**인공지능**

**프로젝트 2: 영화 리뷰 긍정/부정 분류**

**2013012278 하경모**

**0. compilation environment**

- Windows 10

- python 3.6.6

- konlpy 외부 패키지 사용

**1. 목적**

- 주어진 영화 리뷰 데이터와 naïve bayes model을 이용해 test case의 label 분류

- test case가 긍정으로 분류되면 1, 부정으로 분류되면 0으로 분류

**2. 디자인**

- 형태소 분석기(konlpy)로 리뷰(문장)를 특정 단어로 분류

- 긍정과 부정 라벨을 구분해서 저장

- 학습한 데이터와 naïve bayes model을 이용해 test case의 긍정/부정 확률 계산

- 확률이 더 높은 라벨로 분류

**3. version 1.x**

- version 1.0

- 구현 자체에 의의를 두고 제작

- 크게 3개의 함수로 구현

- train(), classify(), write\_result()

- train()

- training data 파일을 읽고 라벨에 따른 단어를 분류

- 파일 open 시 utf-8로 인코딩

- konlpy 모듈이 utf-8 방식을 지원하기 때문

- konlpy 모듈 중 okt.pos 사용

- 구문 분석에 필요한 시간이 적기에 채택

- 구문 분석 후 명사, 형용사, 동사를 training data로 선택



- training data를 긍정, 부정 리스트에 저장

- classify()

- test data 파일을 읽고 data의 라벨을 training data에 따라 분류

- NB\_classifier() 함수 사용

- NB\_classifier()

- list\_pos, list\_neg 안에 해당 단어가 있는지 검색해 각 문장의 긍정, 부정 확률 계산

- list\_pos, list\_neg는 각각 긍정, 부정 단어의 집합 리스트

- P(pos|word) = P(word|pos)\*P(pos)/P(word)

- 이 때, P(pos)는 각각의 댓글이 절반의 확률로

긍정, 부정일 것이라 가정해 계산에서 제외

- P(word)는 공통 분모이므로 비교에는 영향을 미치지 못하므로 제외

- 확률 계산시 문장에 training data에 없는 단어가 나올

확률이 있으므로 laplace smoothing 적용



- pos,neg\_cnt : training data 내의 목표 단어 개수

- num\_pos,neg : 긍정, 부정 단어의 개수

- total\_word : training data내의 단어 종류의 개수

- 확률 곱계산 시 floating point 문제와 underflow가 날 확률이 있으므로

Log함수로 계산

- 긍정 확률이 부정보다 높다면 label을 1로 분류, 그 외는 0

- write\_result()

- label 분류가 완료된 test data를 ratings\_result.txt 파일에 출력

- 실행 결과

- training에 6-7분, 실행 완료까지 무려 1시간 16분 소요

- 정확도를 떠나 너무 느린 프로그램이기에 개선 필요

- version 1.1

- version 1.0의 classify()에서 1시간이 넘게 소요

- classify() 중 리스트에 단어가 몇 개 있는지 O(n)으로 일일이 검색하기에 나온 결과

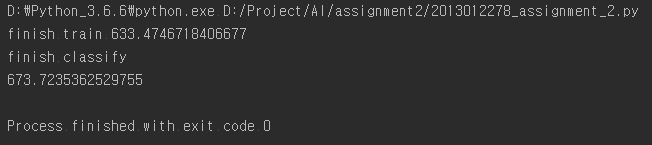
- hashing 필요

- python의 딕셔너리 collections 내장 라이브러리 중 Counter라는 자료구조 활용

 -> version 1.0의 자료 검색

 -> version 1.1의 자료 검색

- 실행 결과



- 실행 완료까지 10분 정도로 획기적으로 단축 (결과의 단위는 초)

