VADER을 활용한 스타벅스 감정 분석

1. 데이터 불러오기 및 전처리

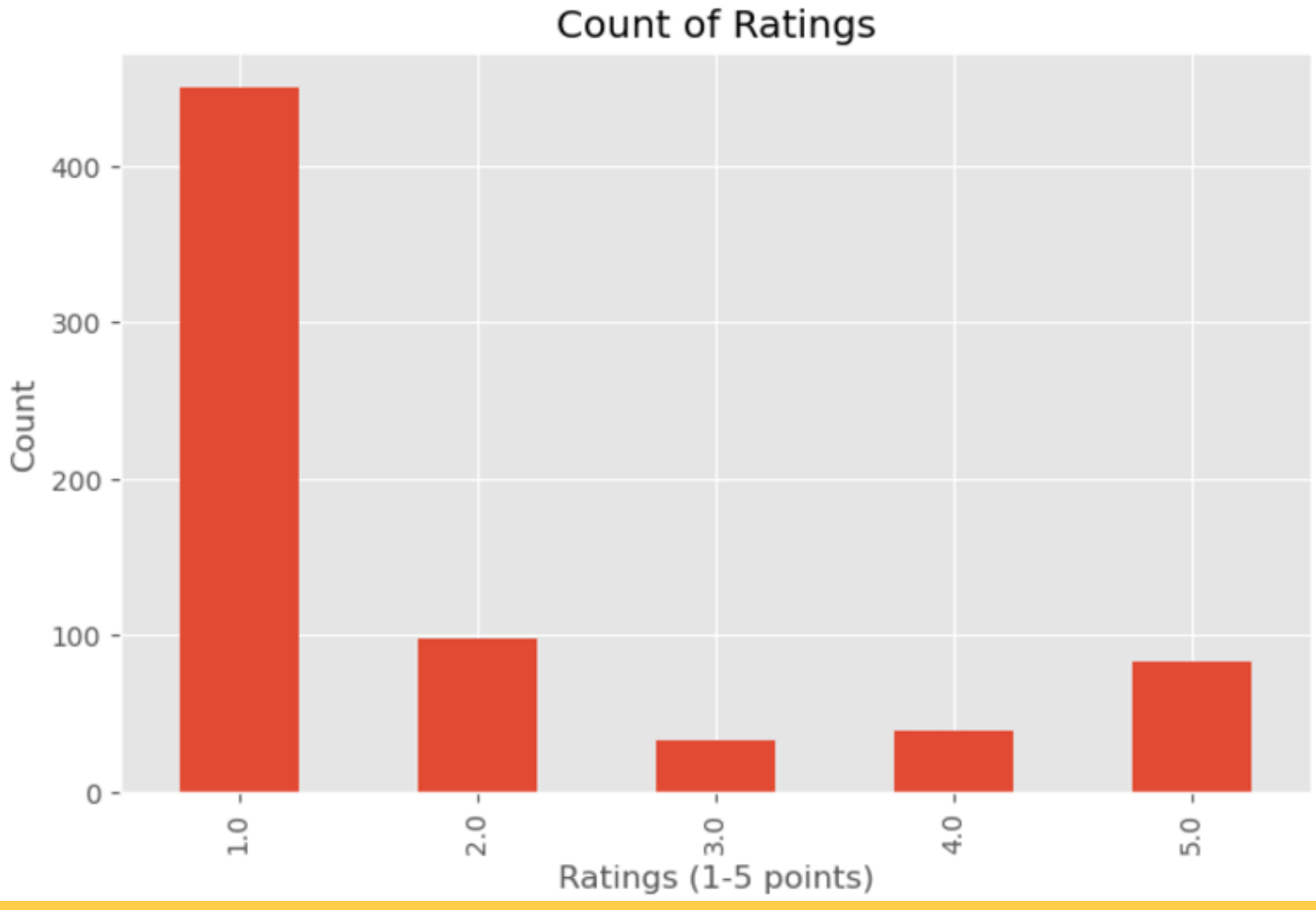
In [35]:
```
df = pd.read_csv("/Users/yeong-eun/Desktop/영은/쿠글/2차프로젝트/reviews_data.csv")
df = df[['Review', 'Rating']]
df = df[df['Review'] != 'No Review Text']
df = df.dropna(subset=['Rating', 'Review']).reset_index(drop=True)
df['ID'] = range(1, len(df) + 1)
df
```

