구름 AI 자연어처리 전문가 양성과정 2기

영-한 번역기 프로젝트

구르미팀(1조)
박주은, 홍진희

목차

1. 프로젝트 개요 / 2. 프로젝트 팀 구성 및 역할

프로젝트 핵심사항

1) Test1, Test2 domain difference

2) Encoder-decoder model

3) Data augumentation by using Back translation

프로젝트 진행 프로세스

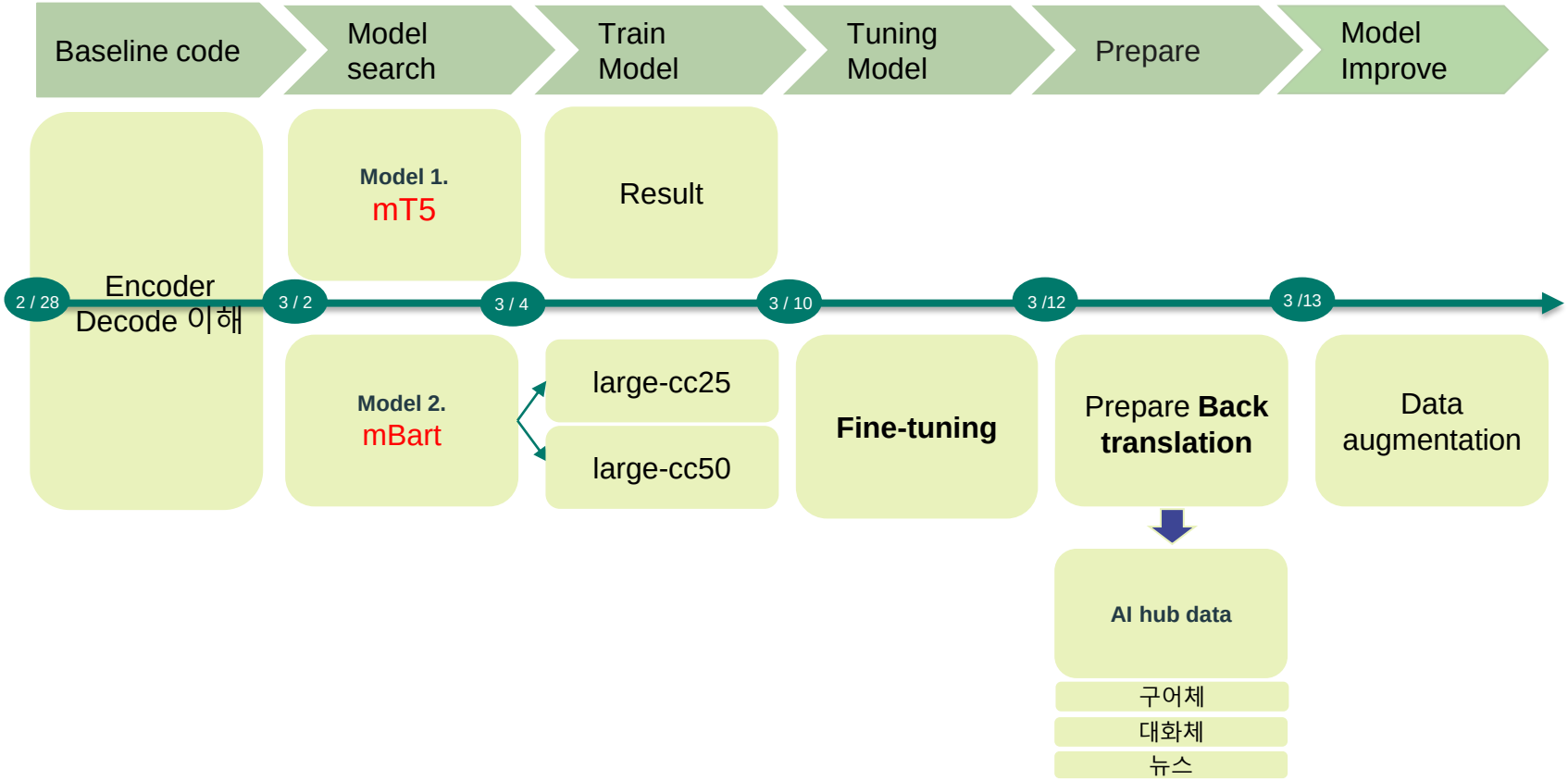
Used model: mbart

활용라이브러리 및 프레임워크

pytorch, SimpleTransformers, Transformers, Pandas, AI hub

훈련생	역할	담당
홍진희	팀원	mBart 25모델 적용 ai hub 데이터셋 추가한 모델 적용 하이퍼파라미터 튜닝 디버깅, 발표
박주은	팀원	mT5모델 mBart 50 모델 적용 ai hub 데이터셋 추가한 모델 적용 하이퍼파라미터 튜닝 디버깅

3. 프로젝트 진행 프로세스



3. Model : mT5

- mT5는 101개 언어 데이터 세트에 대해 사전 학습된 Google T5 모델의 다국어 변형인 mT5는 3억 ~ 130억 개의 매개 변수를 포함해 100개 이상의 언어를 학습할 수 있는 모델

**** Model CODE**

```
model_args = T5Args()
model_args.max_seq_length = 128
model_args.train_batch_size = 8
model_args.eval_batch_size = 8
model_args.num_train_epochs = 3
model_args.evaluate_during_training = True
model_args.evaluate_during_training_steps = 3000
model_args.use_multiprocessing = False
model_args.fp16 = False
model_args.save_steps = -1
model_args.save_eval_checkpoints = False
model_args.no_cache = True
model_args.reprocess_input_data = True
model_args.overwrite_output_dir = True
model_args.preprocess_inputs = False
model_args.num_return_sequences = 1
model_args.wandb_project = "en-ko_translation"

