Multi-label classification(2)

2022.11.17. HAI 1팀

한양대학교

HANYANG UNIVERSITY

방언 분류를 위한 모델 학습

- https://colab.research.google.com/github/GirinMan/HAI-DialectTranslator/blob/main/multi_label_classification/train/train_classifier.ipynb
- 생 지난 주에 봤던 4개의 라벨을 분류할 수 있는 모델을 만드는 파이썬 노트북 입니다.
- 실제로 학습을 진행해 보면 validation accuracy가 매우 낮게 나오는 것을 볼 수 있습니다. 이정도 정확도는 차라리 랜덤하게 뽑는게 더 나을 지경이네요. 어떻게 하면 정확도를 올릴 수 있을까요?

```
start training

training epoch 0: 100%|██████████| 7588/7588 [05:49<00:00, 21.72it/s]

start predict

1720it [00:26, 63.83it/s]

Epoch 0,  Average training loss: 0.7127 , Validation accuracy : 0.1751
```

더 나은 성능의 모델 만들기

- 모델의 학습에 영향을 미치는 것은 어떤 것이 있을까요? 학습 과정에서 Validation accuracy가 최대한 높아지도록 해 봅시다.

HINT 1: 모델에 입력될 최대 길이(**maxlen**), 학습을 진행할 **epoch** 횟수, **DataLoader**가 데이터를 샘플링하는 방식(**Sampler**), 한 번에 학습을 진행하는데 사용될 배치 사이즈(**batch**) 또는 데이터 자체의 전처리 방식 등 다양한 시도를 하며 모델을 학습시켜 봐요!

HINT 2: 우리가 이전에 BERT에 대해 배웠던 것을 기억해 보면, 어떤 **PLM**(Pre-trained language model)을 사용하느냐 에 따라서 성능이 크게 달라질 수 있어요. 한국어 발화 데이터인 만큼, Huggingface hub에서 **한국어**로 **pre-train**된 BERT 계열 모델을 사용하는 것이 아마 더 좋은 성능을 낼 거에요.

우수 참가자 시상식

main ▾ HAI-DialectTranslator / multi_label_classification / train / ···

GirinMan Merge pull request #2 from starhan98/main ··· 1 minute ago History

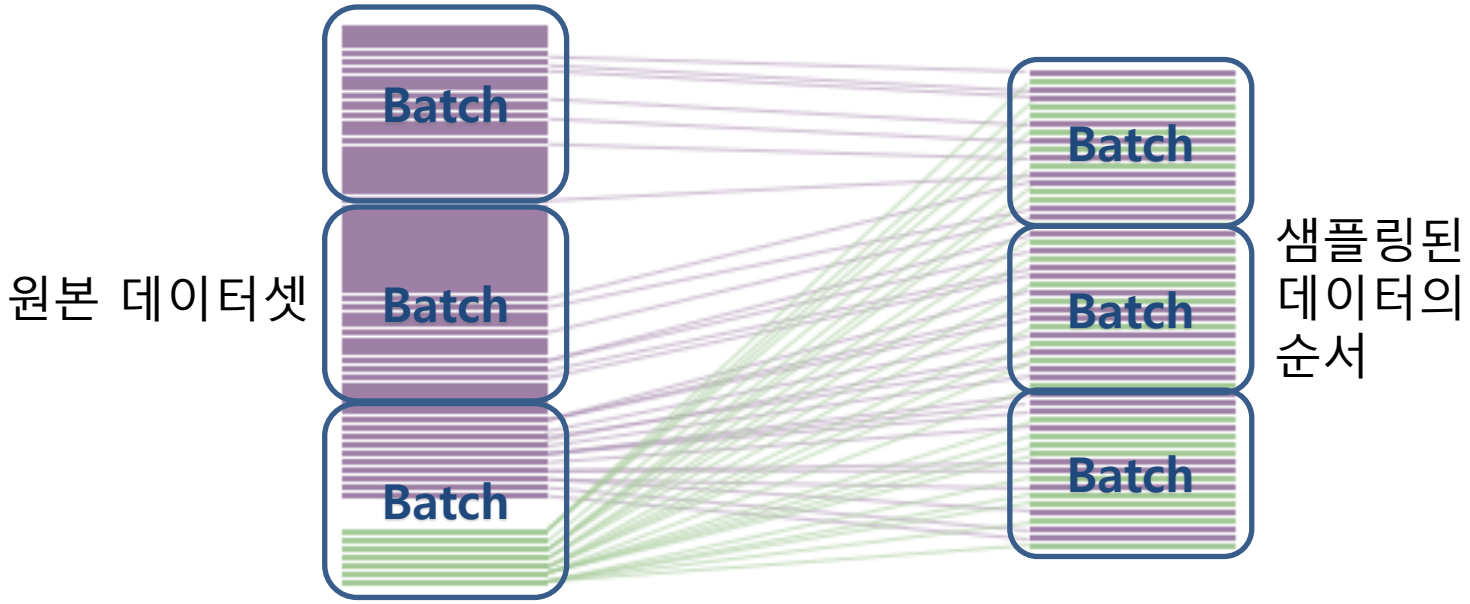
..