Out[35]:

	Review	Rating	ID
0	Amber and LaDonna at the Starbucks on Southwes...	5.0	1
1	** at the Starbucks by the fire station on 436...	5.0	2
2	I just wanted to go out of my way to recognize...	5.0	3
3	Me and my friend were at Starbucks and my card...	5.0	4
4	I'm on this kick of drinking 5 cups of warm wa...	5.0	5

2. 별점 분포 시각화

In [36]:
```
plt.style.use('ggplot')
df['Rating'].value_counts().sort_index().plot(kind='bar', title='Count of Ratings', figsize=(8, 5))
plt.xlabel('Ratings (1-5 points)')
plt.ylabel('Count')
plt.show()
```



3. VADER 이용 감성 분석

*** compound 점수: 리뷰의 전반적인 긍정/부정 정도를 나타내는 점수(-1~1)**

In [39]:
```
SIA = SentimentIntensityAnalyzer()
df['vader_compound'] = df['Review'].apply(lambda x: SIA.polarity_scores(x)['compound'])

def map_compound_to_sentiment(compound):
    if compound > 0.05:
        return 'Positive'
    elif compound >= -0.05:
        return 'Neutral'
    else:
        return 'Negative'

df['predicted_sentiment'] = df['vader_compound'].apply(map_compound_to_sentiment)
df
```

Out [39]:

	Review	Rating	ID	vader_compound	predicted_sentiment
0	Amber and LaDonna at the Starbucks on Southwes...	5.0	1	0.8991	Positive
1	** at the Starbucks by the fire station on 436...	5.0	2	0.7766	Positive
2	I just wanted to go out of my way to recognize...	5.0	3	0.5242	Positive
3	Me and my friend were at Starbucks and my card...	5.0	4	0.9698	Positive
4	I'm on this kick of drinking 5 cups of warm wa...	5.0	5	0.9793	Positive

4. 별점을 기준으로 정답 감정 레이블 생성

In [40]:
```
def map_rating_to_sentiment(rating):
    if rating >= 4:
        return 'Positive'
    elif rating == 3:
        return 'Neutral'
    else:
        return 'Negative'

df['true_sentiment'] = df['Rating'].apply(map_rating_to_sentiment)
df #결과
```

Out [40]:

	Review	Rating	ID	vader_compound	predicted_sentiment	true_sentiment
0	Amber and LaDonna at the Starbucks on Southwes...	5.0	1	0.8991	Positive	Positive
1	** at the Starbucks by the fire station on 436...	5.0	2	0.7766	Positive	Positive
2	I just wanted to go out of my way to recognize...	5.0	3	0.5242	Positive	Positive
3	Me and my friend were at Starbucks and my card...	5.0	4	0.9698	Positive	Positive
4	I'm on this kick of drinking 5 cups of warm wa...	5.0	5	0.9793	Positive	Positive

5. RoBERTa 이용 감성분석

In [41]:
```
# RoBERTa 사전 학습 모델 초기화
MODEL = "cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

리뷰 텍스트를 입력하면 RoBERTa가 부정, 중립, 긍정 점수를 계산.

softmax 함수를 사용해 점수를 확률로 변환.

In [43]:
```
def polarity_scores_roberta(text):
    encoded_text = tokenizer(text, return_tensors='pt')
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    return {
        'roberta_neg': scores[0], # 부정
        'roberta_neu': scores[1], # 중립
        'roberta_pos': scores[2] # 긍정
    }
```

각 리뷰에 대해 VADER와 RoBERTa의 감정 점수를 계산하기

그 후 결과를 병합하여 새로운 데이터프레임 생성.

