Bioinformatics

Lab2

(실습 이해했는지에 대한 문제 하나 나옴
코드 쓰라는건 아님)

Index

개체 명 인식

NER

Language model 정보 추출의 한 프로세스

- 개체 명 인식 (named entity recognition): 미리 정의해 둔 사람, 회사, 장소, 시간, 단위 등에 해당하는 단어(개체 명)을 문서에서 인식하여 추출 분류하는 기법
- 일반적으로 IOB tag를 사용; B-Begin, I-Inside, O-Others

text에서 원하는 entity type을 찾아내는 방법

gene

Species

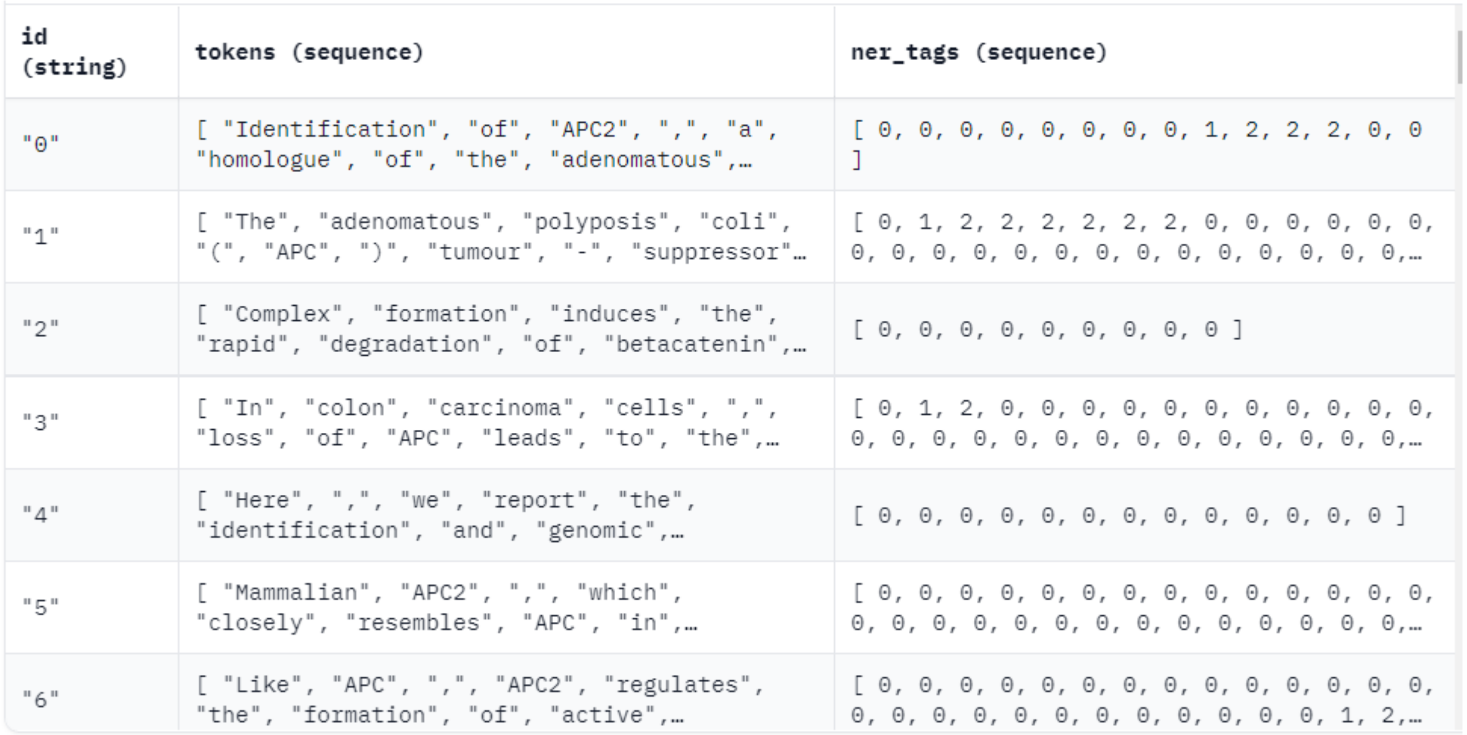
Tetracycline	-	resistance	genes	of	clostridium	perfringens	isolated	from	cattle
B-gene	I-gene	I-gene	I-gene	O	B-species	I-species	O	O	B-species

