게임 인공지능 기말 프로젝트 테크니컬 리포트

홍익대학교 게임학부 게임소프트웨어 전공 C077044 황태훈

1. 서론

한국지능정보사회진흥원에서 운영하는 AI Hub에서 제공하는 감성 대화 말뭉치를 통해서 추출한 정보를 기반으로 6가지 감정을 분석을 진행하는 것을 목표로 하는 프로젝트입니다. 본 프로젝트는 네이버 영화 리뷰 감성 분류하기와 KoBERT를 이용해서 참고하면서 프로젝트를 진행했습니다.

감성 대화 말뭉치 데이터셋과 한국어 음성 데이터셋을 사용설명서를 통해서 감정뿐만 아니라 나이, 성별, 상황키워드, 신체질환으로 구분되어서 추가적으로 구분되어 있는 내용 중에서 감정과 사람문장만 추출해서 기말 프로젝트를 진행했습니다.

2. 본론

A. 데이터셋 수집 및 데이터셋 정리

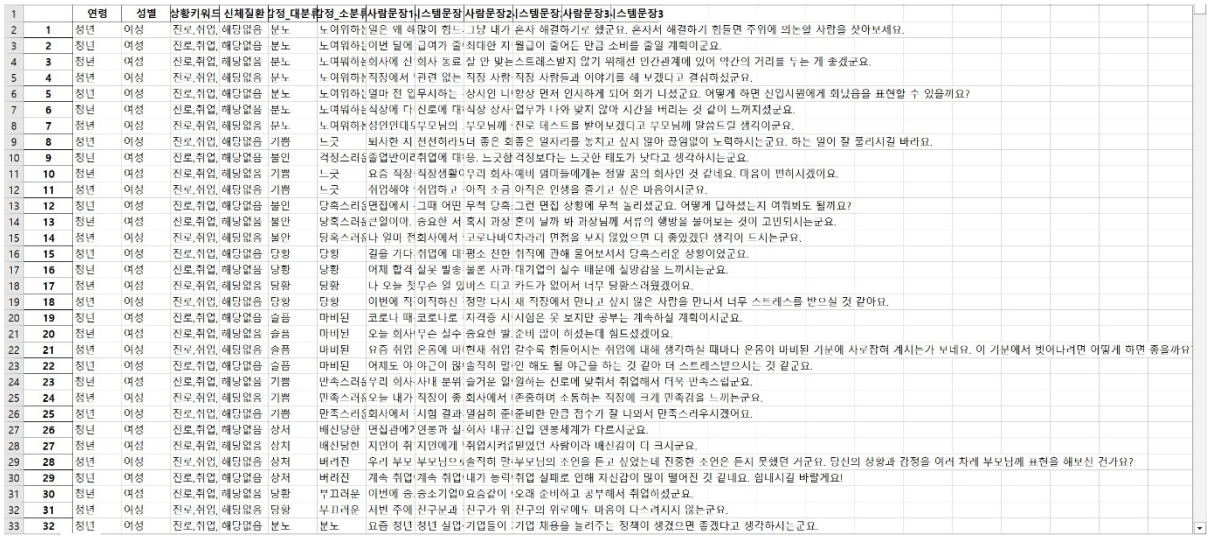
먼저 데이터셋을 만들기 위해서 감정을 분석을 나타나는 데이터셋을 찾아보면서 AI Hub사이트를 알게 되었고 여기 사이트에 있는 '감성 대화 말뭉치' 데이터셋을 사용했습니다. 이 데이터셋은 엑셀에서 기본적으로 재공하는 xlsx파일로 제공해서 데이터셋 정리가 상대적으로 쉬었습니다. 하지만 이 데이터셋도 여러 문제가 있었습니다.

첫 번째로는 너무 많은 구분상황이 많아서 필요한 내용만 추출했습니다. 이 데이터셋은 감성 분석뿐만 아니라 과거에 '심심이'와 같은 인공지능 대화 채팅을 만드는 사용 가능한 데이터셋에서 연령, 나이, 성별, 상황키워드, 신체질환, 감정 대소분류, 사람문장, 시스템문장, 예상 답변 사람문장 등으로 구분되어 있어서 사람문장과 감정 대분류만 추출해서 사용했습니다.

두 번째로는 감정 대분류에 의해서 나뉘어 있는 6가지 종류로 구분되어 있지만 각각의 감정 수가 너무 편차가 심해서 각각 일정한 개수로 조정했습니다. 훈련 데이터셋 경우에는 기쁨은 6126개, 놀람은 8756개, 분노는 9160개, 불안은 9320개, 혐오는 9143개, 슬픔은 9125개로 구성되어 있어서 가장적은 기쁨을 기준으로 모든 감정들을 6126개로 조정해서 데이터셋을 구성했습니다. 테스트 데이터셋 같은 경우 기쁨은 1213개, 놀람은 1048개, 분노는 1257개, 불안은 1113개, 혐오는 1007개, 슬픔은 1003개로 구성되어 있어서 원래는 가장적은 슬픔을 기준으로 1003개로 조정할 생각이었으나 1003개보다는 1000개로 조정해서 데이터셋을 구성했습니다.

또한 한글로 데이터셋을 구분하면 프로그램이 구분하는 문제가 발생이가 높아서 사람문장은 'document'로 감정은 'label'로 명칭으로 변경했습니다. 그리고 기쁨은 '0', 놀람은 '1', 분노는 '2', 불안은 '3', 혐오는 '4', 슬픔은 '5'로 좀더 쉽게 처리하기위해서 변경했습니다.

아래 이미지는 '감성 대화 말뭉치' 데이터셋입니다.