load_infer_model.ipynb	7 days ago
model.pth	7 days ago
model_한상엽_78.pth	9 hours ago
model_황태경.pth	3 days ago
train_classifier.ipynb	7 days ago
train_classifier_한상엽.ipynb	
train_classifier_황태경.ipynb	

```
start training
training epoch 0: 100%|          | 7588/7588 [53:45<00:00,  2.35it/s]
start predict
1720it [03:52,  7.40it/s]
Epoch 0,  Average training loss: 0.3743 , Validation accuracy : 0.7818
training epoch 1: 100%|          | 7588/7588 [53:44<00:00,  2.35it/s]
start predict
1720it [03:52,  7.40it/s]
Epoch 1,  Average training loss: 0.3064 , Validation accuracy : 0.7841
training epoch 2:  64%|          |  4849/7588 [34:17<19:21,  2.36it/s]
```

DataLoader의 샘플링 방식 변경

```
def get_data_loader(inputs, masks, labels, batch_size=args.batch):
    data = TensorDataset(torch.tensor(inputs), torch.tensor(masks), torch.tensor(labels))
    sampler = RandomSampler(data)
    # sampler = SequentialSampler(data)
    data_loader = DataLoader(data, sampler=sampler, batch_size=batch_size)
    return data_loader
```

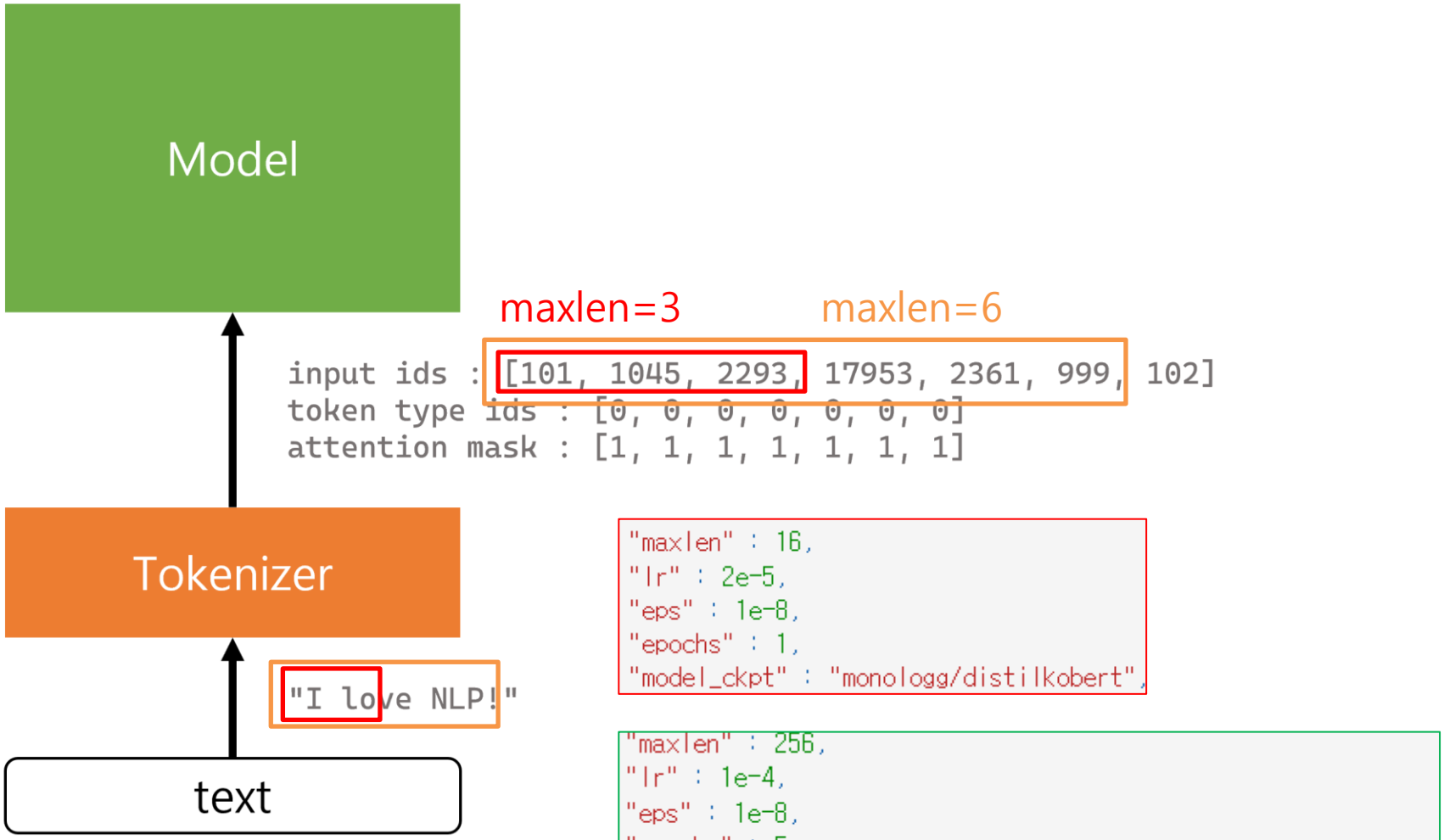


하이퍼파라미터 변경

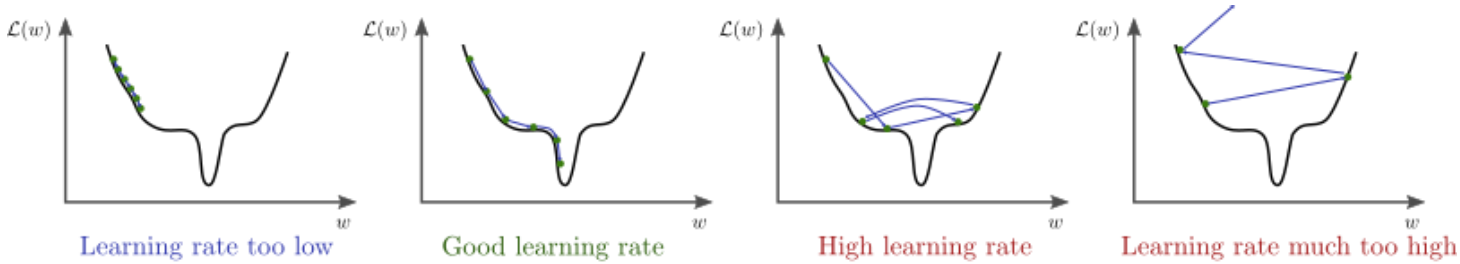
```
args = args = easydict.EasyDict({
  "train_path" : "./data/train_data.csv",
  "valid_path" : "./data/valid_data.csv",
  "device" : 'cpu',
  "mode" : "train",
  "batch" : 256,
  "maxlen" : 16,
  "lr" : 2e-5,
  "eps" : 1e-8,
  "epochs" : 1,
  "model_ckpt" : "monologg/distilkobert",
})

if torch.cuda.is_available():
    args.device = 'cuda'
```

