiate a tokenizer and a model

```
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
model = AutoModelForSeq2SeqLM.from_pretrained("t5-base")
tokenizer = AutoTokenizer.from_pretrained("t5-base")
```

  - Step 2: Define the article to be summarized

```
ARTICLE = """ New York (CNN)When Liana Barrientos was 23 years old, she got married in Westchester County, New York.
A year later, she got married again in Westchester County, but to a different man and without divorcing her first husband.
Only 18 days after that marriage, she got hitched yet again. Then, Barrientos declared "I do" five more times, sometimes only within two weeks of each other.
In 2010, she married once more, this time in the Bronx. In an application for a marriage license, she stated it was her "first and only" marriage.
Barrientos, now 39, is facing two criminal counts of "offering a false instrument for filing in the first degree," referring to her false statements on the
2010 marriage license application, according to court documents.
Prosecutors said the marriages were part of an immigration scam.
On Friday, she pleaded not guilty at State Supreme Court in the Bronx, according to her attorney, Christopher Wright, who declined to comment further.
After leaving court, Barrientos was arrested and charged with theft of service and criminal trespass for allegedly sneaking into the New York subway through an
Annette Markowski, a police spokeswoman. In total, Barrientos has been married 10 times, with nine of her marriages occurring between 1999 and 2002.
All occurred either in Westchester County, Long Island, New Jersey or the Bronx. She is believed to still be married to four men, and at one time, she was marri
Prosecutors said the immigration scam involved some of her husbands, who filed for permanent residence status shortly after the marriages.
Any divorces happened only after such filings were approved. It was unclear whether any of the men will be prosecuted.
The case was referred to the Bronx District Attorney\'s Office by Immigration and Customs Enforcement and the Department of Homeland Security\'s
Investigation Division. Seven of the men are from so-called "red-flagged" countries, including Egypt, Turkey, Georgia, Pakistan and Mali.
Her eighth husband, Rashid Rajput, was deported in 2006 to his native Pakistan after an investigation by the Joint Terrorism Task Force.
If convicted, Barrientos faces up to four years in prison.  Her next court appearance is scheduled for May 18.
"""
# T5 uses a max_length of 512 so we cut the article to 512 tokens.
inputs = tokenizer("summarize: " + ARTICLE, return_tensors="pt", max_length=512, truncation=True)
```

  - Step 3: Add the specific prefix “summarize: “

```
outputs = model.generate(
    inputs["input_ids"], max_length=150, min_length=40, length_penalty=2.0, num_beams=4, early_stopping=True
)
```

  - Step 4: Generate the summary

```
>>> print(tokenizer.decode(outputs[0]))
<pad> prosecutors say the marriages were part of an immigration scam. if convicted, barrientos faces two criminal
counts of "offering a false instrument for filing in the first degree" she has been married 10 times, nine of them
between 1999 and 2002.</s>
```

Text Summarization using Pre-trained Model

- Text summarization (in Korean) using koBART

```python
from transformers import PreTrainedTokenizerFast, BartForConditionalGeneration

#  Load Model and Tokenize
tokenizer = PreTrainedTokenizerFast.from_pretrained("ainize/kobart-news")
model = BartForConditionalGeneration.from_pretrained("ainize/kobart-news")
```

Korean BART

```python
# Encode Input Text
input_text = '국내 전반적인 경기침체로 상가 건물주의 수익도 전국적인 감소세를 보이고 있는 것으로 나타났다. 수익형 부동산 연구
input_ids = tokenizer.encode(input_text, return_tensors="pt")
```

```python
# Generate Summary Text Ids
summary_text_ids = model.generate(
    input_ids=input_ids,
    bos_token_id=model.config.bos_token_id,
    eos_token_id=model.config.eos_token_id,
    length_penalty=2.0,
    max_length=142,
    min_length=56,
    num_beams=4,
)
```

```python
# Decoding Text
print(tokenizer.decode(summary_text_ids[0], skip_special_tokens=True))
```

```
전반적인 경기 침체로 건물주 수익이 전국적인 감소세를 보이면서 전국 중대형 상가 순영업소득이 3분기 2만5800원으로 감소했으며
조현택 상가정보연구소 연구원은 앞으로 지역, 콘텐츠에 따른 상권 양극화 현상이 심화될 것으로 보인다고 밝혔다.
```

Translation

- Translation is the task of *translating a text from one language to another*.
- Simple translation example using pipeline

```
from transformers import pipeline
```

```
translation = pipeline("translation_en_to_de")
## same with
## translation = pipeline("translation_en_to_de", model="t5-base", tok
enizer="t5-base")
```

```
text = "I like to study Data Science and Machine Learning"
```

```
translated_text = translation(text, max_length=40)[0]['translation_tex
t']
print(translated_text)
```

```
Ich studiere gerne Datenwissenschaft und maschinelles Lernen
```

Translation using Pre-trained Model

- Translate Any Two Languages (Korean to English)

Korean to English Translation Example

```
from transformers import MarianTokenizer, MarianMTModel
```

```
lang = "ko"
target_lang = "en"
model_name = f'Helsinki-NLP/opus-mt-{lang}-{target_lang}'
```

```
# Download the model and the tokenizer
model = MarianMTModel.from_pretrained(model_name)
tokenizer = MarianTokenizer.from_pretrained(model_name)
```

```
text = "HuggingFace는 최근 헬싱키 대학의 1,000개 이상의 번역 모델을 트랜스포머 모델에 통합하였습니다. 번역 시스템을 구축하는 것은 라이브러리
```

```
# Tokenize the text
batch = tokenizer([text], return_tensors="pt", padding=True)
```

```
# Make sure that the tokenized text does not exceed the maximum
# allowed size of 512
batch["input_ids"] = batch["input_ids"][:, :512]
batch["attention_mask"] = batch["attention_mask"][:, :512]
```

```
# Perform the translation and decode the output
translation = model.generate(**batch)
tokenizer.batch_decode(translation, skip_special_tokens=True)
```

```
['HuggingFace recently integrated more than 1,000 translation models from Helsinki University into the Transform Model, which is as simple
as copying documents from libraries.']
```

Q&A

Thank You