과제1 요약 및 소감문

#코드별 주석 달아주기

#가능한 수직 정렬과 들여쓰기를 활용할 것

#총 줄의 개수가 79줄을 넘지 않게 할 것

- 민정

  여태껏 과제 위주형 코딩을 하다보니 혼자 이해할 수 있는 형태로만 코드를 정리하였는데, 해당 가이드를 읽으며 앞으로는 가독성 좋은 코드 정리 방식을 적용해보아야겠다는 깨달음이 있었습니다!

  다음 프로젝트나 컴퍼런스 때는 github 를 활용하는 데 그치지 않고, 독자가 충분히 이해하기 쉬운 형태로 전달해보고 싶습니다!

- 종원

  협업을 할때, Python code 를 어떻게 써야 깔끔하고, 모두가 알아볼 수 있게 코드를 작성할 수 있는지, 실제 방법론을 다룬 글이었습니다.

  이전까지 코드를 작성할때는 함수의 위치라던지, 인수를 어떻게 정리해야하는지 그런 부분을 전혀 고려하지 않았기에, 그동안의 코드 작성 습관에 대해서 되돌아볼 수 있는 계기가 되었습니다.

  PEP8 을 읽고 난 후 협업의 관점에서 코드를 작성한다는 것의 의미를 알게 되었으며, 해당 내용을 알고 실천하는 것이 중요함을 느꼈습니다.

과제2 보고서

- 각 class 를 디자인할 때 고려한 부분
  - typing 사용하여 type 을 명확히 지정

    - 수직 정렬 활용하여 가독성 높임
    - 적절한 분기처리 활용
- 각 method 의 구조와 작동 원리
    - TextPreprocessor 클래스는 텍스트 전처리를 위한 클래스임. 소문자 변환과 구두점 제거 같은 전처리 기능 수행함.
    - BaseTokenizer 클래스는 토큰화를 위한 기본 클래스임. 어휘 사전 관리하고 텍스트를 토큰으로 변환하는 메소드를 포함함.
    - WordTokenizer 클래스는 **BaseTokenizer**를 상속받음. 단어 기반 토큰화를 수행함. 특별한 훈련 과정 필요 없음.
    - BPETokenizer 클래스도 **BaseTokenizer**를 상속받음. BPE 알고리즘을 사용해 토큰화 수행함. 특정 단어 쌍의 빈도를 기반으로 최적의 토큰화 수행하도록 훈련됨.
    - **BPETokenizer**의 train 메소드는 BPE 알고리즘 훈련을 위해 사용됨. 가장 빈번한 단어 쌍을 찾아 병합하고 어휘를 업데이트함.
    - update_frequencies 메소드는 병합된 단어 쌍의 빈도를 업데이트함. 이는 BPE 토큰화의 효율성을 높이기 위해 필요함.
- 제시된 조건들을 충족시킨 방법
    - typing 을 활용하여 가독성을 높였습니다.
    - 가장 흔한 쌍이 먼저 병합되도록,이미 병합된 pair 는 다시 계산할 필요없도록 하여 traning 과정의 효율성을 높였습니다.
    - 전처리 클래스 TextPreprocessor 와 공통 기능을 하는 클래스 BaseTokenizer 를 구현하여 상속기능을 사용하였습니다.
- 협업 방식
    - 노션을 통해 참고자료와 진행 상황을 공유하며 토크나이저와 BPE 알고리즘을 함께 정확히 이해하고자 노력했습니다.
    - 서로 다른 토크나이즈를 구현하며 코드 진행 과정을 공유하고, 서로의 이해 방식을 교환하며 과제 수행의 능률을 높였습니다.