model = T5Model("mt5", "google/mt5-base", use_cuda=True, args=model_args)
```

***Simple Transformers?**
: 훈련 전달인자만 전달하면 알아서 훈련하는 시스템

```
model.train_model(train_df2, eval_data=eval_df2)
```

▶장점

*Simple Transformer를 사용함으로서 모델을 선언하고 훈련하는데 간단한 코드 작성으로 끝낼 수 있다. (Convenient)

▶단점

- 메모리 문제로 max_seq_length, batch_size에 대한 실험이 어렵다.

- 훈련 시간이 오래 걸린다.

3. Model : Limitation of mT5

Test1	Test2	제출일시	HM(Test1,Test2)
0.1960964643	0.134175654	3월10일 19:07	0.1593314718
0.2110145662	0.1577592851	3월10일 19:07	0.1805415812
0.2024448071	0.162674098	3월10일 20:28	0.1803934331

mBart process

import transformers

▼

Tokenizing

▼

훈련 전달인자 선언

▼

모델 선언 및 훈련

Takes 3hours

mT5 process

import simpletransformers

▼

훈련 전달인자 선언

Gpu 한계:
-Max length 조절 불가
-Batch size 조절 불가 등

▼

모델 초기화 및 훈련

Takes 7hours

3. Model : mBart-cc25 & 50

- **mBart 의 장점 :**

- Self supervised learning
  - **Masked language model**
  - **Denosing autoencoder (양방향 문맥)**

- Encoder-Decoder가 같이 존재

- Max position embeddings = 1024

- mBart-large-cc25 & mBart-large-50 차이

“”” MBart-50 is created using the original mbart-large-cc25 checkpoint by extendeding its embedding layers with randomly initialized vectors for an extra set of 25 language tokens and then pretrained on 50 languages. “””