affected	with	malignant	edema	.
O	O	B-disease	I-disease	O

disease

데이터 세트

- https://huggingface.co/datasets/ncbi_disease



id (string)	tokens (sequence)	ner_tags (sequence)
"0"	["Identification", "of", "APC2", ",", "a", "homologue", "of", "the", "adenomatous",…	[0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 0, 0]
"1"	["The", "adenomatous", "polyposis", "coli", "(", "APC", ")", "tumour", "-", "suppressor"…	[0, 1, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
"2"	["Complex", "formation", "induces", "the", "rapid", "degradation", "of", "betacatenin",…	[0, 0, 0, 0, 0, 0, 0, 0, 0]
"3"	["In", "colon", "carcinoma", "cells", ",", "loss", "of", "APC", "leads", "to", "the",…	[0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
"4"	["Here", ",", "we", "report", "the", "identification", "and", "genomic",…	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
"5"	["Mammalian", "APC2", ",", "which", "closely", "resembles", "APC", "in",…	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
"6"	["Like", "APC", ",", "APC2", "regulates", "the", "formation", "of", "active",…	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2,…

ner_tags (IOB tag)

- 0 – O
- 1 – B-disease
- 2 – I-disease

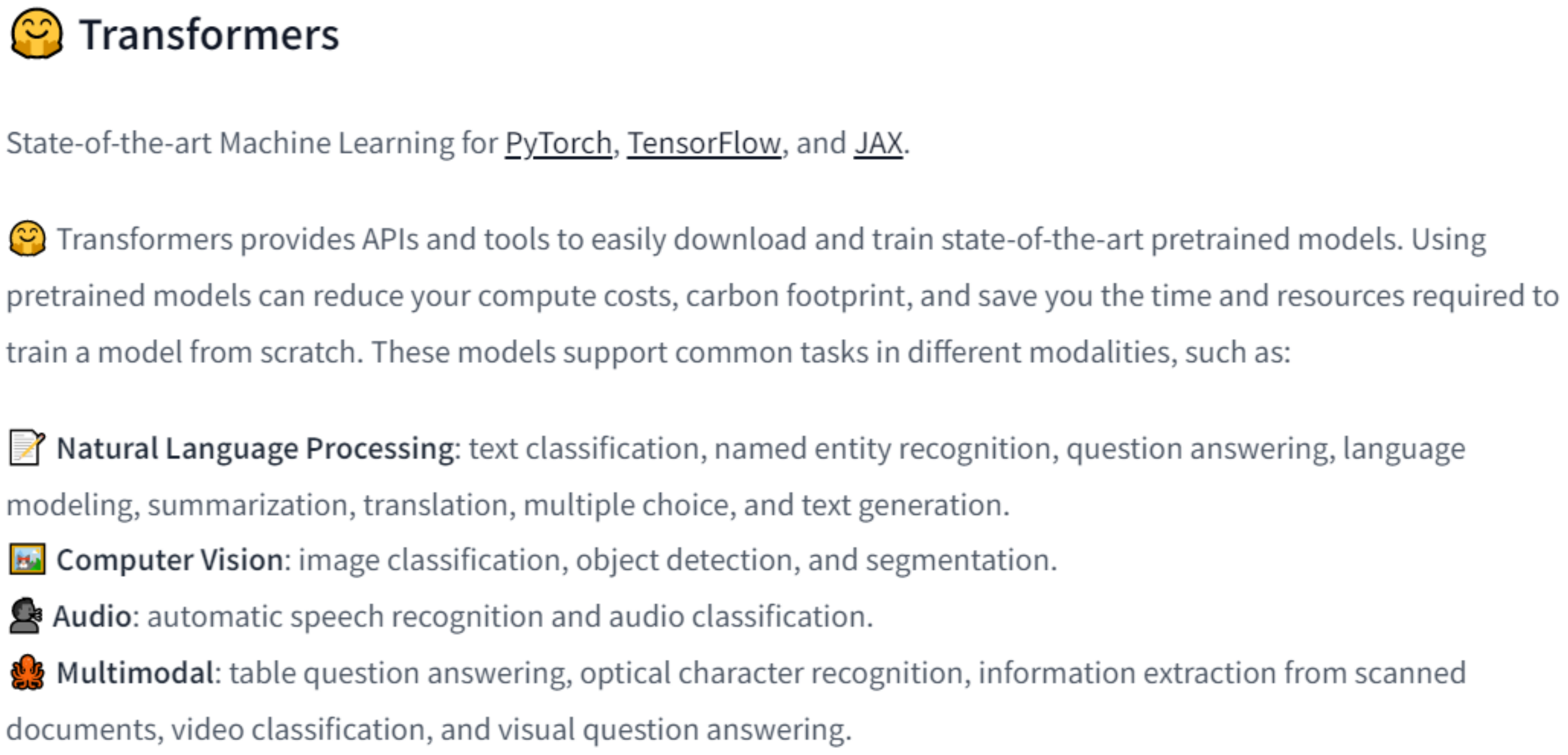
Environment

- Colab
  - Python 3.7.13
  - CUDA 11.2
  - Numpy 1.21.6
  - Scikit-learn 1.0.2
- Pytorch 1.12.0
- Transformers 4.24.0
- Datasets 2.7.1
- Evaluate 0.3.0 성능평가 도와줌
- seqeval

정답을 알아서 정의해주는 라이브러리

Transformers

- https://huggingface.co/docs/transformers/index



🤗 Transformers

State-of-the-art Machine Learning for PyTorch, TensorFlow, and JAX.