```
# VADER 및 RoBERTa 결과 병합
results = {}
for _, row in tqdm(df.iterrows(), total=len(df)):
    vader_result = SIA.polarity_scores(row['Review'])
    roberta_result = polarity_scores_roberta(row['Review'])
    results[row['ID']] = {**vader_result, **roberta_result}

results_df = pd.DataFrame(results).T.reset_index().rename(columns={'index': 'ID'})
results_df = results_df.merge(df, how='left')
results_df
```

100%|

Out[44]:

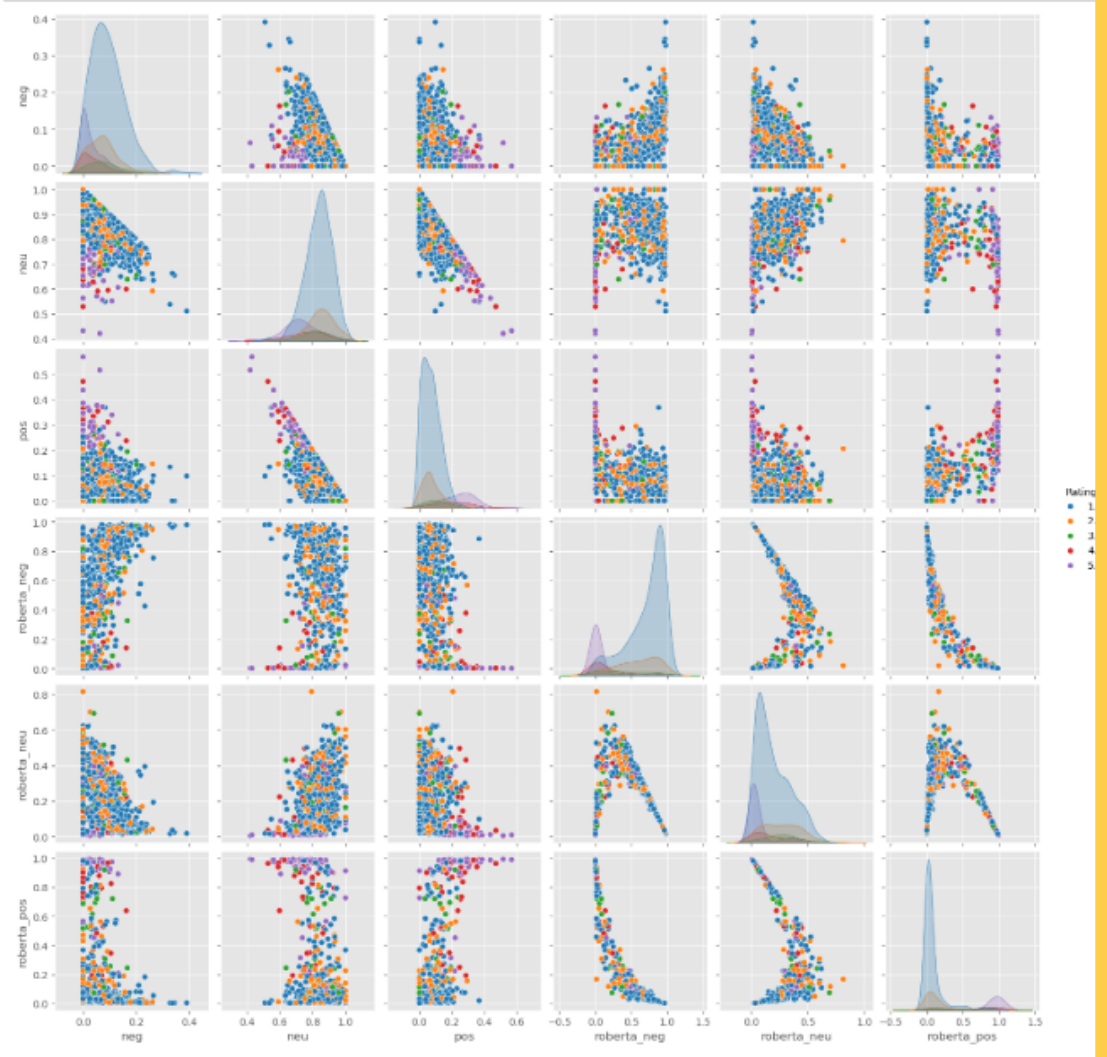
	ID	neg	neu	pos	compound	roberta_neg	roberta_neu	roberta_pos	Review	Rating	vader_compound	predicted_sentiment	true_sentiment