		연령	성별	상황키워드	신체질환	감정_대분류	감정_소분류	사람문장1	시스템문장1	사람문장2	시스템문장2	사람문장3	시스템문장3
1		연령	성별	상황키워드	신체질환	감정_대분류	감정_소분류	사람문장1	시스템문장1	사람문장2	시스템문장2	사람문장3	시스템문장3
2	1	청년	여성	직로,취업,	해당없음	분노		노여워하는	원은 왜 이렇게 힘드...	그냥 내가 혼자 해결하기로 했군요.		혼자서 해결하기 힘들면 주위에 의논할 사람을 찾아보세요.	
3	2	청년	여성	직로,취업,	해당없음	분노		노여워하는	이번 달에 급여가 줄...	최대한 지출을 줄여도 만큼 소비를 줄일 계획이군요.			
4	3	청년	여성	직로,취업,	해당없음	분노		노여워하는	회사에 신입사원 한...	회사 놀로 참 안 맞는스트레스받지 않기 위해선 인간관계에 있어 약간의 거리를 두는 게 좋겠군요.			
5	4	청년	여성	직로,취업,	해당없음	분노		노여워하는	직장에서 권련 없는...	직장 사람들과 이야기를 해 보겠다고 결심하셨군요.			
6	5	청년	여성	직로,취업,	해당없음	분노		노여워하는	열마 전 입무시하는...	상시인 나항상 먼저 인사하게 되어 화기 나셨군요. 어떻게 하면 신입사원에게 화났음을 표현할 수 있을까요?			
7	6	청년	여성	직로,취업,	해당없음	분노		노여워하는	식장에 다니로에 대...	식장 상사·업무가 나와 맞지 않아 시간을 버리는 것 같이 느껴지셨군요.			
8	7	청년	여성	직로,취업,	해당없음	분노		노여워하는	성인인데도부모님의...	부모님께 진로 데스트를 받아보겠다고 부모님께 말씀드릴 생각이군요.			
9	8	청년	여성	직로,취업,	해당없음	기쁨		느긋	퇴사한 지 선천히 5...	5년 동은 회종은 멀저리를 놓치고 싶지 않아 끈임없이 노력하시는군요. 하는 일이 잘 풀리시길 바라요.			
10	9	청년	여성	직로,취업,	해당없음	불안		걱정스러운	졸업반이라취업에 대...	5. 느긋한 걱정보다는 느긋한 태도가 낫다고 생각하시는군요.			
11	10	청년	여성	직로,취업,	해당없음	기쁨		느긋	요즘 직장·직장생활이...	우리 회사·에비 엄마들에게는 정말 꿈의 회사인 것 같네요. 마음이 편하시겠어요.			
12	11	청년	여성	직로,취업,	해당없음	기쁨		느긋	취업해야 하고 있...	아직 조금 아직은 안생을 즐기고 싶은 마음이시군요.			
13	12	청년	여성	직로,취업,	해당없음	불안		당혹스러운	면접에서 그때 어떤 무척 당혹...	그런 면접 상황에 무척 놀라셨군요. 어떻게 답하셨는지 여쭤봐도 될까요?			
14	13	청년	여성	직로,취업,	해당없음	불안		당혹스러운	근엄이야. 승요한 서...	축시 과장·혼이 날까 봐 과장님께 서류의 행방을 묻어보는 것이 고민되시는군요.			
15	14	청년	여성	직로,취업,	해당없음	불안		당혹스러운	나 엄마 전화사에서 ...	고구나네 차라리 면접을 보지 않았으면 더 좋았겠다 생각이 드시는군요.			
16	15	청년	여성	직로,취업,	해당없음	당황		당황	글을 기다·취업에 대...	평소 친한 취직에 관해 물어보셔서 당혹스러운 상황이었군요.			
17	16	청년	여성	직로,취업,	해당없음	당황		당황	여제 합격 실못 발송·불론 사과·대기업이 실수 때문에 실망감을 느끼시는군요.				
18	17	청년	여성	직로,취업,	해당없음	당황		당황	나 오늘 첫무슨 일 있버스 디고 카드가 없어서 너무 당황스러웠겠어요.				
19	18	청년	여성	직로,취업,	해당없음	당황		당황	이번에 직·이직하신...	정말 다시 새 직장에서 만나고 싶지 않은 사람을 만나서 너무 스트레스를 받으실 것 같아요.			
20	19	청년	여성	직로,취업,	해당없음	슬픔		마비된	코로나 때·코로나로 ...	지격증 시·시험은 못 보지만 공부는 계속하실 계획이시군요.			
21	20	청년	여성	직로,취업,	해당없음	슬픔		마비된	오늘 회사무슨 실수 중요한 발...	준비 많이 하셨는데 원도셨겠어요.			
22	21	청년	여성	직로,취업,	해당없음	슬픔		마비된	요즘 취업 온몸에 마·현재 취업 감수록 힘들어시는 취업에 대해 생각하실 때마니 온몸이 마비된 기분에 사로잡혀 계시는가 보네요. 이 기분에서 벗어나려면 어떻게 하면 좋을까요?				
23	22	청년	여성	직로,취업,	해당없음	슬픔		마비된	여제도 아 아근이 많·솔직히 말·인 해도 될 아근을 하는 것 같아 더 스트레스받으시는 것 같군요.				
24	23	청년	여성	직로,취업,	해당없음	기쁨		만족스러운	회사·사내 분위·슬거운 일·월하는 직로에 맞춰서 취업해서 더욱 만족스럽군요.				
25	24	청년	여성	직로,취업,	해당없음	기쁨		만족스러운	우리 오늘 내가 직장이 좋·회사에서 존중하며 소통하는 직장에 크게 만족감을 느끼는군요.				
26	25	청년	여성	직로,취업,	해당없음	기쁨		만족스러운	회사에서 시험 결과·열심히 준·준비한 만큼 점수가 잘 나와서 만족스러우시겠어요.				
27	26	청년	여성	직로,취업,	해당없음	상처		배신당한	면접관에게·연봉까 실·이시 내규·신입 연봉체계가 다르시군요.				
28	27	청년	여성	직로,취업,	해당없음	상처		배신당한	지연이 취·지연에게 '취업시켜줄	뿐였던 사람이라 배신감이 더 크시군요.			
29	28	청년	여성	직로,취업,	해당없음	상처		버려진	우리 부모·부모님으로·솔직히 만·부모님의 조언을 듣고 싶었는데 전중한 조언은 듣시 못했던 거군요. 당신의 상왕과 감정을 이러 차례 부모님께 표현을 해보신 건가요?				
30	29	청년	여성	직로,취업,	해당없음	상처		버려진	계속 취업·계속 취업·내기 능력·취업 실패로 인해 지신감이 많이 떨어진 것 같네요. 쉼내시길 바랄게요!				
31	30	청년	여성	직로,취업,	해당없음	당황		부끄러운	이번에 중·중소기업·요즘같이 '오래 준비하고 공부해서 취업하셨군요.				
32	31	청년	여성	직로,취업,	해당없음	당황		부끄러운	시번 주에 친구분과 ...	친구가 위 친구의 위로에도 마음이 다스려지지 않는군요.			
33	32	청년	여성	직로,취업,	해당없음	분노		분노	요즘 청년 청년 실업·기업들이 ...	기업 채용을 늘려주는 정책이 생겼으면 좋겠다고 생각하시는군요.			