🤗 Transformers provides APIs and tools to easily download and train state-of-the-art pretrained models. Using pretrained models can reduce your compute costs, carbon footprint, and save you the time and resources required to train a model from scratch. These models support common tasks in different modalities, such as:

📝 **Natural Language Processing**: text classification, named entity recognition, question answering, language modeling, summarization, translation, multiple choice, and text generation.
🖼️ **Computer Vision**: image classification, object detection, and segmentation.
🗣️ **Audio**: automatic speech recognition and audio classification.
🐙 **Multimodal**: table question answering, optical character recognition, information extraction from scanned documents, video classification, and visual question answering.

How to train NER using transformers

Code: https://github.com/yesol-park/Bioinformatics2022

- Get dataset from dataset repository on the Hub

```
train_dataset = load_dataset("ncbi_disease", split="train")
test_dataset = load_dataset("ncbi_disease", split="test")

print(train_dataset)
print(test_dataset)
```

```
Dataset({
    features: ['id', 'tokens', 'ner_tags'],
    num_rows: 5433
})
Dataset({
    features: ['id', 'tokens', 'ner_tags'],
    num_rows: 941
})
```

How to train NER using transformers

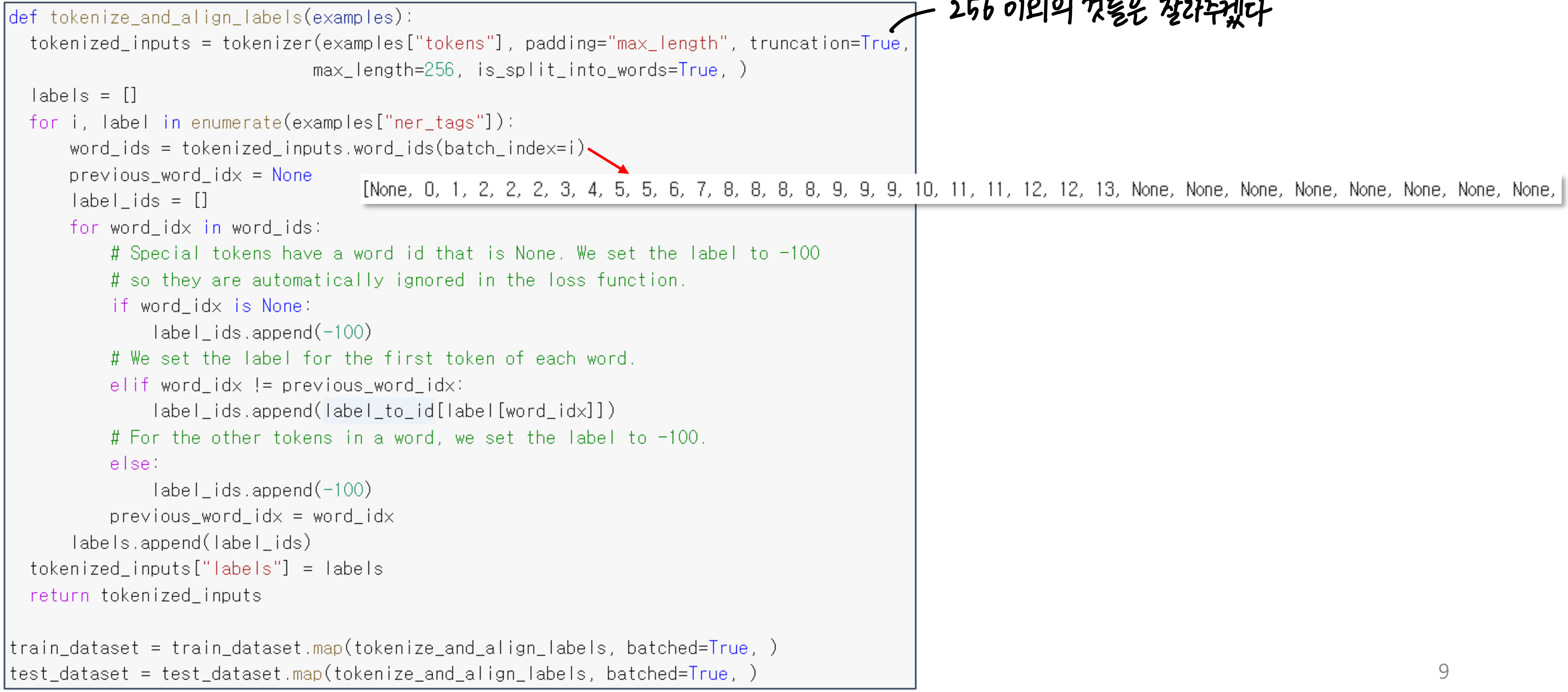
- Load tokenizer and pretrained model

```
model_name = "bert-base-uncased"

tokenizer = AutoTokenizer.from_pretrained(model_name)

config = AutoConfig.from_pretrained(
        model_name,
        num_labels=num_labels,
        finetuinnig_task="ner")
model = AutoModelForTokenClassification.from_pretrained(
    model_name,
    config=config,
)
```