0	1	0.000	0.797	0.203	0.8991	0.001954	0.009608	0.988438	Amber and LaDonna at the Starbucks on Southwes...	5.0	0.8991	Positive	Positive
1	2	0.099	0.755	0.145	0.7766	0.030875	0.063873	0.905252	** at the Starbucks by the fire station on 436...	5.0	0.7766	Positive	Positive
2	3	0.087	0.767	0.145	0.5242	0.007287	0.027140	0.965573	I just wanted to go out of my way to recognize...	5.0	0.5242	Positive	Positive

6.VADER와 RoBETRa 비교 시각화

- VADER와 RoBERTa 감정 점수를 시각적으로 비교.

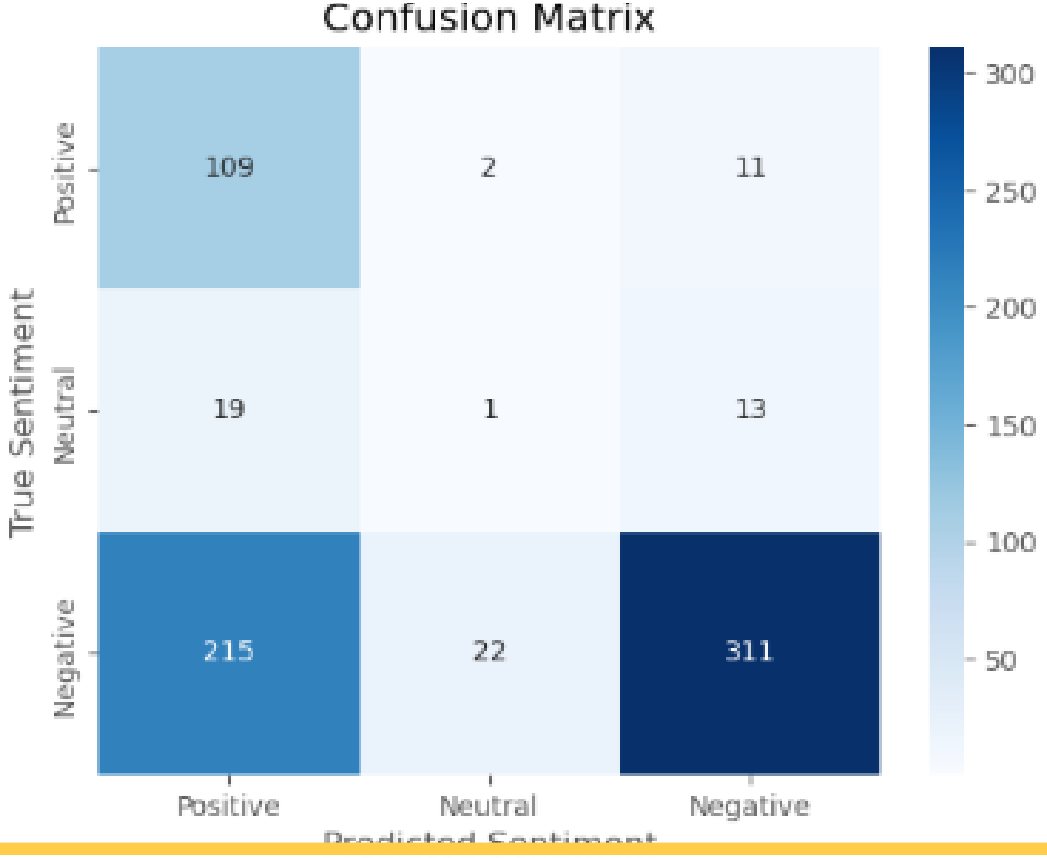
- 별점(Rating)에 따른 감정 점수 분포를 확인.

