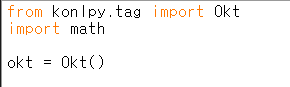
Assignment2 보고서

컴퓨터소프트웨어 학부

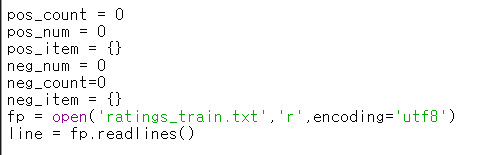
2015004439

김재홍

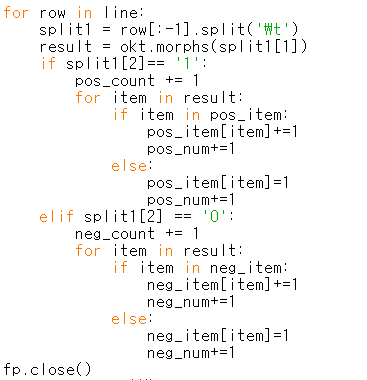
**코드설명**



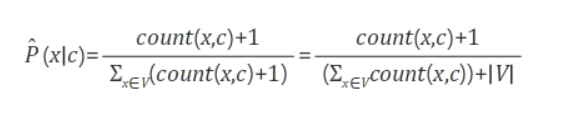
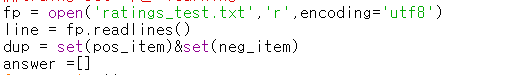
들어오는 문장의 형태소를 분석하기 위하야 Okt를 import하여 사용하였습니다.



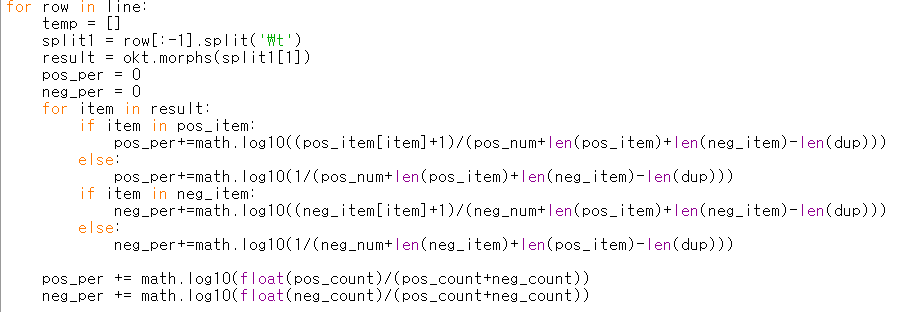
Training 파일에서 들어오는 긍정 문장을 세어주는 pos\_count, 그 형태소를 저장해주는 pos\_item, 총 긍정 형태소의 수를 나타내는 pos\_num, 부정 문장을 세어주는 neg\_count, 그 부정 형태소를 저장하는 neg\_item, 부정 형태소의 수를 세어주는 neg\_num 변수를 사용하였습니다. 그리고 ratings\_train의 파일을 읽어왔습니다.



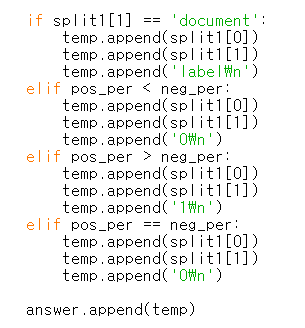
읽어들어오는 문장을 okt.morphs를 통해 형태소로 쪼개고, 그 읽어들인 문장의 결과가 긍정인지 부정인지에 따라 각각 다르게 처리해주었습니다. 위에 있는 for문의 결과 training set에서 형태소에 따라 그것이 부정인지 긍정인지를 계산할 수 있도록 학습시켰습니다.



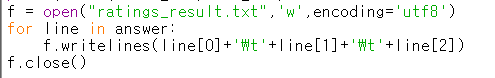
이후 학습한 데이터를 기반으로 ratings\_test 파일을 읽어 그 결과를 확인하는 실험을 진행하였습니다. 저는 위와 같은 라플라스 스무딩을 진행한 공식을 사용하였기에 dup라는 변수를 통해 pos\_item과 neg\_item에서 중복되는 형태소의 개수를 세었습니다.



들어오는 문장에 대해 형태소 별로 분류를 진행하였고 그것을 긍정에 포함되는지 부정에 포함되는지를 위 계산공식에 맞게 계산하였습니다. 그리고 underflow 방지를 위하여 log를 사용하였습니다.



그리고 각 결과에 맞게 1과 0을 선택하여 문장 뒤에 넣어주었습니다. 이때 answer라는 리스트를 활용하여 최종 결과값들을 넣었습니다. 저는 만약 긍정 확률과 부정 확률이 같게 나온다면 부정으로 처리하도록 프로그램을 구현하였습니다.

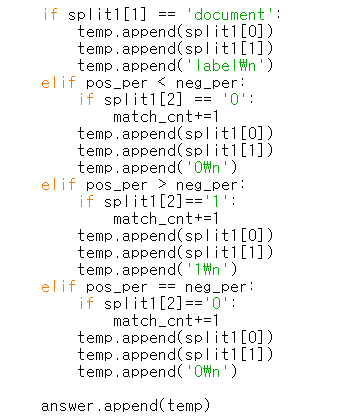


마지막으로 answer에 저장한 결과물을 ratings\_result 텍스트 파일에 입력하였습니다.

**실험결과**

과제에서 주어진 ratings\_valid 파일을 이용하여 제 코드의 정확도를 검색하였습니다.

기존 training을 한 코드는 그대로 사용하였고, learning을 진행하는 곳에서 코드를 수정하였습니다.



Ratings\_valid 파일에는 숫자의 결과가 남아 있으므로, 그 결과를 기준으로 제 프로그램의 결과와 일치하는 결과들의 수를 세서 정확도를 계산하였습니다.

그 결과,

 약 85.14%의 결과를 얻을 수 있었습니다.