B. 데이터셋 내용 정리 진행 후 데이터셋

데이터셋 정리가 끝난 데이터셋 기반으로 모델에 학습을 진행했습니다. 아래에 있는 1개의 이미지는 훈련 데이터셋 일부 사진입니다.

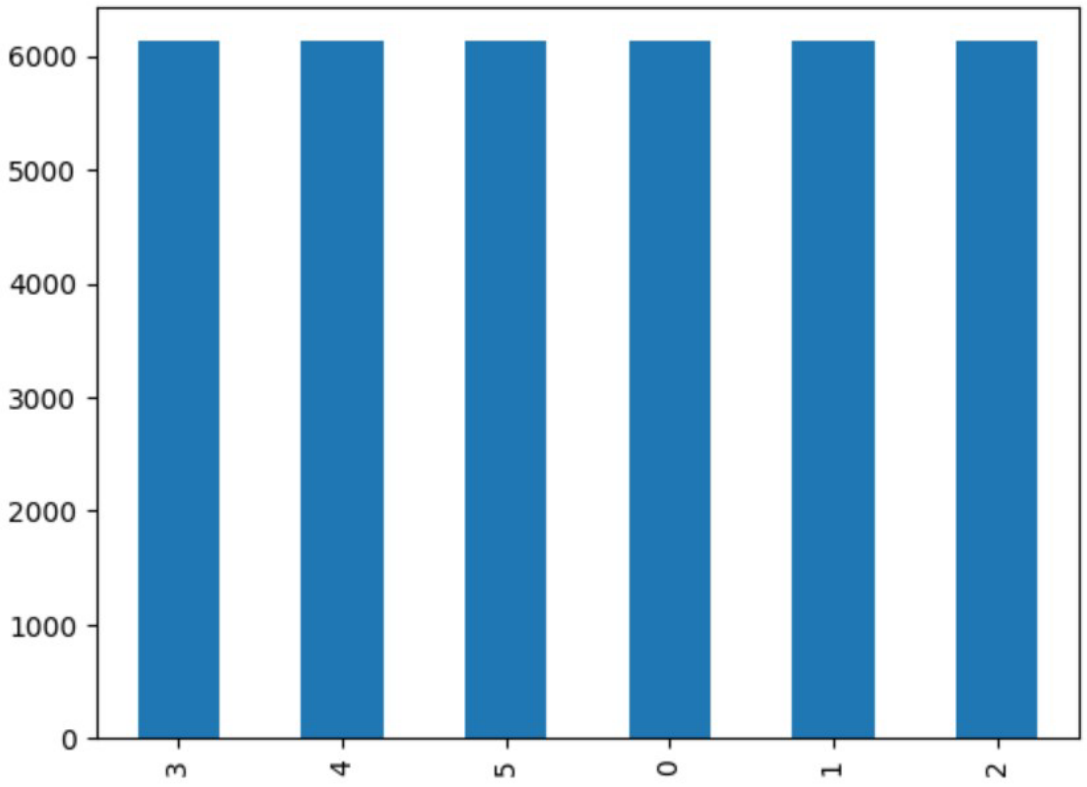
```
document                                label
퇴사한 지 얼마 안 됐지만 천천히 직장을 구해보려고. 0
요즘 직장생활이 너무 편하고 좋은 것 같아! 0
취업해야 할 나이인데 취업하고 싶지가 않아. 0
우리 회사는 정말 사내 분위기가 좋아. 0
오늘 내가 다니는 회사가 참 좋은 직장이라는 생각이 들었어. 0
회사에서 전공시험을 봤어. 오늘 시험 결과가 나왔어. 0
오늘 입사 면접을 봤어. 0
회사에서 나를 참 신뢰하는 것 같아. 그건 기분이 좋아. 0
먼저 취업한 선배가 면접 비결을 알려줬어. 곧 면접 보러 가는데 든든해. 0
직장 상사로부터 칭찬을 받았는데 너무 신이 나! 0
취업해서 너무 신이 나! 0
오늘 사장님께 칭찬받아서 매우 기뻐. 0
지난주에 정말 가고 싶던 회사에 원서 접수했는데 합격했어. 매우 기뻐. 0
나 드디어 원하는 회사에 취업했어! 0
나 오늘 네이버 인턴 면접 봤는데 내가 봐도 잘 본 것 같아. 0
결혼한 지 삼 년 만에 아이를 낳았어. 진짜 기뻐. 0
그녀가 나에게 사귀자고 고백했어. 나도 원하고 있었는데 지금 기분이 너무 좋아. 0
```

C. 데이터셋 분석

준비된 두 가지 데이터셋을 데이터셋 정리를 진행한 후에 훈련 데이터셋 안에 중복 데이터셋을 제거한 후에 데이터셋을 구성합니다. 아래 있는 이미지는 훈련 데이터셋을 구성을 그래프와 수치로 표시한 이미지입니다.

```
1 train_data['label'].value_counts().plot(kind='bar')
```

```
<Axes: >
```




```
[14]  1 print(train_data.groupby('label').size().reset_index(name = 'count'))
```

