Artificial Intelligence HW #2 Report

-영화 리뷰 긍정/부정 분류하기-

2014005178 지현승

■ 코드 설명

ⓐ 전체 구조

① 외부 패키지

이 프로그램은 형태소 분석을 위해 KoNLPy를 사용합니다. 따라서 pip를 통해 KoNLPy를 설치하지 않은 경우 실행할 수 없습니다.

② 코드는 아래의 다섯 가지 함수를 활용합니다.

Learn : ratings_train.txt에서 training data를 추출하여 내부 DB(코드에서 S_DB라는 Dictionary 자료형으로 정의)에 저장한 뒤 이를 return 합니다. Import_Dictionary 함수를 통해서 이미 S_DB를 불러온 경우 이 함수를 실행하지 않습니다.

Calculate : 내부 DB(S_DB)를 바탕으로 ratings_test.txt에 있는 리뷰들의 긍정/부정을 평가하는 함수입니다. 평가 결과를 R_DB(리스트 자료형)에 저장한 뒤 이를 return 합니다.

Export_Dictionary : 내부 DB(S_DB)를 dictionary_twitter.txt로 저장합니다.

Export_Result : R_DB(리뷰들의 긍정/부정 분류 결과)를 ratings_result.txt에 저장하는 함수입니다.

Import_Dictionary : 외부 DB(dictionary_twitter.txt)를 입력 받아 내부 DB(S_DB)를 만들어 이를 return 하는 함수입니다.

③ 코드 실행과정은 다음과 같습니다.

1) 경로 상에 외부 DB(dictionary_twitter.txt)의 존재 여부를 확인합니다.

1-1) 경로 상에 외부 DB가 없는 경우 Learn과 Export_Dictionary 함수를 실행합니다.

1-2) 외부 DB가 존재하는 경우 위 두 함수 대신 Import_Dictionary를 통해

외부 DB를 내부 DB로 전환합니다.

2) Calculate를 실행합니다.

3) Export_Result를 실행합니다.

ⓑ Learn

① 변수를 선언합니다. Learn의 변수 중 함수의 핵심 변수는 다음과 같습니다.

T_DB : 파일을 읽어 나가면서 어느 한 형태소의 긍정/부정 여부를 임시로 등록하는 list 자료형입니다.

S_DB : T_DB에 중복으로 등록된 형태소들을 하나로 합하여 등록합니다.
Dictionary 자료형입니다. 함수는 이 변수를 return합니다.

number_of_bad : training data에서 부정적 평가를 한 리뷰의 개수를 나타냅니다.

number_of_good : training data에서 긍정적 평가를 한 리뷰의 개수를 나타냅니다.

② 파일에서 한 line을 불러온 뒤 이를 line 변수에 저장합니다.

③ String 자료형의 line을 konlpy의 twitter 태그를 사용하여 형태소 집합 리스트로 변환시킵니다. 이제 line에는 형태소들이 저장되어 있습니다.

④ 리스트에서 가장 마지막에 있는 '₩n'를 삭제한 뒤, 평가번호(0 or 1)를 리스트의 처음으로 옮깁니다. 이제 리스트의 첫번째 요소는 한 리뷰의 평가 번호를 나타냅니다..

⑤ 이제 list에 저장된 형태소들을 T_DB에 등록합니다. ['형태소',긍정여부,부정여부] 라는 리스트를 T_DB에 append 합니다. 예를 들어 '아름다움'이라는 형태소가 긍정적 리뷰에서 나온 경우 ['아름다움',0,1]로 등록합니다.

⑥ 같은 리뷰에서 2개 이상 나오는 형태소들은 중복하여 사전에 등재하지 않습니다.

⑦ 모든 리뷰에서 나온 형태소들을 T_DB에 등록한 뒤 T_DB를 sort시켜 줍니다. 이는 중복으로 등재된 형태소들의 S_DB 등재를 쉽게 하기 위해서 입니다(중복된 단어들이 연속적인 형태로 나타남).

⑧ 중복 등록된 형태소들을 찾아가면서 긍정적 반응과 부정적 반응 각각의 누적합을 구한 뒤 (각각 good, bad 변수) 이를 S_DB에 등록시켜 줍니다.

⑨ T_DB에 등록된 모든 형태소들에 대해서 8번 과정을 실행한 뒤 S_DB를 return 합니다.

ⓒ Calculate

① 확률은 log를 적용하여 계산합니다. 아주 작은 수에서 발생하는 컴퓨터의 연산 에러를 방지하기 위해서 입니다.

② 긍정평가의 확률 변수를 Good, 부정평가의 확률 변수를 ㄱGood이라 하고(원인), 단어들을 각각 w1,w2,...,주(ex. w1 ='아름다움') 이라 할 때, conditionally independent하다고 하면 Naïve Bayesian net의 성질에 따라 P(Good | w1,w2,...,wn), P(ㄱGood | w1,w2,...,wn)을 계산한 뒤 대소를 비교하여 주어진 문장이 긍정평가인지 부정평가인지 계산할 수 있습니다.