Bert는 token 단위로 나눔

How to train NER using transformers

- Preprocessing the dataset



```python
def tokenize_and_align_labels(examples):
  tokenized_inputs = tokenizer(examples["tokens"], padding="max_length", truncation=True,
                               max_length=256, is_split_into_words=True, )
  labels = []
  for i, label in enumerate(examples["ner_tags"]):
      word_ids = tokenized_inputs.word_ids(batch_index=i)
      previous_word_idx = None
      label_ids = []
      for word_idx in word_ids:
          # Special tokens have a word id that is None. We set the label to -100
          # so they are automatically ignored in the loss function.
          if word_idx is None:
              label_ids.append(-100)
          # We set the label for the first token of each word.
          elif word_idx != previous_word_idx:
              label_ids.append(label_to_id[label[word_idx]])
          # For the other tokens in a word, we set the label to -100.
          else:
              label_ids.append(-100)
          previous_word_idx = word_idx
      labels.append(label_ids)
  tokenized_inputs["labels"] = labels
  return tokenized_inputs

train_dataset = train_dataset.map(tokenize_and_align_labels, batched=True, )
test_dataset = test_dataset.map(tokenize_and_align_labels, batched=True, )
```

```
[None, 0, 1, 2, 2, 2, 3, 4, 5, 5, 6, 7, 8, 8, 8, 8, 9, 9, 9, 10, 11, 11, 12, 12, 13, None, None, None, None, None, None, None, None,
```

256 이외의 것들은 잘라주겠다

How to train NER using transformers

- Preprocessing the dataset – set tag ids

tokenized 하고
label 변경 해주는 작업

Raw text	the	adenomatous	polyposis	coli	tumour
label ids	0	1	2	2	2

Tokenized	the	aden	##oma	##tou	##s	poly	##po	##sis	coli	tu	##mour
label ids	0	1	-100	-100	-100	2	-100	-100	2	2	-100

go having

going → ?