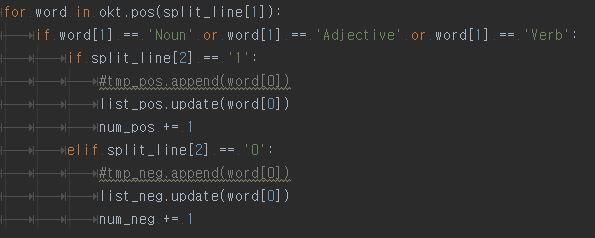
- 하지만 training에 아직 많은 시간이 필요해 개선 필요

- version 1.2

- version 1.1 train()은 읽은 파일의 데이터를 리스트로 저장 후 Counter로 변환

- 이 과정에서 시간 소모

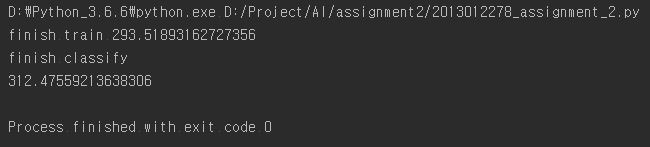
- 처음 데이터를 저장할 시 Counter에 저장하는 것으로 변경

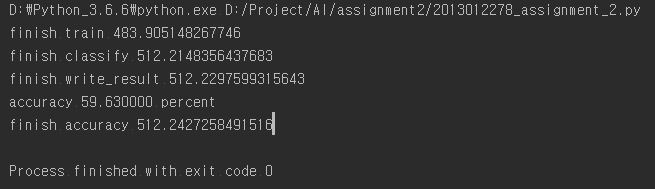


- accuracy() 함수 추가

- ratings\_valid.txt를 통해 실제로 얼마나 정확히 분류했는지 확인

- 실행 결과





- 예상과 같이 시간은 감소

- 하지만 정확도가 낮기에 분류 방식의 개선 필요

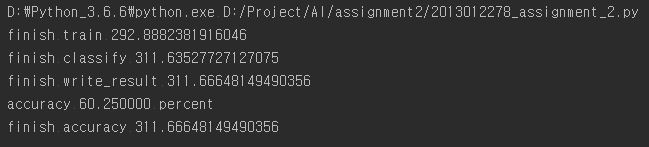
- version 1.3

- 이전 version들은 분류 시 명사, 동사, 형용사만 사용

- 부사와 인터넷 글임을 감안해 이모티콘 추가 분류



- 실행결과



- 미미한 성능 향상

- 근본적인 분류 방식의 개선 필요

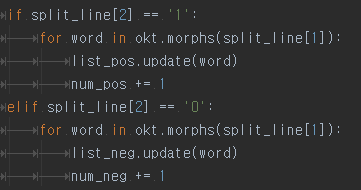
- version 1.4

- 이전 version들은 data 분류 시 okt.pos 사용

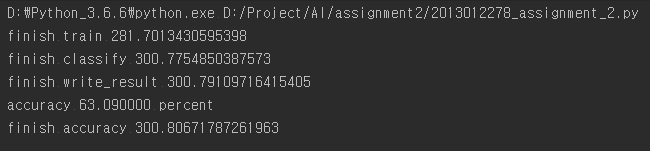
- okt 모듈에 의해 분류된 단어의 종류 중 선택해 학습

- version 1.4는 okt.morphs 사용

- okt 모듈에 의해 문장이 형태소로 분류돼 학습

 -> version 1.4의 train() 함수

- 실행결과

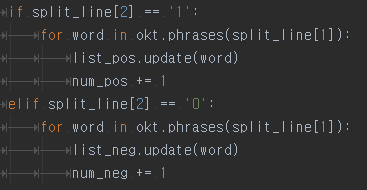


- 미미한 성능 향상

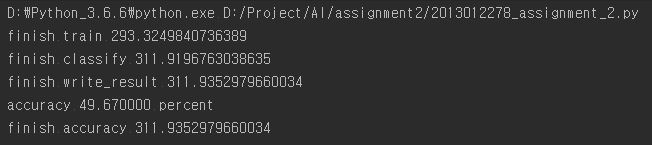
- 추가 개선 시도

- version 1.5

- 분류 방식을 okt.phrases()로 사용

 -> version 1.5의 train()함수

- 실행결과



- 정확도 대폭 감소

- 근본적으로 다른 방식의 개선 필요

**4. version 2.x**

- version 1.x의 학습방식의 문제 발견

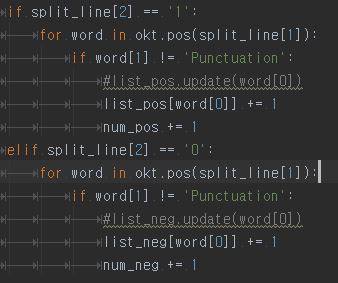
- Counter.update()의 잘못된 활용

- 수많은 허수 데이터가 포함돼 잘못된 학습을 함

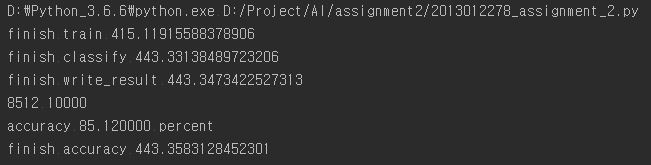
- version 2.0

- train() 함수 수정

- 분류 시 영향을 덜 주는 구두점들은 제외



- 실행결과



- 비약적인 정확도 상승

- 추가 개선으로 정확도 상승 시도

- version 2.1