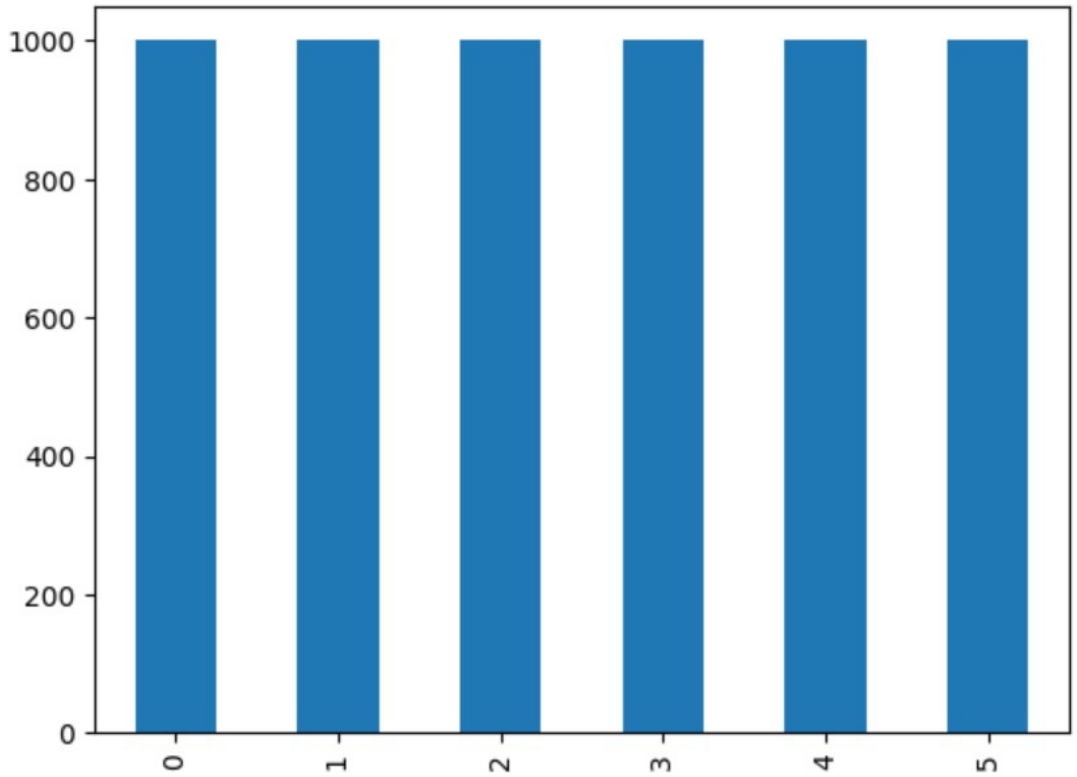
```
   label  count
0      0   6125
1      1   6125
2      2   6125
3      3   6126
4      4   6126
5      5   6126
```

위 이미지에 알 수 있듯이 label 0(기쁨), 1(놀람), 2(분노)에서 중복 값이 하나씩 나왔지만 모든 감정에 대한 정보가 거의 1:1:1:1:1:1로 나누어서 데이터셋 편중이 없다는 가정하에 이상 데이터셋 정리가 없이 진행되었습니다. 훈련 데이터셋 중복 제거화를 거친 후에 테스트 데이터셋 중복 데이터셋화를 진행했습니다.

아래 있는 이미지는 테스트 데이터셋을 구성을 그래프 및 수치로 낸 것입니다.

```
1 test_data['label'].value_counts().plot(kind='bar')
```

```
<Axes: >
```




```
[28]  1 print(test_data.groupby('label').size().reset_index(name = 'count'))
```

```
   label  count
0      0   1000
1      1   1000
2      2   1000
3      3   1000
4      4   1000
5      5   1000
```

위 이미지에 알 수 있듯이 모든 데이터셋 정확하게 1:1:1:1:1:1로 정확하게 균일함을 알 수 있어서 더 이상 데이터셋 정리 없이 내용 분석을 진행했습니다.

D. 데이터셋 패딩

데이터셋 패딩을 위해서 훈련 데이터셋에 있는 불용어 리스트를 통해서 불용어를 제거하고 토큰화를 진행했습니다. 아래의 이미지는 토큰화를 진행한 후 단어집합의 결과입니다.

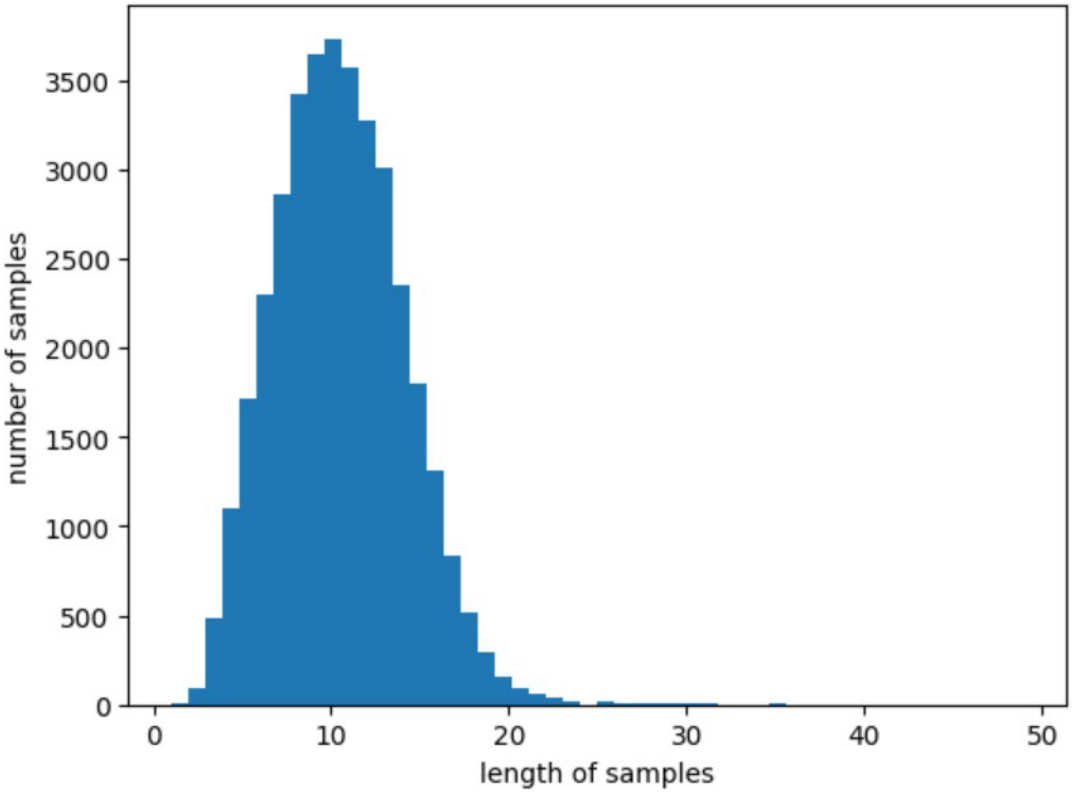
```
단어 집합(vocabulary)의 크기 : 9711
등장 빈도가 2번 이하인 희귀 단어의 수: 4386
단어 집합에서 희귀 단어의 비율: 45.165276490577696
전체 등장 빈도에서 희귀 단어 등장 빈도 비율: 0.14708338579581515
```

```
[26]  1 vocab_size = total_cnt - rare_cnt + 1
      2 print('단어 집합의 크기 :',vocab_size)

단어 집합의 크기 : 5326
```