Tag ID	Tag
1	B
2	I
-100	<sub_token>

How to train NER using transformers

- Set metric function

성능평가

transformer에 같이 넣어줄

```python
def compute_metrics(p):
    metric = evaluate.load("seqeval")

    predictions, labels = p
    predictions = np.argmax(predictions, axis=2)

    # Remove ignored index (special tokens)
    true_predictions = [
        [label_list[p] for (p, l) in zip(prediction, label) if l != -100]
        for prediction, label in zip(predictions, labels)
    ]
    true_labels = [
        [label_list[l] for (p, l) in zip(prediction, label) if l != -100]
        for prediction, label in zip(predictions, labels)
    ]

    results = metric.compute(predictions=true_predictions, references=true_labels)
    return {
        "precision": results["overall_precision"],
        "recall": results["overall_recall"],
        "f1": results["overall_f1"],
        "accuracy": results["overall_accuracy"],
    }
```

성능평가에 유효한 애들만 모음

How to train NER using transformers

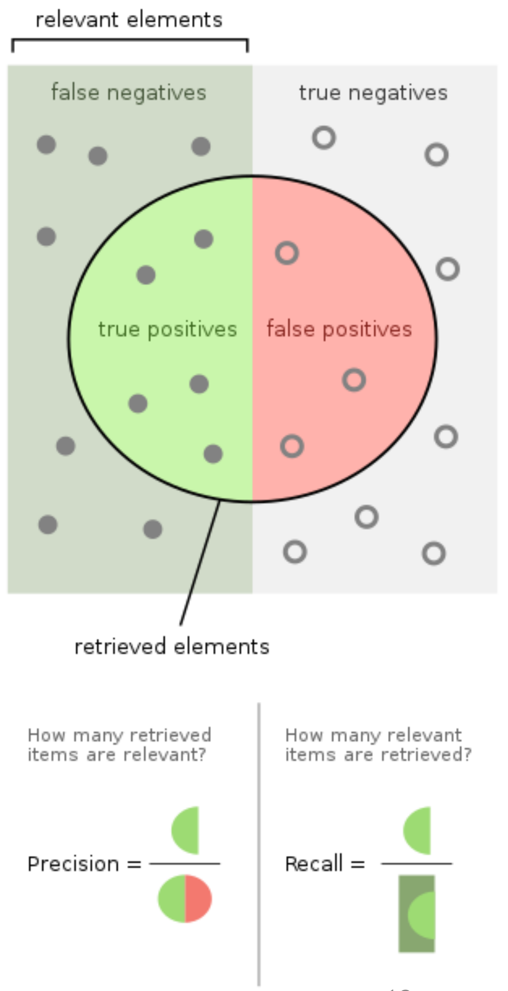
- Set metric function

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$



How to train NER using transformers

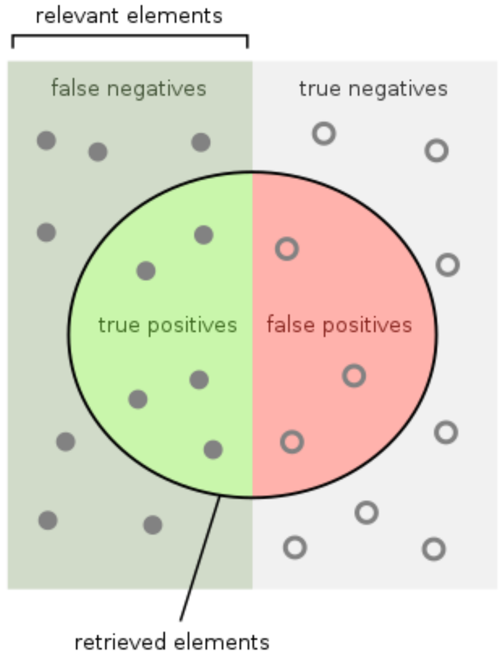
- Set metric function

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$



positive

Identification	of	ACP2	,	a	homologue	of	the	adenomatous	polyposis	coli	tumour	suppressor	.
O	O	O	O	O	O	O	O	B	I	I	I	O	O