③ Bayesian net에서

P(Good|w1,w2,...,wn) = P(w1|Good)P(w2|Good).....P(wn|Good)P(Good)

여기에 log를 적용하면 (log N(w1,Good)-log(N(Good))) + (log N(w2,Good)-log(N(Good))) + ... + (log N(wn,Good)-log(N(Good))) + log(P(good))

(* N(x)는 전체 문서에서 x를 만족하는 문서의 개수)

위와 같이 합 연산으로 표현할 수 있습니다. 이 코드에서는 이렇게 로그를 적용하여 합 연산을 적용하였습니다.

number_of_bad(N(Bad))와 number_of_good(N(Good)) 변수는 앞서 설명한 Learn에서의 변수들과 같은 용도(부정적 평가,긍정적 평가 개수)로 사용할 변수들입니다. 이 두 변수를 사용해 prob_bad와 prob_good을 계산합니다.

prob_bad = log(N(Bad)/N(Bad+Good)) = log(P(Bad)) = log(P(ㄱGood))

prob_good = log(N(Good)/N(Bad+Good)) = log(P(Good))

④ Learn 함수에서와 같이 파일에서 string을 불러온 뒤 twitter 태그를 사용하여 형태소 집합으로 바꿉니다. 형태소들을 S_DB에 등록된 형태소들과 비교합니다. 같은 리뷰에서 2개 이상 나오는 형태소들은 S_DB와 중복 비교하지 않습니다.

⑤ P(wi|Good) = 0일 때, 이 값으로 인해 전체 확률값이 0으로 될 위험성이 존재합

니다(로그에서는 무한대). 이를 막기 위해 빈도수에 1을 더해줍니다 (N(wi)+1).

⑥ 형태소가 S_DB에 존재하지 않을 경우 그 형태소는 확률 계산에 포함시키지 않습니다.

⑦ 한 string에서 모든 형태소들의 확률계산이 끝났다면 값을 비교해서 긍정(1)/부정(0)을 분류합니다. 값이 같을 경우 중립 또는 자료부족(-1)으로 평가합니다.

■ 실험 결과

ⓐ 정확도

① ratings_valid.txt를 평가하게 한 뒤 참값과 비교했습니다. 그 결과 정확도는 85.02%가 나왔습니다.

② 대략 15% 정도의 실패율로써 이를 줄이기 위해 다음과 같은 방법을 시도했습니다.

(ㄱ) String을 띄어쓰기 단위로 자른다(list.slice). 결과로 생성된 단어들을 앞에서 n 글자 씩 잘라서 DB에 등록한다.

- 가장 먼저 시도한 방법입니다. 정확도는 83~84% 정도 나왔습니다.
- Ex) '나는 감자칩이 좋다.' -> '나는', '감자', '좋다' (2글자로 자른 경우)

(ㄴ) (ㄱ)을 조금 바꿨습니다. 띄어쓰기 단위로 자른 단어들을 앞에서 1~n 글자 씩 자른 뒤 각각의 결과를 DB에 등록한다.

- 정확도는 83~85%로 (ㄱ) 보다 조금 더 나은 정확도가 나왔습니다.

(ㄷ) tf-idf를 사용하여 단어에 신뢰도를 적용시킴.

- 정확도는 82~84%가 나왔습니다.

③ 이외에도 변수를 바꾸거나 확률 공식을 변환하는 실험을 진행했습니다. 결과적으로는 방법에 따라 82~85% 확률을 보여줬습니다.

④ 형태소 분석을 base로 한 이상 여기에서 정확도가 크게 상승하지 않을 것으로 생각합니다. 예측이 틀린 리뷰를 분석한 결과는 다음과 같습니다.

- 긍정적 단어에 (ex 아름답다) 부정사(ex 않-)가 붙은 경우. '않다' 자체는 부정적인 사용 빈도가 많았지만 '않-'으로 만든 단어 같은 경우 긍정적 사용 빈도 역시 많았습니다. 즉 '않-'은 중립적인 어감이 강하지만 단어 특성상 다른 단어에 붙으면 다른 단어의 어감을 바꿔버리는 특징이 있습니다(ex 아름답지 않다). 이 코드에서는 그런 특징이 반영되지 않았기 때문에 에러가 생긴 것으로 추측합니다.
- 영문 리뷰는 상대적으로 training data가 적어 예측하기 쉽지 않습니다.
- 리뷰에 반어법(혹은 비꼬기)을 사용하는 경우 뉘앙스 판단 기능이 없기 때문에 틀리는 경우가 많았습니다.
- 리뷰 자체가 신뢰성이 떨어지는 경우(ex 광고) 틀리는 경우가 많았습니다.

⑤ 결국 정확한 문법을 분석기에 적용하는 방법, 단어의 여러 조합을 DB에 저장하여 활용하는 방법, 문장 전체를 사용하여 딥러닝을 적용하는 방법 등으로 개선할 수 있을 것으로 예상합니다.

⑹ 소모 시간

외부 DB가 없는 경우 프로그램 종료까지 대략 5분 50초~6분의 시간이 소모됩니다.

외부 DB가 존재하는 경우 프로그램 종료까지 대략 30초의 시간이 소모됩니다.

외부 DB의 존재에 따라 시간 소모량의 차이가 매우 큰 것을 확인할 수 있습니다. 그 원인은 konlpy의 twitter 태그의 성능 문제일 것으로 판단합니다.

반면 twitter 태그가 아닌 '띄어쓰기로 분리한 단어를 n글자씩 누적해서 자르기' 방법을 사용할 경우 5개를 누적해서 자를 때 외부 DB가 없을 경우 50초, 있는 경우 35초 정도 소모 되는 것을 확인했습니다.

정확도는 twitter 방법에 비해 근소하게 떨어지나 그 성능(소모 시간 기준)은 상대적으로 우수합니다. 정확도 차이가 0.1~1% 정도이기 때문에 엄밀한 정확도를 요구하지 않는 이상 '누적 자르기' 방법을 사용하는 것이 더 좋을 것으로 생각합니다. 이번 과제의 경우 정확도를 우선으로 하기 때문에 twitter 태그를 사용한 코드를 제출했습니다.