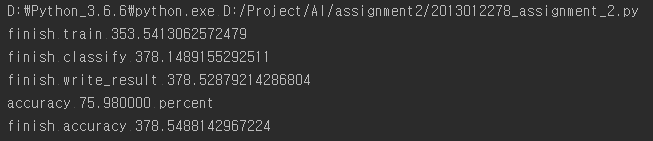
- train()시 구두점도 포함

- 하지만, 실행결과 version 2.0과 똑같은 정확도 도출

- version 2.2

- data 분류 시 okt.pos() 대신 okt.morphs() 사용

- 실행결과



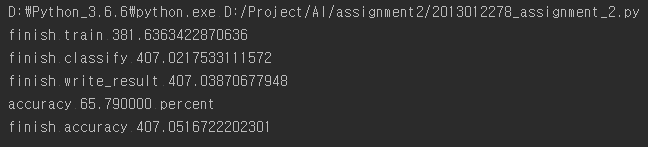
- 정확도 하락

- 다른 시도 필요

- version 2.3

- data 분류 시 okt.phrases() 활용

- 실행결과



- 정확도 하락

- version 2.4

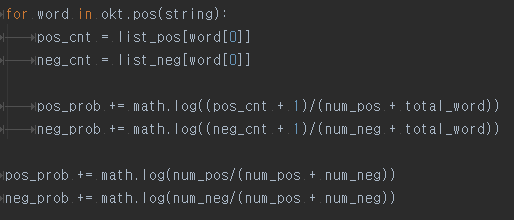
- naïve bayes model을 이용해 확률 계산시 현재까지의 version들은

P(pos)와 P(word)를 제외함

- P(pos|word) = P(word|pos)\*P(pos)/P(word)

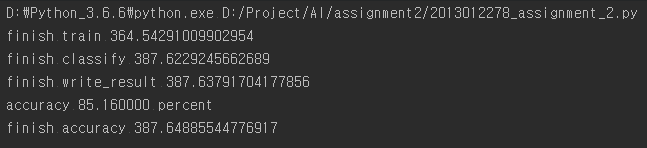
- 이 중 P(pos)를 절반의 확률로 가정하고 제외했으나

실제 데이터의 값을 넣어 확률 계산에 포함



- konlpy의 Okt 중 가장 정확도가 높은 것은 pos()를 활용

- 실행 결과



- 아주 미미한 정확도 상승, 사실상 성능의 차이는 없다고 봐도 무방

**5. version 3.x**

- konlpy의 Okt를 komoran으로 대체

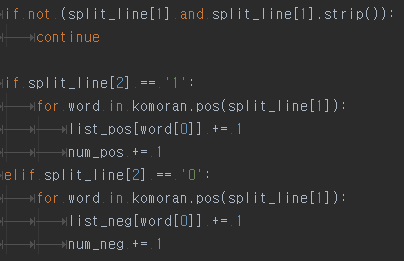
- 현재 komoran 모듈은 windows 내에서 빈 문자열을 parsing 할 시

오류가 발생하므로 빈 문자열은 continue로 parsing 방지

- version 3.0

- Komoran.pos() 활용

- 오류 방지 구문 추가



- 실행결과

- 상당히 오랜 시간 동안 실행시켰지만 결과가 나오지 않음

- 학습에는 3분 밖에 걸리지 않아 오히려 빠른 결과였지만

분류에 너무 많은 시간을 소모

- 정확도를 차치해도 실행완료에 너무 많은 시간이 걸리기에

Komoran 모듈은 제외

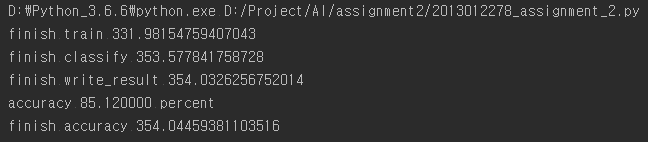
- kkma 모듈도 마찬가지

**6. 결론**

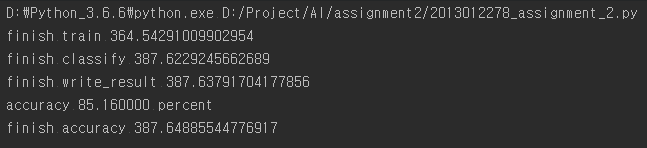
- 정확도와 실행시간을 고려해 가장 적당한 분류 모델은

Konlpy의 pos() 모듈을 이용한 모델로 결론

- version 2.0 & 2.1 & 2.4 실행시간: 5~6분 정확도: 85% 채택



- version 2.0 & 2.1 test ratings\_valid.txt



- version 2.4 test ratings\_valid.txt



- version 2.4 test ratings\_test.txt

- 결과물은 ratings\_result.txt에 저장