* Tokenizer result :

cc25

```
Source input ids : [[9563, 4527, 10, 16777, 147, 2729, 53, 6056, 64457, 5, 2, 250004, 1, 1, 1, 1, 1, 1, 1,
Target input ids : [[2625, 124601, 16777, 147, 211032, 413, 162993, 32265, 6, 243797, 1875, 5, 2, 250014,
```

Eng tokenizer

Kor tokenizer

large-50

```
'src_input_ids': tensor([250004,   9563,   4527,     10,  16777,    147,   2729,     53,   6056,
         64457,      5,      2,      1,      1,      1,      1,      1,      1,
'tgt_input_ids': tensor([250014,   2625, 124601,  16777,    147, 211032,    413, 162993,  32265,
             6, 243797,   1875,      5,      2,      1,      1,      1,      1,
```

Eng tokenizer

Kor tokenizer

3. Model : mBart debugging

** **Data Loader** 부분 (train)

‘’’ 사용자 정의 Datatset class는 반드시 3개 함수를 구현해야 한다 : __init__, __len__, and __getitem__ ‘’’

```
#Dataloader
import torch

class Dataset(torch.utils.data.Dataset):
    def __init__(self, encodings):
        self.encodings = encodings

    def __getitem__(self, idx):
        return {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}

    def __len__(self):
        return len(self.encodings['src_input_ids'])
#TRAIN
train_dataset = Dataset(tokenized_train)

#VALID
dev_dataset = Dataset(tokenized_dev)
```

```
from torch.utils.data import DataLoader, Dataset
from transformers import  AdamW

epochs = 4
batch_size = 64
accumulation = 16

train_loader = DataLoader(train_dataset, batch_size=batch_size//accumulation,shuffle=True)
```

4. Project result

mT5		mBart (cc25)		mBart(large-50)		Tuning
test1	test2	test1	test2	test1	test2	
0.1961	0.1341	beam:5 **0.2024**	**0.1626**	beam:1 0.2110	0.1577	
		0.2100	**0.158**	0.2024	0.1627	Fine-tuning
		beam:3, lr: 1e -5 no_repeat ngram size :3 **0.2213**	**0.1693 (lr:1e-5)**			Data Augument-ation
		0.2318	**0.1820 (lr:3e-5)**	Beam:3, LR: 3e-5, no_repeat_ngram_size :3		

Table 1. all model BLEU score

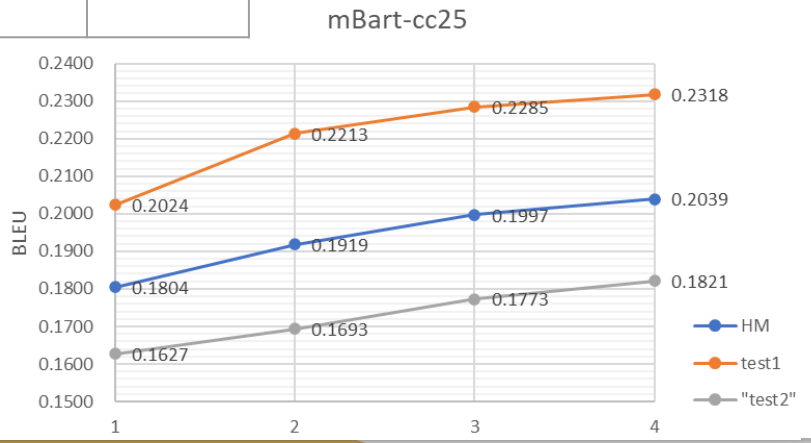
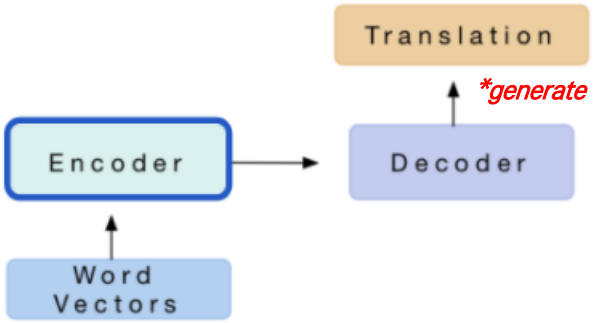