```
sns.pairplot(data=results_df, vars=[
    'neg', 'neu', 'pos', 'roberta_neg', 'roberta_neu', 'roberta_pos'],
    hue='Rating', palette='tab10')
plt.show()
```



7. 혼동행렬: 모델이 예측한 감정과 실제 감정을 비교하여 정확도 시각화.

In [46]:
```
y_true = df['true_sentiment'] # 실제 감정
y_pred = df['predicted_sentiment'] # 예측 감정
cm = confusion_matrix(y_true, y_pred, labels=['Positive', 'Neutral', 'Negative'])
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Positive', 'Neutral', 'Negative'], yticklabels=['Positive
plt.xlabel('Predicted Sentiment')
plt.ylabel('True Sentiment')
plt.title('Confusion Matrix')
plt.show()
```



8. 분류 성능

- 정밀도(Precision): 예측 중 실제로 맞춘 비율.

- 재현율(Recall): 실제 감정 중 예측이 맞은 비율.

- F1 점수: 정밀도와 재현율의 조화 평균.

In [48]:
```
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=['Positive', 'Neutral', 'Negative']))
```

```
Classification Report:
              precision    recall  f1-score   support

    Positive       0.93      0.57      0.70       548
     Neutral       0.04      0.03      0.03        33
    Negative       0.32      0.89      0.47       122

    accuracy                           0.60       703
   macro avg       0.43      0.50      0.40       703
weighted avg       0.78      0.60      0.63       703
```

“GRU를 활용한 스타벅스 감정리뷰 예측”

데이터:
https://www.kaggle.com/datasets/harshalhonde/starbucks-reviews-dataset/data
“Starbucks Reviews Dataset”

4조

1.데이터 로드

```
[ ] ##스타벅스 리뷰 데이터 불어오기
    import pandas as pd
    import numpy as np

    df = pd.read_csv('/content/drive/MyDrive/쿠글/2차 프로젝트 데이터/reviews_data.csv')

    # 리뷰 텍스트와 평점 추출
    total_data = df[['Rating', 'Review']]
    print('전체 리뷰 개수 :',len(total_data)) # 전체 리뷰 개수 출력$
```

```
전체 리뷰 개수 : 850
```

2. 데이터 정제

```
[ ] ##결측치 확인 + 결측치 제거 및 중복된 리뷰 텍스트 제거

    total_data.isna().sum()
    total_data.drop_duplicates(subset=['Review'], inplace=True) # reviews 열에서 중복인 내용이 있다면 중복 제거
    total_data.dropna(inplace=True) #null값 제거
    print('총 샘플의 수 :',len(total_data))
```

```
총 샘플의 수 : 704
```

```
[ ] #label 열 추가
    total_data['label'] = np.select([total_data.Rating > 3], [1], default=0)
    total_data[:6]
```

	Rating	Review	label
0	5.0	Amber and LaDonna at the Starbucks on Southwes...	1
1	5.0	** at the Starbucks by the fire station on 436...	1
2	5.0	I just wanted to go out of my way to recognize...	1
3	5.0	Me and my friend were at Starbucks and my card...	1
4	5.0	I'm on this kick of drinking 5 cups of warm wa...	1
5	1.0	We had to correct them on our order 3 times. T...	0

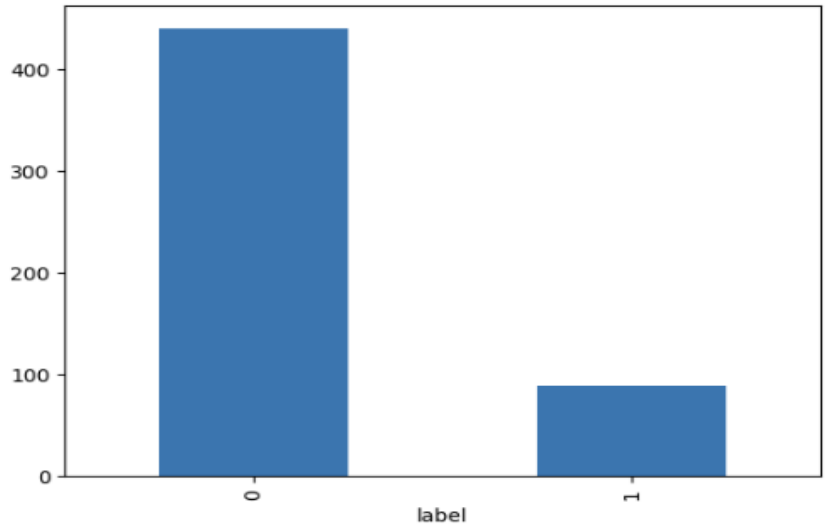
```
[ ] ##train_test data -> split
    from sklearn.model_selection import train_test_split

    train_data, test_data = train_test_split(total_data, test_size = 0.25, random_state = 42)
    print('훈련용 리뷰의 개수 :', len(train_data))
    print('테스트용 리뷰의 개수 :', len(test_data))
```

```
훈련용 리뷰의 개수 : 528
테스트용 리뷰의 개수 : 176
```

```
##train데이터 레이블 열 생성 및 분포 확인
print(train_data.groupby('label').size().reset_index(name = 'count'))
```