전체 단어집합에서 중복된 단어를 제외하니 단어의 집합의 크기가 5326인걸 확인할 수 있습니다. 단어집합의 결과 후에 전체 단어 길이의 샘플들의 길이를 동일하게 위해서 전체 데이터셋에서 최대길이와 평균 길이를 해보았습니다. 아래 이미지는 최대 길이, 평균 길이와 문장 길이 분포를 그래프로 나타낸 것입니다.

```
문장의 최대 길이 : 49
문장의 평균 길이 : 10.432971458112263
```



이 이미지로 알 수 있듯이 문장의 최대 길이는 49자이며, 평균길이는 약 10자인 것을 알 수 있습니다. 그리고 20자 이내면 전체 데이터셋에 거의 99퍼센트 정도는 차지한 것을 알 수 있습니다.

```
1 max_len = 17
2 below_threshold_len(max_len, X_train)
```

```
전체 샘플 중 길이가 17 이하인 샘플의 비율: 96.57715016461242
```

위 이미지를 통해서 길이가 17자이하면 샘플의 비율의 96%를 차지하므로 데이터셋의 모든 문장들을 17자로 맞추었습니다.

E. LSMT 학습

임베팅 벡터의 차원은 100, 은닉 상태의 크기는 128, 배치 크기는 64, 에포크는 15회, 훈련데이터셋은 20%를 검증 데이터셋 분리해서 LSMT 학습을 진행해서 정확도를 분석했습니다. 아래의 두 개의 이미지는 학습 완료 및 정확도입니다.

```
Epoch 13/15
460/460 [==============================] - ETA: 0s - loss: -680.1390 - acc: 0.2083
Epoch 13: val_acc did not improve from 0.00000
460/460 [==============================] - 41s 89ms/step - loss: -680.1390 - acc: 0.2083 - val_loss: -2956.2969 - val_acc: 0.0000e+00
Epoch 14/15
460/460 [==============================] - ETA: 0s - loss: -733.5145 - acc: 0.2083
Epoch 14: val_acc did not improve from 0.00000
460/460 [==============================] - 30s 66ms/step - loss: -733.5145 - acc: 0.2083 - val_loss: -3179.6814 - val_acc: 0.0000e+00
Epoch 15/15
460/460 [==============================] - ETA: 0s - loss: -786.7718 - acc: 0.2083
Epoch 15: val_acc did not improve from 0.00000
460/460 [==============================] - 25s 54ms/step - loss: -786.7718 - acc: 0.2083 - val_loss: -3402.1113 - val_acc: 0.0000e+00
```

```
[38] 1 loaded_model = load_model('best_model.h5')
     2 print("₩n 테스트 정확도: %.4f" % (loaded_model.evaluate(X_test, y_test)[1]))
```

```
188/188 [==============================] - 3s 13ms/step - loss: -108.2141 - acc: 0.1667

테스트 정확도: 0.1667
```

이미지를 통해서 정확도가 16.67%를 얻어 내었습니다. 이 정확도를 보아 LSMT학습으로 감정 분석을 할 수 없을 알 수 있었습니다.

F. LSMT 학습 결과 및 문제점과 개선 방법 구상

아래 이미지는 LSMT의 단어 분석을 나타낸 결과입니다.

```
1 sentiment_predict('기쁨')
2 sentiment_predict('놀람')
3 sentiment_predict('분노')
4 sentiment_predict('불안')
5 sentiment_predict('혐오')
6 sentiment_predict('슬픔')
```

```
1/1 [==============================] - 0s 50ms/step
1.0
1/1 [==============================] - 0s 37ms/step
1.0
1/1 [==============================] - 0s 37ms/step
1.0
1/1 [==============================] - 0s 39ms/step
1.0
1/1 [==============================] - 0s 37ms/step
1.0
1/1 [==============================] - 0s 36ms/step
1.0
```

LSMT학습으로는 기본적인 감정을 분석하지 못하고 다 '놀람'이라는 값인 '1'로 출력했습니다.

아래 이미지는 문장 분석을 나타낸 결과입니다.

```
[46]  1 sentiment_predict('나 오늘 합격했어')

      1/1 [==============================] - 0s 22ms/step
      1.0
```

```
[47]  1 sentiment_predict('짜증나! 오늘 과제 너무 많아')

      1/1 [==============================] - 0s 22ms/step
      1.0
```

```
[48]  1 sentiment_predict('앗, 깜짝이야!')

      1/1 [==============================] - 0s 38ms/step
      1.0
```

```
[49]  1 sentiment_predict('오늘 큰이모께서 돌아가셨어!')

      1/1 [==============================] - 0s 46ms/step
      1.0
```

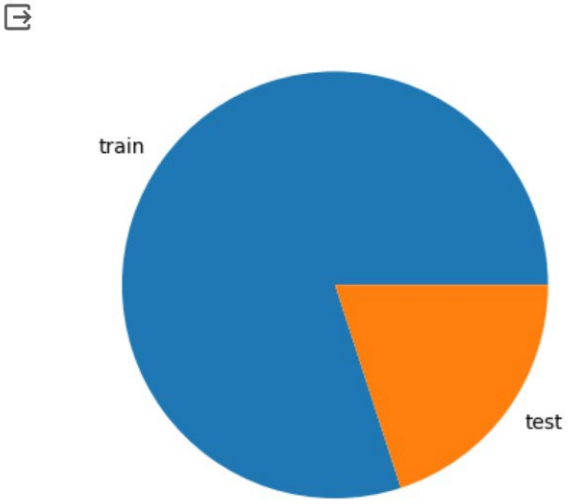
마찬가지로 문장에 대한 감정 분석도 하지 못합니다. 그런 이유로 에포크와 배치 크기, 은닉 크기 등 값을 변경하면 정확도가 늘어날까 생각도 했지만 LSMT의 학습에 이미지를 보아 에포크가 증가한다고 해도 변함이 없다는 것을 알 수 있습니다. 그러므로 개선 방법으로 LSMT 학습이 아닌 다른 학습 방법인 KoBERT 학습을 사용했습니다.