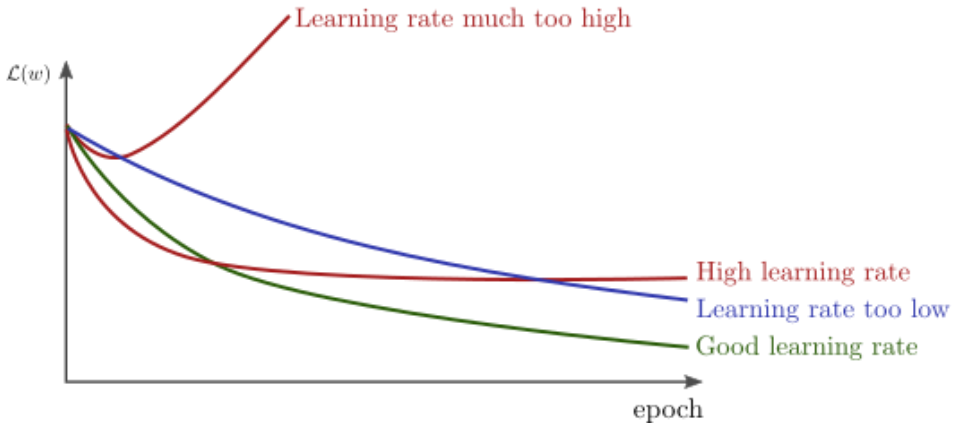
```
args = args = easydict.EasyDict({
  "train_path" : "./data/train_data.csv",
  "valid_path" : "./data/valid_data.csv",
  "device" : 'cpu',
  "mode" : "train",
  "batch" : 256,
  "maxlen" : 256,
  "lr" : 1e-4,
  "eps" : 1e-8,
  "epochs" : 5,
  "model_ckpt" : "monologg/koelectra-small-v3-discriminator",
})

if torch.cuda.is_available():
    args.device = 'cuda'
```