```
   label  count
0      0    440
1      1     88
```

```
#레이블의 분포 시각화
train_data['label'].value_counts().plot(kind = 'bar')
```




```
[ ] ##영어와 공백을 제외하고 모두 제거
    train_data['Review'] = train_data['Review'].str.replace("[^a-zA-Z ]", "", regex=True)
    train_data['Review'].replace('', inplace=True)  # 공백은 Null 값으로 변경
    print(train_data.isnull().sum())
    print('전처리 후 테스트용 샘플의 개수 :',len(train_data))

    test_data['Review'] = test_data['Review'].str.replace("[^a-zA-Z ]","", regex=True)
    test_data['Review'].replace('', inplace=True)
    print(test_data.isnull().sum())
    print('전처리 후 테스트용 샘플의 개수 :',len(test_data))
```

```
Rating    0
Review    0
label     0
dtype: int64
전처리 후 테스트용 샘플의 개수 : 528
Rating    0
Review    0
label     0
dtype: int64
전처리 후 테스트용 샘플의 개수 : 176
```

3.토큰화

```
[ ] #spacy설치
    !pip install spacy
    !python -m spacy download en_core_web_sm
```

spaCy: 자연어 처리를 위한 Python 기반 라이브러리로, 빠르고 효율적인 NLP 작업을 지원(영어 형태소 분석기)

```
[ ] import spacy
    nlp = spacy.load("en_core_web_sm")

    # spaCy 기반 토큰화 함수 정의
    def spacy_tokenize(text):
        if isinstance(text, str):  # 입력이 문자열인 경우에만 처리
            doc = nlp(text)  # 텍스트 분석
            tokens = [token.text for token in doc]
            return tokens
        else:
            return []  # 문자열이 아니면 빈 리스트 반환

    stopwords = ['the', 'i', 'to', 'and', 'a', 'my', 'starbucks', 'of', 'was', 'in', 'it', 'for', 'that', 'is', 'they', 'me', 'at', 'on', 'have', 'coffee',
    'this', 'with', 'but', 'you', 'she', 'are', 'so', 'had', 'be',
    'when', 'their', 'customer', 'get', 'one', 'there', 'drink',
    'as', 'store', 'service', 'we', 'time', 'from', 'all', 'or', 'about', 'he']

    train_data['Review'] = train_data['Review'].fillna('')
    test_data['Review'] = test_data['Review'].fillna('')
    train_data['tokenized'] = train_data['Review'].apply(spacy_tokenize)
    train_data['tokenized'] = train_data['tokenized'].apply(lambda x: [item.lower() for item in x if item.lower() not in stopwords])
    test_data['tokenized'] = test_data['Review'].apply(spacy_tokenize)
    test_data['tokenized'] = test_data['tokenized'].apply(lambda x: [item.lower() for item in x if item.lower() not in stopwords])
```

```
[ ] #spaCy -> X_train, X_test, y_train, y_test
    X_train = train_data['tokenized'].values #학습용 리뷰 데이터(토큰화된 텍스트).
    y_train = train_data['label'].values #학습용 레이블(정답값).
    X_test= test_data['tokenized'].values #테스트용 리뷰 데이터(토큰화된 텍스트).
    y_test = test_data['label'].value #테스트용 레이블(정답값).
```

4.정수 인코딩

```
[ ] ## 4) 정수 인코딩
    from tensorflow.keras.preprocessing.text import Tokenizer

    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(X_train)

    threshold = 2
    total_cnt = len(tokenizer.word_index) # 단어의 수
    rare_cnt = 0 # 등장 빈도수가 threshold보다 작은 단어의 개수를 카운트
    total_freq = 0 # 훈련 데이터의 전체 단어 빈도수 총 합
    rare_freq = 0 # 등장 빈도수가 threshold보다 작은 단어의 등장 빈도수의 총 합

    # 단어와 빈도수의 쌍(pair)을 key와 value로 받는다.
    for key, value in tokenizer.word_counts.items():
        total_freq = total_freq + value

        # 단어의 등장 빈도수가 threshold보다 작으면
        if(value < threshold):
            rare_cnt = rare_cnt + 1
            rare_freq = rare_freq + value

    print('단어 집합(vocabulary)의 크기 :',total_cnt)
    print('등장 빈도가 %s번 이하인 희귀 단어의 수: %s'%(threshold - 1, rare_cnt))
    print("단어 집합에서 희귀 단어의 비율:", (rare_cnt / total_cnt)*100)
    print("전체 등장 빈도에서 희귀 단어 등장 빈도 비율:", (rare_freq / total_freq)*100)
```