G. 테스트 데이터셋 생성 및 토큰화 및 패딩

위 동일한 훈련 데이터셋을 사용하기 때문에 데이터셋 분석은 진행하지 않고 진행하였고 KoBERT는 테스트 데이터셋(검증 데이터셋)를 훈련 데이터셋을 20%를 이용해서 만들어서 사용했습니다. 아래 이미지는 테스트 데이터셋과 훈련 데이터셋 수치를 보여주고 있습니다.

```
훈련 데이터 수:  29404
테스트 데이터 수:  7352
```

다음 페이지에 있는 이미지는 데이터셋과 훈련 데이터셋 수치를 파이 그래프로 보여준 것입니다.



위 두 이미지를 통해서 train(훈련 데이터셋)과 test(검증 데이터셋)의 비율 80%:20%임을 알 수 있습니다. 토큰화와 패딩은 KoBERT에 제공하는 방식으로 진행했습니다.

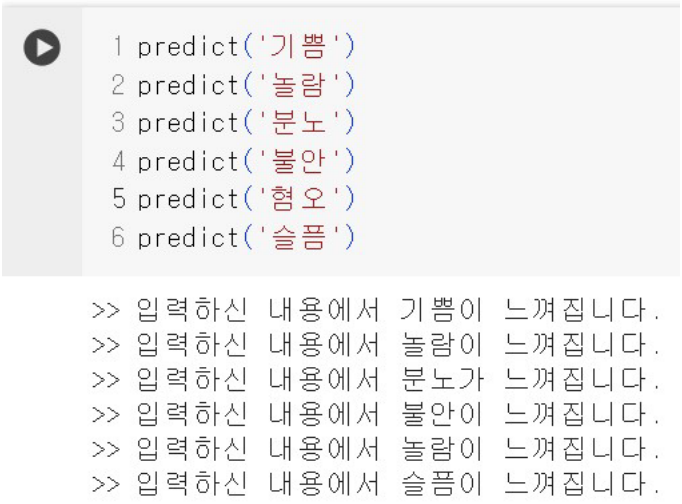
H. KoBERT 학습

위의 작업을 완료하고 에포크는 5회, 배치크기는 64로 KoBERT 학습을 진행했습니다. 아래 이미지는 KoBERT학습완료 것입니다.

```
100% 460/460 [05:02<00:00, 1.80it/s]
epoch 3 batch id 1 loss 0.9763628840446472 train acc 0.671875
epoch 3 batch id 201 loss 0.7148489952087402 train acc 0.6671330845771144
epoch 3 batch id 401 loss 0.8296995759010315 train acc 0.6896430798004988
epoch 3 train acc 0.6946622670807453
100% 115/115 [00:25<00:00, 4.83it/s]
epoch 3 test acc 0.6242041925465838
100% 460/460 [05:01<00:00, 1.79it/s]
epoch 4 batch id 1 loss 0.7871519923210144 train acc 0.71875
epoch 4 batch id 201 loss 0.5586015582084656 train acc 0.7309546019900498
epoch 4 batch id 401 loss 0.6954258680343628 train acc 0.7531951371571073
epoch 4 train acc 0.7565945263975156
100% 115/115 [00:25<00:00, 4.68it/s]
epoch 4 test acc 0.6314052795031055
100% 460/460 [05:01<00:00, 1.77it/s]
epoch 5 batch id 1 loss 0.6367465853691101 train acc 0.796875
epoch 5 batch id 201 loss 0.4930635988712311 train acc 0.7838152985074627
epoch 5 batch id 401 loss 0.5313295722007751 train acc 0.7998753117206983
epoch 5 train acc 0.801848796583851
100% 115/115 [00:24<00:00, 4.75it/s]
epoch 5 test acc 0.6280085403726707
```

위 이미지를 통해서 정확도는 62.8%로 LSMT학습보다 약 6배 정도 차이가 났습니다. 그렇지만 62% 또한 낮은 정확도여서 에포크와 배치크기를 올릴까 생각도 해보았지만 KoBERT학습과정이

결린 시간이 30분 정도로 오래 걸리는 문제와 각각의 에포크 횟수마다 나오는 정확도를 보았을 때 수치를 조정해도 개선이 안된다고 생각해서 수치 변경 없이 결과를 출력해 보았습니다.
아래의 이미지는 단어를 감성분석한 결과입니다.



위의 결과를 통해서 단어 분석이 '혐오'에 대한 예측 부분이 제외하면 모든 단어의 감정분석이 맞는 것을 알 수 있었습니다. 아래의 이미지는 문장에 대한 감성 분석한 결과입니다.

'앗, 깜짝이야!'를 제외하면 모든 문장의 감정분석이 맞는 것을 알 수 있었습니다. 두 결과를 통해서 정확도가 62.8%인 것을 알 수 있었습니다.

I. 추가적인 작업

긍정적인 감정과 부정적인 감정을 구분해보았습니다. 긍정적인 감정은 기쁨, 부정적인 감정은 기쁨을 제외한 감정들로 규정했습니다. 그리고 긍정적인 감정의 비율이 더 높으면 긍정적인 비율로

나타나도록 출력했고 부정적인 감정의 비율이 더 높으면 부정적인 비율 나타나도록 출력했습니다. 아래 두개의 이미지는 긍정적인 감정과 부정적인 감정 결과입니다.

```
[39]  1 predict('기쁨')
      2 predict('놀람')
      3 predict('분노')
      4 predict('불안')
      5 predict('혐오')
      6 predict('슬픔')
```

```
>> 입력하신 내용에서 기쁨이 느껴집니다.
100.00% 확률로 긍정 문장입니다.
>> 입력하신 내용에서 놀람이 느껴집니다.
100.00% 확률로 부정 문장입니다.
>> 입력하신 내용에서 분노가 느껴집니다.
100.00% 확률로 부정 문장입니다.
>> 입력하신 내용에서 불안이 느껴집니다.
100.00% 확률로 부정 문장입니다.
>> 입력하신 내용에서 놀람이 느껴집니다.
100.00% 확률로 부정 문장입니다.
>> 입력하신 내용에서 슬픔이 느껴집니다.
100.00% 확률로 부정 문장입니다.
```

```
[40]  1 predict('나 오늘 합격했어.')
```

```
>> 입력하신 내용에서 기쁨이 느껴집니다.
100.00% 확률로 긍정 문장입니다.
```

```
[41]  1 predict('짜증나! 오늘 과제 너무 많아')
```

```
>> 입력하신 내용에서 분노가 느껴집니다.
100.00% 확률로 부정 문장입니다.
```

```
[42]  1 predict('앗, 깜짝이야!')
```

```
>> 입력하신 내용에서 기쁨이 느껴집니다.
77.03% 확률로 긍정 문장입니다.
```

```
      1 predict('오늘 큰이모께서 돌아가셨어!')
```