Tokenizer의 작동방식과 maxlen



Learning rate와 epochs

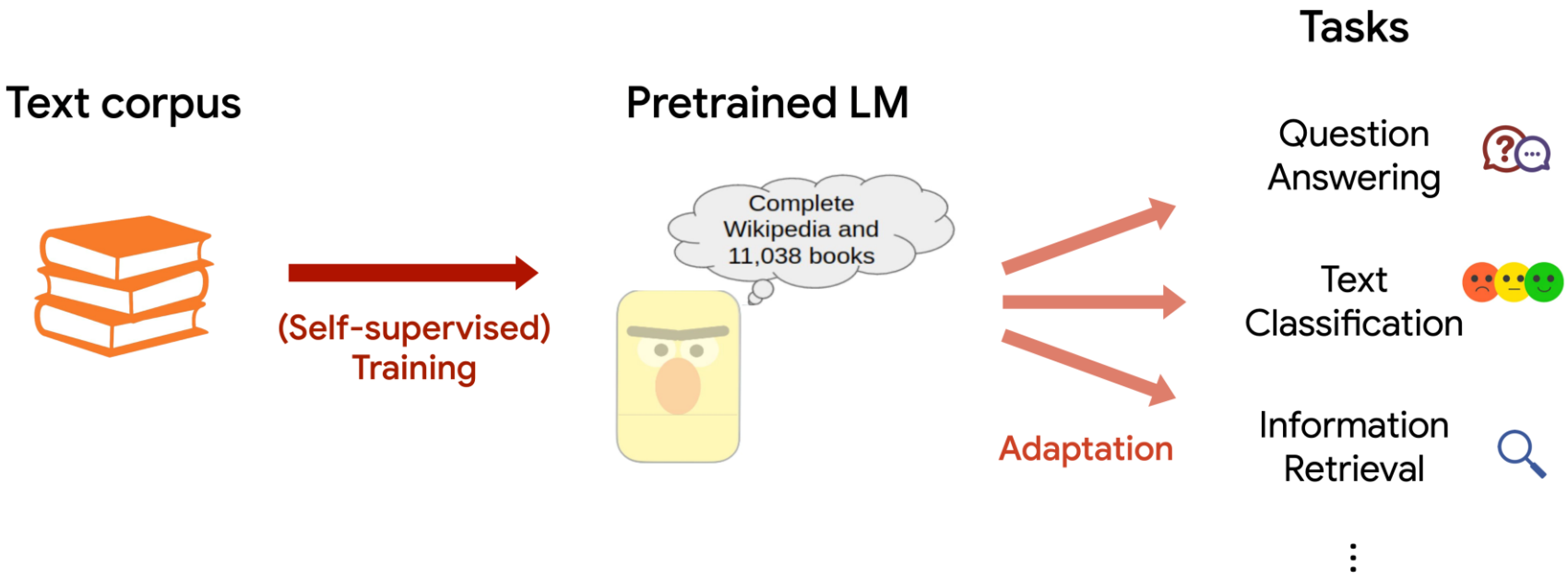




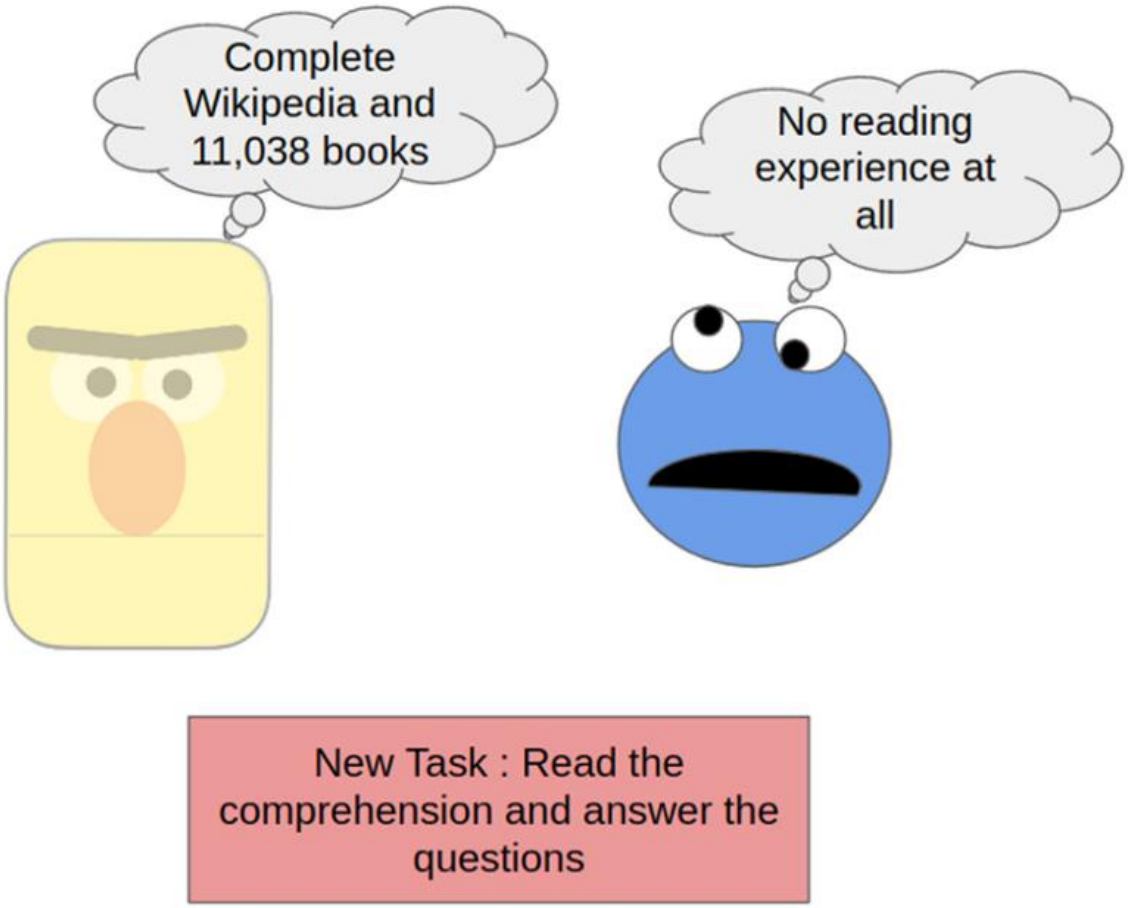

```
"maxlen" : 16,
"lr" : 2e-5,
"eps" : 1e-8,
"epochs" : 1,
"model_ckpt" : "monologg/distilkobert",
```

```
"maxlen" : 256,
"lr" : 1e-4,
"eps" : 1e-8,
"epochs" : 5,
"model_ckpt" : "monologg/koelectra-small-v3-discriminator",
```