```
단어 집합(vocabulary)의 크기 : 4542
등장 빈도가 1번 이하인 희귀 단어의 수: 2260
단어 집합에서 희귀 단어의 비율: 49.75781594011448
전체 등장 빈도에서 희귀 단어 등장 빈도 비율: 8.125988781820796
```

Tokenizer: 텍스트 데이터를 머신러닝 모델에 입력하기 전에 숫자로 변환하거나, 단어/문장을 나누는 작업(토큰화)을 수행하는 도구

```
[ ] # 전체 단어 개수 중 빈도수 2이하인 단어 개수는 제거.
    # 0번 패딩 토큰과 1번 OOV 토큰을 고려하여 +2
    vocab_size = total_cnt - rare_cnt + 2
    print('단어 집합의 크기 :',vocab_size)
```

```
단어 집합의 크기 : 2284
```

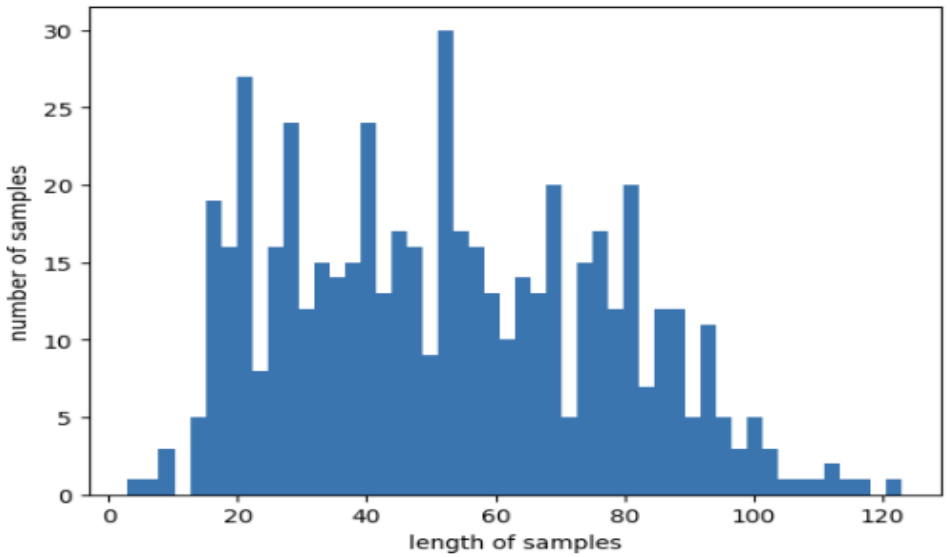
```
[ ] tokenizer = Tokenizer(vocab_size, oov_token = 'OOV')
    tokenizer.fit_on_texts(X_train)
    X_train = tokenizer.texts_to_sequences(X_train)
    X_test = tokenizer.texts_to_sequences(X_test)
```

5.패딩

pad_sequences: TensorFlow/Keras의 텍스트 전처리 도구로, 길이가 다른 시퀀스(sequence)를 동일한 길이로 맞춰주는 데 사용됩니다. 짧은 시퀀스는 **패딩(padding)**을 추가해 길이를 늘림. 긴 시퀀스는 잘라냄(truncating).

```
from matplotlib import pyplot as plt
from tensorflow.keras.preprocessing.sequence import pad_sequences

print('리뷰의 최대 길이 :',max(len(review) for review in X_train))
print('리뷰의 평균 길이 :',sum(map(len, X_train))/len(X_train))
plt.hist([len(review) for review in X_train], bins=50)
plt.xlabel('length of samples')
plt.ylabel('number of samples')
plt.show()
```

```
리뷰의 최대 길이 : 123
리뷰의 평균 길이 : 52.67424242424242
```




```
[ ] #일정 길이 이하 샘플 비율 함수 정의
    def below_threshold_len(max_len, nested_list):
      count = 0
      for sentence in nested_list:
        if(len(sentence) <= max_len):
            count = count + 1
      print('전체 샘플 중 길이가 %s 이하인 샘플의 비율: %s'%(max_len, (count / len(nested_list))*100))
```

```
[ ] max_len = 80
    below_threshold_len(max_len, X_train)