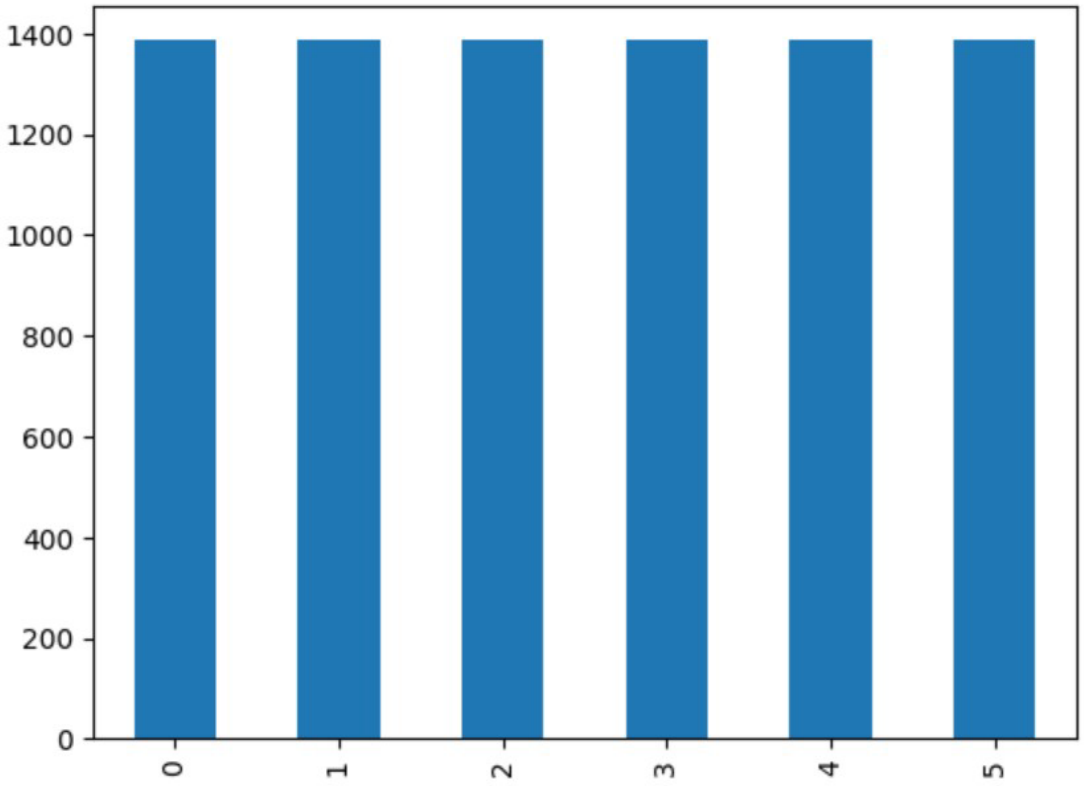
```
>> 입력하신 내용에서 슬픔이 느껴집니다.
100.00% 확률로 부정 문장입니다.
```

J. 새로운 데이터셋으로 KoBERT학습

위에 기존의 데이터셋 정확도가 62.8%여서 마음에 들지 않아서 새로운 훈련 데이터셋을 이용해서 KoBERT학습을 진행했습니다. 먼저 훈련 데이터셋이 균일한지 확인해보았습니다. 아래의 이미지는 훈련 데이터셋 수치 그래프 및 수치입니다.

```
1 train_data['label'].value_counts().plot(kind='bar')
```

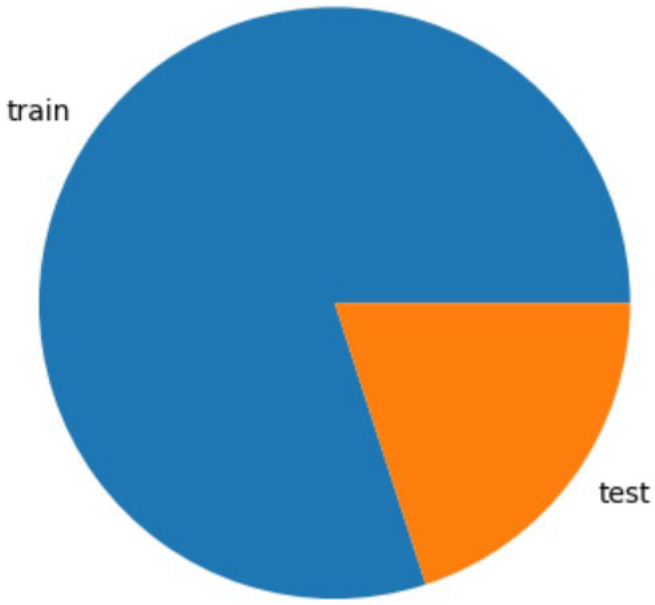
```
<Axes: >
```




```
[9]  1 print(train_data.groupby('label').size().reset_index(name = 'count'))
```

```
   label  count
0      0   1386
1      1   1386
2      2   1386
3      3   1386
4      4   1386
5      5   1386
```

그 다음에는 훈련 데이터셋을 테스트 데이터셋으로 만들어 보았습니다. 다음페이지 이미지는 훈련 데이터셋과 테스트 데이터셋을 보여주는 파이 그래프와 수치입니다.