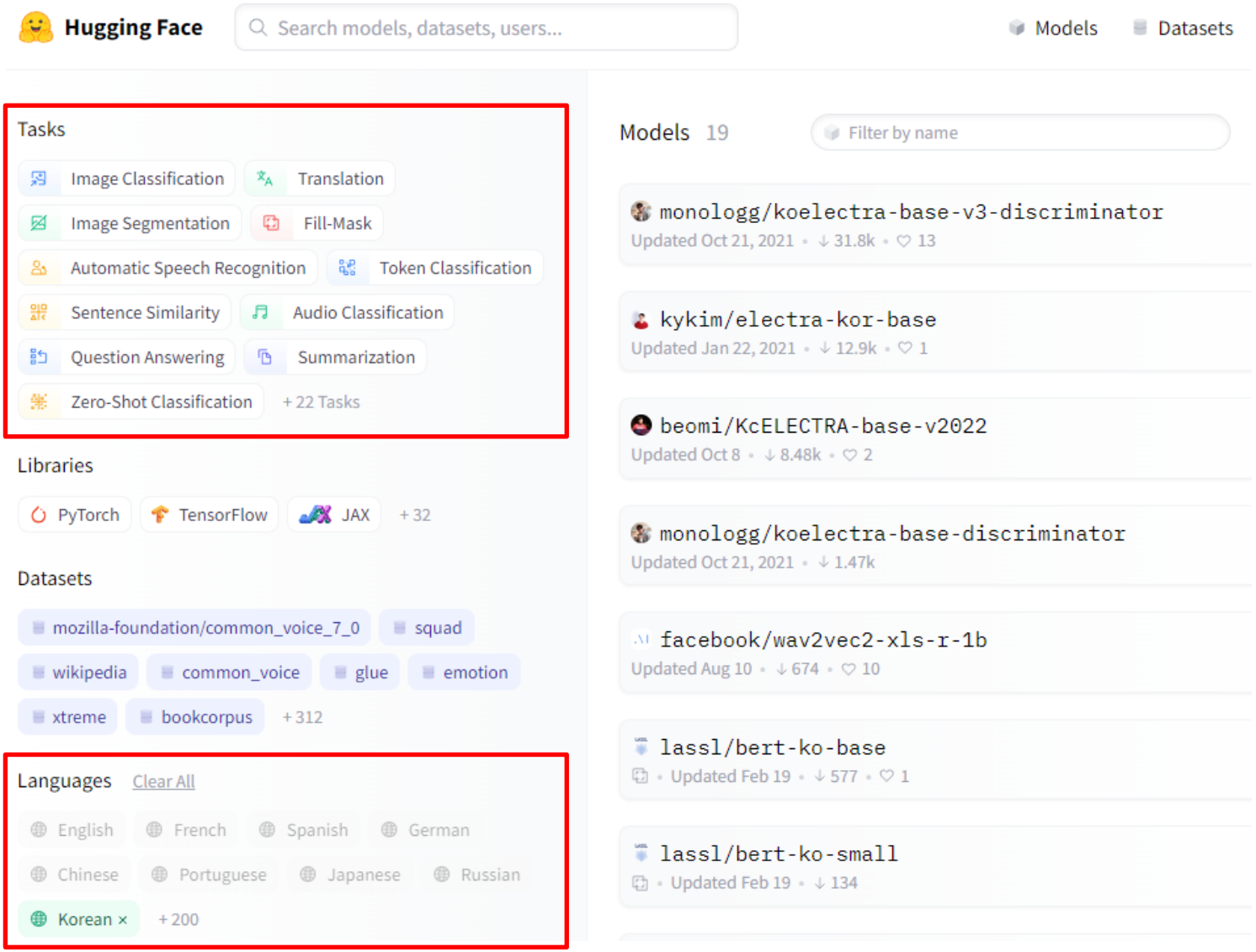
Pretrained Language Model



Pretrained Language Model



한국어 데이터로 학습된 PLM



모델 성능 벤치마크 비교

Small Model

	Size	NSMC (acc)	Naver NER (F1)	PAWS (acc)	KorNLI (acc)	KorSTS (spearman)	Question Pair (acc)	KorQuaD (Dev) (EM/F1)
DistilKoBERT	108M	88.41	**84.13**	62.55	70.55	73.21	92.48	54.12 / 77.80
KoELECTRA-Small	53M	**88.76**	84.11	**74.15**	**76.27**	**77.00**	**93.01**	**58.13 / 86.82**

```
"maxlen" : 16,
"lr" : 2e-5,
"eps" : 1e-8,
"epochs" : 1,
"model_ckpt" : "monologg/distilkobert",
```

```
"maxlen" : 256,
"lr" : 1e-4,
"eps" : 1e-8,
"epochs" : 5,
"model_ckpt" : "monologg/koelectra-small-v3-discriminator",
```

추가 정확도 향상 가능성...?

- PLM과 하이퍼파라미터를 조금 바꾼 것 만으로도 학습 과정에서 엄청나게 큰 변화가 있다는 것을 알 수 있었어요.