How to train NER using transformers

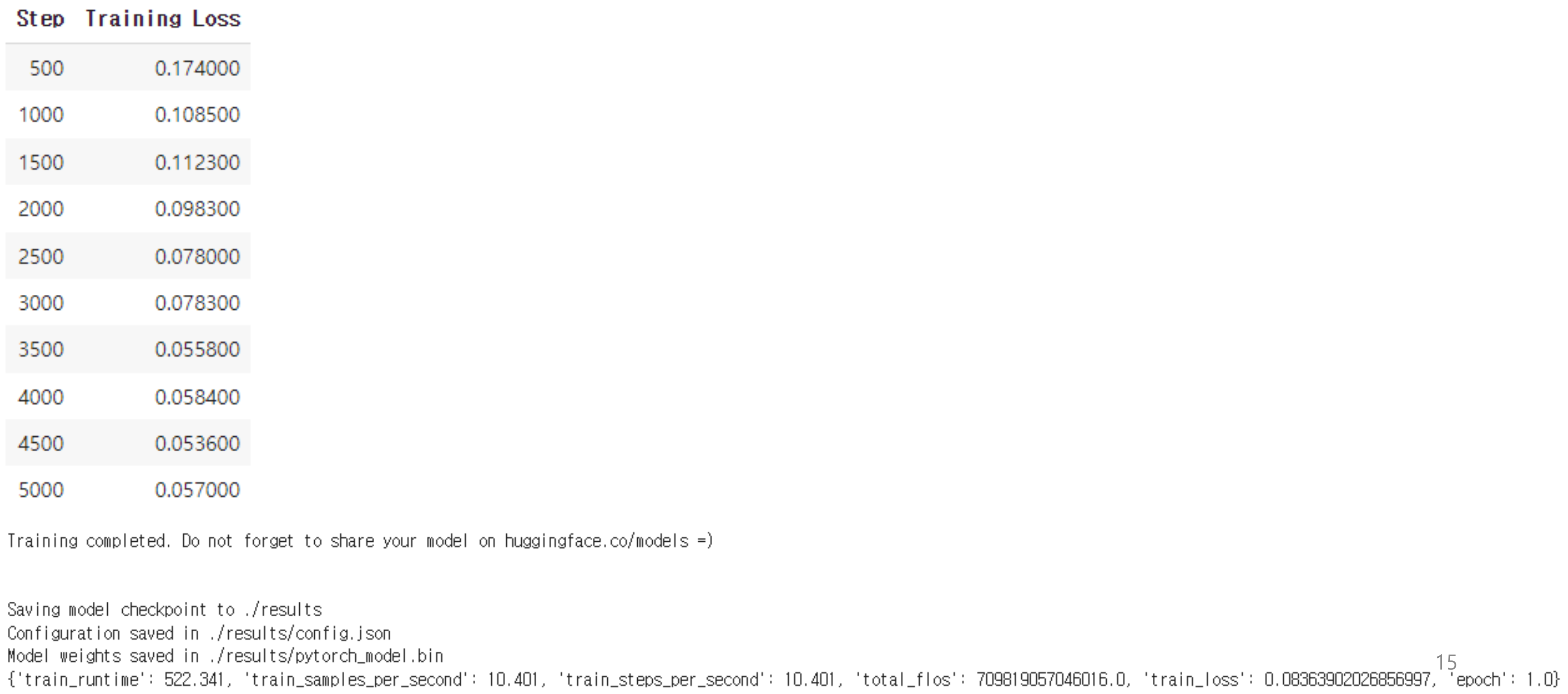
- Train NER model

Dataset 학습시 배치 사이즈로 묶어 학습함

```
data_collator = DataCollatorForTokenClassification(tokenizer, )
trainer = Trainer(
    model=model,  Bert
    args=training_args,
    train_dataset=train_dataset,
    data_collator=data_collator,
    compute_metrics=compute_metrics
)

train_result = trainer.train()
metrics = train_result.metrics
print(metrics)
```

How to train NER using transformers

- Train NER model



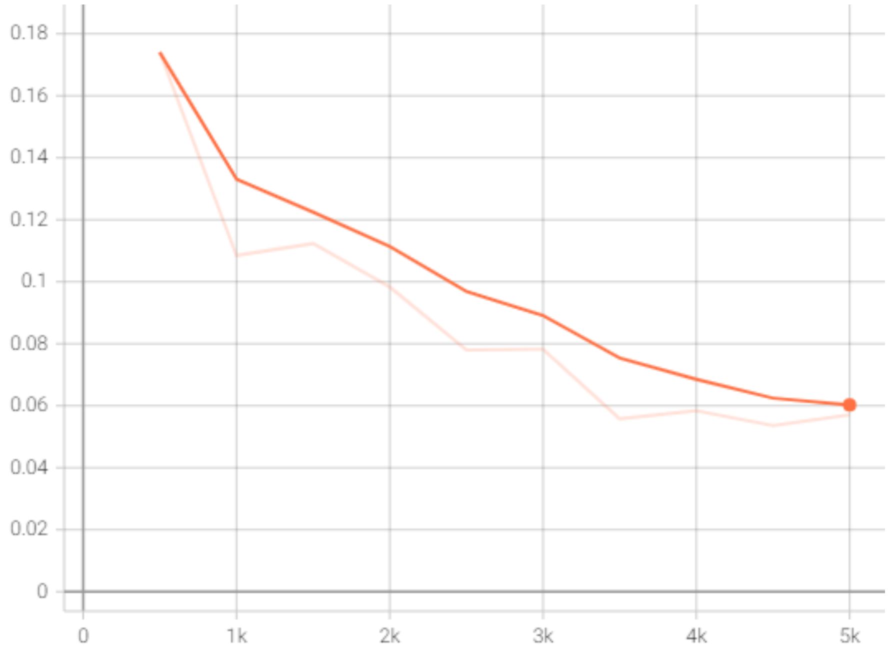
Step	Training Loss
500	0.174000
1000	0.108500
1500	0.112300
2000	0.098300
2500	0.078000
3000	0.078300
3500	0.055800
4000	0.058400
4500	0.053600
5000	0.057000

```
Training completed. Do not forget to share your model on huggingface.co/models =)


Saving model checkpoint to ./results
Configuration saved in ./results/config.json
Model weights saved in ./results/pytorch_model.bin
{'train_runtime': 522.341, 'train_samples_per_second': 10.401, 'train_steps_per_second': 10.401, 'total_flos': 709819057046016.0, 'train_loss': 0.08363902026856997, 'epoch': 1.0}
```