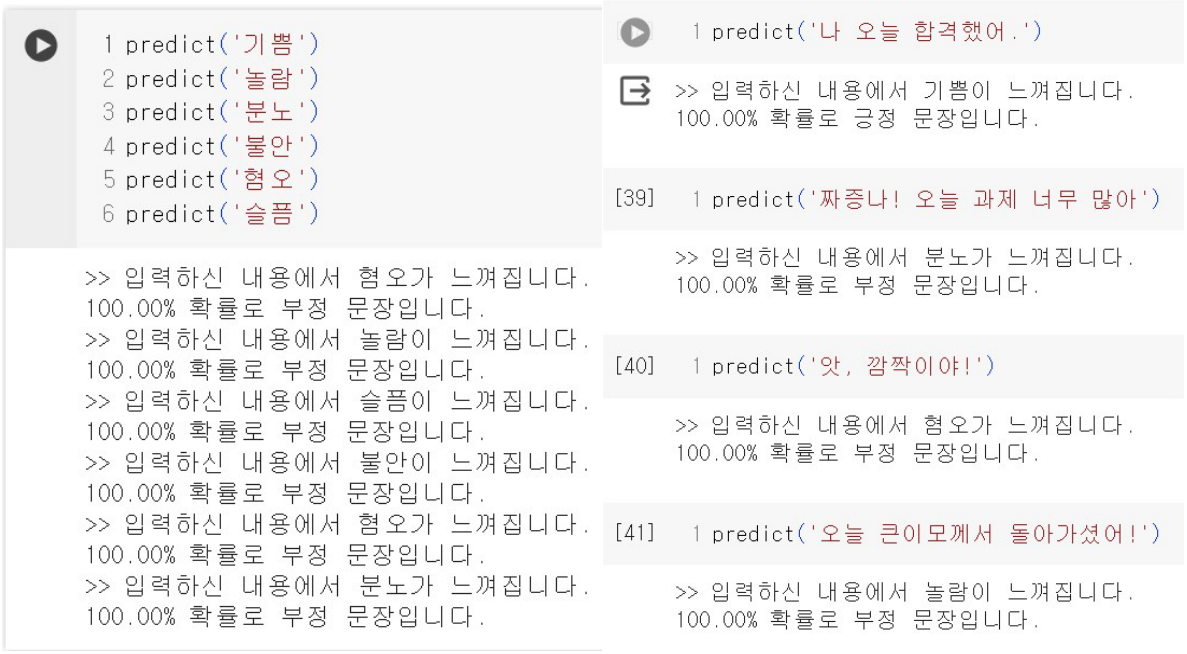

```
[15]  1 print('훈련 데이터 수: ', len(dataset_train))
      2 print('테스트 데이터 수: ', len(dataset_test))
```

```
훈련 데이터 수:  6652
테스트 데이터 수:  1664
```

나머지 모든 과정을 거치고나서 위와 동일한 값으로 KoBERT학습을 진행했습니다. 아래 이미지는 KoBERT학습의 결과입니다.

```
100% 104/104 [01:06<00:00, 1.59it/s]
epoch 3 batch id 1 loss 0.30741211771965027 train acc 0.90625
epoch 3 train acc 0.960917467948718
100% 26/26 [00:05<00:00, 4.87it/s]
epoch 3 test acc 0.9483173076923077
100% 104/104 [01:06<00:00, 1.61it/s]
epoch 4 batch id 1 loss 0.13945536315441132 train acc 0.9375
epoch 4 train acc 0.9792668269230769
100% 26/26 [00:05<00:00, 4.67it/s]
epoch 4 test acc 0.9537259615384616
100% 104/104 [01:07<00:00, 1.61it/s]
epoch 5 batch id 1 loss 0.093752421438694 train acc 0.953125
epoch 5 train acc 0.9885817307692307
100% 26/26 [00:05<00:00, 4.86it/s]
epoch 5 test acc 0.9543269230769231
```

위 이미지에 정확도가 95.4%로 정확도 나왔습니다. 결과를 출력해보았습니다. 아래 두개의 이미지는 결과입니다.



결과를 통해서 "불안", "놀람", "혐오", "나 오늘 합격했어.", "짜증나! 오늘 과제 너무 많아"을 제외하고 전부 감정이 틀렸다는 것을 알 수 있습니다. 이 결과를 보면 정확도가 95%가 아닌 50%정도라고 알 수 있었습니다.

3. 결론

총 3개의 결과 값을 통해서 알 수 있는 점은 첫 번째 6가지의 감정 분류는 LSTM 학습으로는 감정분석이 힘들다는 것을 알 수 있었습니다. 두 번째로는 정확도에 대한 생각입니다. 새로운 데이터셋은 기존 데이터셋보다 정확도는 높지만 결과는 기존 데이터셋보다 못한 것을 알 수 있습니다. 저는 왜 이런 결과가 나왔는지에 대해서 조사해보았습니다. 첫번째 가정으로는 KoBERT가 정확도 분석이 문제가 있는가에 대해서 가정해보았지만 여러가지 자료를 찾아보니 KoBERT 많은 대규모 말뭉치를 학습한 프로그램이어서 KoBERT문제는 아니라는 것을 알 수 있었다. 두번째 가정으로 기존 데이터셋 보다 새로운 데이터셋이 적어서 정확도가 비정상적으로 높게 나왔다는 가정이다. 새로운 데이터셋 크기는 8316, 기존의 데이터셋은 36756으로 거의 4배 정도 차이가 나는 것을 알 수 있었다. 데이터셋 수가 크게 차이가 나는데 어떻게 해서 높게 정확도가 높게 나오는 것이 말이 되지 않는다. 그렇다면 결론은 KoBERT는 새로운 데이터셋에 20%의 검증데이터셋을 사용하므로 검증 데이터셋 수가 적어서 학습할 때 정확도 높게 나왔다는 결론이 나온다.

향후 이 프로젝트를 진행하면서 AI 감정분석은 데이터셋 보다는 학습할 프로그램도 중요하지만 데이터셋 양 또한 무시하지 못하는 것을 알 수 있었습니다. 그리고 더 정확한 데이터셋을 분석을 위해서 60000개 정도가 필요하다는 것으로 예상됩니다.

4. 참고자료

A. [Python, KoBERT] 7가지 감정의 다중감성분류모델 구현하기
https://velog.io/@fhflwhwl5/Python-KoBERT-7%EA%B0%80%EC%A7%80-%EA%B0%90%EC%A0%95%EC%9D%98-%EB%8B%A4%EC%A4%91%EA%B0%90%EC%84%B1%EB%B6%84%EB%A5%98%EB%AA%A8%EB%8D%B8-%EA%B5%AC%ED%98%84%ED%95%98%EA%B8%B0

B. 10-06 네이버 영화 리뷰 감성 분류하기(Naver Movie Review Sentiment Analysis)
https://wikidocs.net/44249

5. 부록

A. 기말프로젝트 실습 진행 LSTM 사용한 collab 링크
https://colab.research.google.com/drive/1vIiS0yTmQvvG2cK7_rjUrdf_zHBRodFs?usp=sharing

B. 기말프로젝트 실습 진행 KoBERT 사용한 collab 링크
https://colab.research.google.com/drive/1d8PtjkV7zg96uJSBAj83BMBPxr08iBJt?usp=sharing

C. 사용한 감성 데이터셋 원본
https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&dataSetSn=86

D. 새로운 감성 데이터셋 원본
https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&dataSetSn=123