- 하지만 아직까지 모델이 4가지 종류의 지역 방언을 완벽하게 분류할 수 있지는 않은 것 같아요. 어떻게 하면 더 정확한 모델을 만들 수 있을까요?
- 모델이 학습하는 데이터셋에 중복되거나 정상적이지 않은 데이터는 없는지, 각 라벨들 간의 데이터 수가 균일하게 분포되어 있는지, 특별히 전처리를 할 필요는 없는 지 등을 살펴보면 어떨까요?

- 충분히 학습이 잘 된 PLM은 fine-tuning 과정에서 목표한 downstream task에 적합하게 변화하지만 한국어 문맥을 잘 이해하는 능력이 다소 희석될 수 있어요. Classification을 위한 ouput layer를 제외한 나머지 레이어들 중 일부의 가중치가 업데이트되지 않게 하면 어떨까요?

- Huggingface의 classification 모델에 기본적으로 사용되는 Cross-entropy 함수 외에 다른 loss function을 사용해 보는 것은 어떨까요?

To be continued…

- 다음 시간에는 누군가 Huggingface에 업로드해둔 방언 번역 모델을 우리가 학습시킨 분류 모델과 조합하여 번역기 프로그램을 구성해 볼 거에요!

- 출발 언어와 목표 언어가 매칭되는 조합은 상당히 여러 가지가 있을 텐데, 어떻게 하면 번역 정확도도 높이면서 서비스를 원활하게 제공할 수 있도록 할 수 있을까요?

- 완성된 시스템은 **streamlit**이라는 데이터 사이언스 애플리케이션 프로토타입 개발 API를 이용하여 보기 쉽고 사용하기 쉬운 GUI 형태의 웹 애플리케이션으로 완성할 거에요.

- 실제 AI 모델을 활용한 서비스를 제공하기 위해서는 어떤 요소들을 고려해야 할까요?