How to train NER using transformers

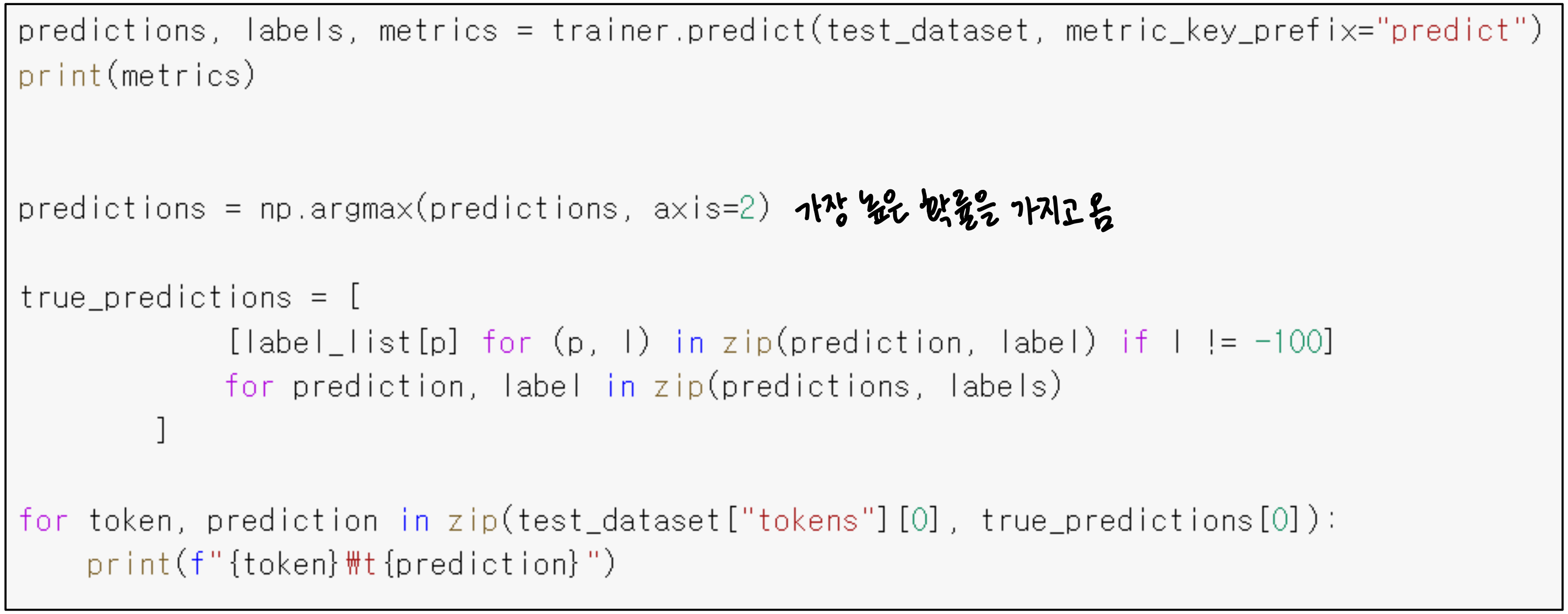
```
%load_ext tensorboard
%tensorboard --logdir="./results"
```

