PEP 0008 소감문

23기 오재현

'가독성이 좋은 코드'에 대해서 고민할 일이 잘 없었습니다. 코드를 작성해야 할 일이라고는, '어떤 기능을 구현하세요' 라는 instruction에 맞는 코드를 작성하여 제출해야 하는 일들 뿐이었고, 누군가에게 코드를 보여지는 모양새에 신경을 쓸 일은 없었습니다.

물론 코드를 제출할 때 마다, 주석을 다듬고 줄넘김을 치며 코드가 예뻐보이기를 바란적은 많지만, 그것은 코드 제출 3분전의 스쳐가는 바람이었을 뿐 언젠가 코드가 가독성이 좋아보이게 쓰는 법을 배우자는 생각으로는 이어지지 않았습니다.

그러다 파이토치 딥러닝에 친숙해지고자 들어온 캠프에서 PEP 0008 문서를 보게 되었고, 그동안 봐왔던 github의 모범적 코드들의 포맷이 이해되었습니다. 왜 함수와 클래스 사이의 줄넘김은 두개로 통일되어 있었는지, 복수의 아규먼트들에 대한 줄넘김이 왜 그런 형태로 그려져 있었는지에 대한 이해가 순식간에 되었습니다.

코드를 가지고 협업할 일은 많을 것 같은데, 그 때마다 오늘 읽은 PEP-8 자료가 기억 날 것 같습니다. 비단 오늘 하루의 경험 뿐 아니라, 그동안 가독성 좋게 쓰여진 코드들이 사실은 어떠한 암묵적 규칙을 지키고 있었다는 것을 알게 되었기 때문입니다.

Code Report

1. 클래스 별 구현 고려 사항, 구조 및 작동 원리

BPETokenizer 클래스

(1)구현 고려 사항

corpus를 입력으로 받아 초기화하고, 추가적인 corpus는 add_corpus 메소드를 통해 추가 가능합니다.
학습은 train 메소드를 호출하여 주어진 반복 횟수만큼 수행됩니다.
텍스트 토큰화는 tokenize 메소드를 통해 수행되며, padding 및 max_length 옵션을 고려합니다.
상속을 활용하여 공통 기능을 부모 클래스로 추상화하였습니다.

(2)메소드 구조 및 작동 원리

_preprocess_corpus: 주어진 corpus를 전처리하여 리스트 형태로 반환합니다.
_preprocess_text: 주어진 텍스트를 전처리하여 반환합니다.

WordTokenizer 클래스

(1)구현 고려 사항

BPETokenizer와 유사한 인터페이스를 가지나, BPE 대신 간단한 공백을 기준으로 단어를 토큰화합니다.
train 메소드는 사용되지 않으며, 인자를 받지 않아도 됩니다.

(2) 메소드 구조 및 작동 원리

_tokenize_single: 주어진 텍스트를 단어 단위로 토큰화하여 반환합니다.
train: 더미 구현으로, 인자를 받지 않아도 동작합니다.
나머지 메소드는 BPETokenizer 클래스와 유사하게 작동하며, 공통 부분은 부모 클래스로 추상화하였습니다.

2. 주어진 조건 충족 방법

(1) 모듈화

BPETokenizer와 WordTokenizer를 tokenizers 모듈로 모듈화하여 구현하였습니다.

텍스트 전처리 관련 기능을 preprocessing 모듈로 모듈화하여 구현하였습니다.

(2) 상속 및 공통 기능 추상화

BPETokenizer와 WordTokenizer 클래스에서 공통된 기능을 추상화하기 위해 부모 클래스를 생성하고 상속을 활용하였습니다.

(3) 데코레이터 및 타입 힌트 사용

데코레이터는 현재 코드에서 사용되지 않았습니다. (...)
typing 모듈을 사용하여 매개변수와 반환값에 명시적으로 타입을 지정하였습니다.

(4) 텍스트 전처리 구현

Preprocessor 클래스를 통해 주어진 조건에 맞게 텍스트 전처리를 구현하였습니다.

추신)
코드에 주석을 추가하여 각 메소드와 클래스의 동작 방식을 설명하였습니다.
사용자가 입력한 인자를 받아오기 위해 argparse 모듈을 사용하였습니다.