Fig1. mBart-cc25 BLEU score

5. Model 성능 개선

A) 모델 관점



*generate

no_repeat_ngram_size (`int`, *optional*, defaults to 0) — If set to int > 0, all ngrams of that size can only occur once.

encoder_no_repeat_ngram_size (`int`, *optional*, defaults to 0) — If set to int > 0, all ngrams of that size that occur in the `encoder_input_ids` cannot occur in the `decoder_input_ids`.

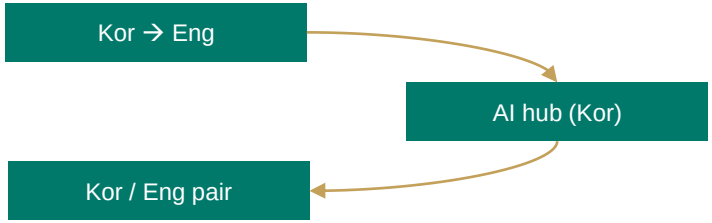
bad_words_ids(`List[List[int]]`, *optional*) — List of token ids that are not allowed to be generated. In order to get the token ids of the words that should not appear in the generated text, use `tokenizer(bad_words, add_prefix_space=True, add_special_tokens=False).input_ids`.

num_return_sequences(`int`, *optional*, defaults to 1) — The number of independently computed returned sequences for each element in the batch.

reference : Models (huggingface.co)

B) 데이터 관점

1. Back translation 으로 Ko/En pair 준비



2. 준비된 Ko/En pair + 기존 train/dev data

5. Model 성능 개선 : 모델 관점

문제점 : 특정단어가 중복되어 나타나는 현상

test-436	나는 어렸을 때 수영장에 자주 갔었어.
test-437	나는 나잇속에 내 셔츠를 찢었어요.
test-438	양자 결합 결합 결합 결합 결합 결합 결합 결합 결합 결합 결합 결합 결합 결합 결합 결합 결합

test-437	저는 셔츠를 손톱에 찢었어요.
test-438	두 겹의 베이스를 잡은 글리코시디의 결합들은 한꺼번에 helix로부터 직진합니다.

해결 방안 : model.generate option 사용 및 변경

* no_repeat_ngram_size → 지정된 사이즈와 같은 ngram발생 제한
* num_return_sequences → 반환 시퀀스의 수 제한
* num_beams → beam search의 beam 개수 설정

5. Model 성능 개선 : 데이터 관점

AI Hub 개방 데이터 외부 데이터 활용 사례 개발 지원 경진

개방 데이터

비전

한국어-영어 번역(병렬) 말뭉치 소개

뉴스	뉴스 텍스트	80만 문장
정부 웹사이트/저널	정부/지자체 홈페이지,간행물	10만 문장
법률	행정 규칙,자치 법규	10만 문장
한국문화	한국 역사,문화 콘텐츠	10만 문장
구어체	자연스러운 구어체 문장	40만 문장
대화체	상황/시나리오 기반 대화 세트	10만 문장

Data augumentation by back translation :

Train 15만 + Dev 1만 +

뉴스: 4만,
대화체: 1만
구어체: 3.4만

뉴스: 0.67만
대화체: 0.67만
구어체: 0.27만

Back translation
: 15hours

Train 23.4만 Dev 2.6만

Result:

test1	test2
0.2024	0.1626
0.2100	0.158
0.2213	0.1693 (lr:1e-5)
0.2318	0.1820 (lr:3e-5)

6. 자체평가 및 보완

▶아쉬운점 : wandb 사용하지 못한 점

▶느낀점 : fine tuning의 정확한 의미와 encoder, decoder의 기능에 대해 알아갈 수 있었다.

Q/A