train/loss



How to train NER using transformers

- Test NER model

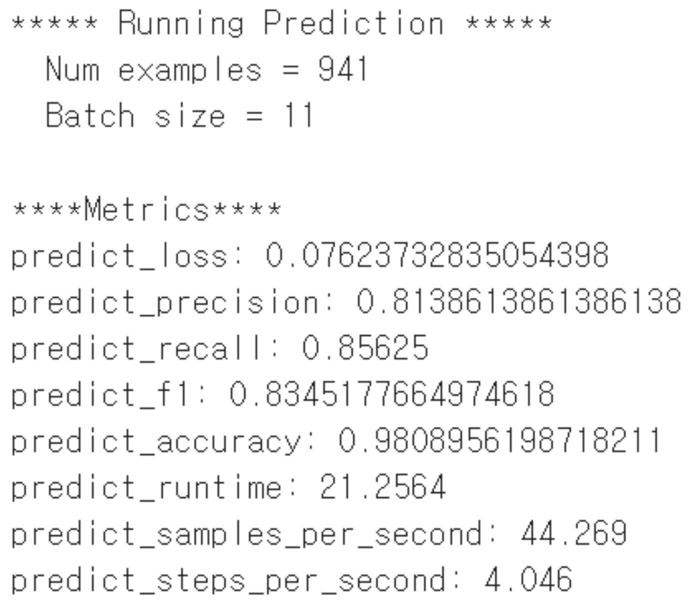


```
predictions, labels, metrics = trainer.predict(test_dataset, metric_key_prefix="predict")
print(metrics)


predictions = np.argmax(predictions, axis=2)

true_predictions = [
            [label_list[p] for (p, l) in zip(prediction, label) if l != -100]
            for prediction, label in zip(predictions, labels)
        ]

for token, prediction in zip(test_dataset["tokens"][0], true_predictions[0]):
    print(f"{token}₩t{prediction}")
```

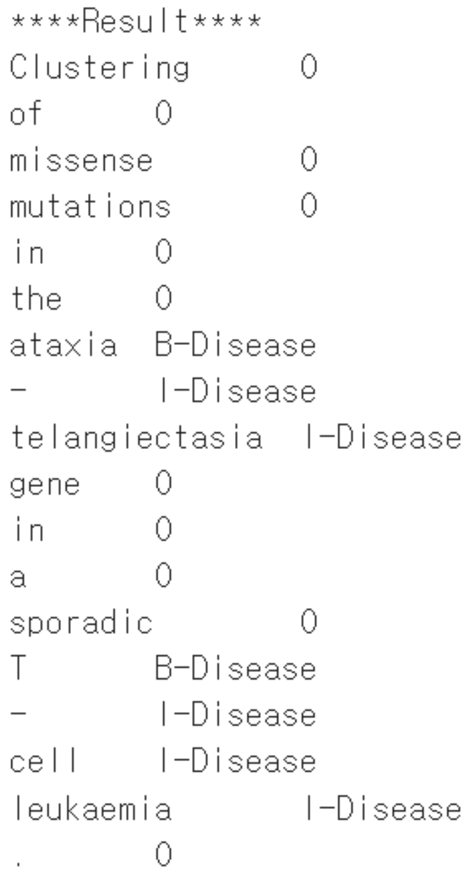
가장 높은 확률을 가지고옴

How to train NER using transformers

- Test NER model



```
***** Running Prediction *****
  Num examples = 941
  Batch size = 11

****Metrics****
predict_loss: 0.07623732835054398
predict_precision: 0.8138613861386138
predict_recall: 0.85625
predict_f1: 0.8345177664974618
predict_accuracy: 0.9808956198718211
predict_runtime: 21.2564
predict_samples_per_second: 44.269
predict_steps_per_second: 4.046
```



```
****Result****
Clustering      O
of      O
missense        O
mutations       O
in      O
the     O
ataxia  B-Disease
-       I-Disease
telangiectasia  I-Disease
gene    O
in      O
a       O
sporadic        O
T       B-Disease
-       I-Disease
cell    I-Disease
leukaemia       I-Disease
.       O
```

Ataxia-telangiectasia is a rare inherited disorder that affects the nervous system, immune system, and other body systems.

T-cell leukemia is an uncommon type of blood cell cancer that affects your white blood cells.