    X_train = pad_sequences(X_train, maxlen=max_len)
    X_test = pad_sequences(X_test, maxlen=max_len)
```

```
전체 샘플 중 길이가 80 이하인 샘플의 비율: 84.0909090909091
```

길이가 **max_len보다 짧은 시퀀스**: 앞부분에 0을 추가(패딩).
길이가 **max_len보다 긴 시퀀스**: 앞부분을 잘라내어 고정 길이로 만듦.

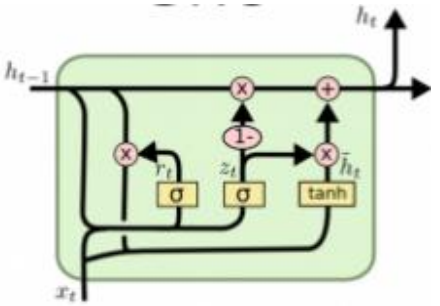
6. GRU로 스타벅스 감정리뷰 예측

```
# TensorFlow/Keras에서 필요한 모듈 임포트
from tensorflow.keras.layers import Embedding, Dense, GRU  # Embedding, Dense, GRU 레이어
from tensorflow.keras.models import Sequential  # Sequential 모델
from tensorflow.keras.models import load_model  # 모델 저장 및 로드
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint  # 콜백 설정 (조기 종료, 모델 체크포인트)

# 임베딩 차원과 GRU의 은닉 상태 크기 정의
embedding_dim = 100  # 임베딩 벡터의 차원 (단어를 100차원 벡터로 변환)
hidden_units = 128  # GRU 레이어의 은닉 상태 크기

# Sequential 모델 생성
model = Sequential()

# Embedding 레이어 추가
# - vocab_size: 전체 단어 집합의 크기 (단어 인덱스의 개수)
# - embedding_dim: 단어를 변환할 임베딩 벡터의 차원
model.add(Embedding(vocab_size, embedding_dim))

# GRU 레이어 추가
# - hidden_units: GRU의 은닉 상태 크기
model.add(GRU(hidden_units))

# Dense 레이어 추가
# - 출력 노드: 1개 (이진 분류)
# - 활성화 함수: 'sigmoid' (출력값을 0~1 사이로 변환)
model.add(Dense(1, activation='sigmoid'))

# EarlyStopping 콜백 설정
# - val_loss(검증 손실)을 모니터링
# - 연속으로 4번(val_loss 개선 없음) 동안 학습 종료
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)

# ModelCheckpoint 콜백 설정
# - val_acc(검증 정확도)를 모니터링
# - 가장 높은 val_acc를 가진 모델을 'best_model.keras' 파일로 저장
mc = ModelCheckpoint('best_model.keras', monitor='val_acc', mode='max', verbose=1, save_best_only=True)

# 모델 컴파일
# - optimizer: 'rmsprop' (RMSProp 최적화 알고리즘)
# - loss: 'binary_crossentropy' (이진 분류를 위한 손실 함수)
# - metrics: 'acc' (정확도를 모니터링)
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])

# 모델 학습
# - epochs: 15번 반복 학습
# - batch_size: 미니 배치 크기 (64)
# - validation_split: 학습 데이터의 20%를 검증 데이터로 사용
# - callbacks: EarlyStopping과 ModelCheckpoint 콜백을 사용
history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=64, validation_split=0.2)
```

```
Epoch 10: val_acc did not improve from 0.89623
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s
Epoch 10: early stopping

164ms/step - acc: 0.9960 - loss: 0.0232 - val_acc: 0.8962 - val_loss: 0.4425
```

<Embedding 레이어>
단어를 고차원 벡터로 변환.
입력 크기: vocab_size (전체 단어 집합의 크기).
출력 크기: embedding_dim (단어 벡터의 차원).

<GRU 레이어>
순환 신경망(RNN)의 변형으로, LSTM과 비슷한 기능 수행하지만, 더 간단한 구조.
2개(업데이트,리셋 게이트)게이트를 사용.
간단한 데이터에 적합.



<Dense 레이어>
출력 노드가 1개이며, sigmoid 활성화 함수를 통해 출력값을 0~1 사이로 제한.

<RMSProp>
딥러닝에서 사용되는 경사 하강법(SGD)의 업그레이드 버전

```
[ ] #모델 학습 후 저장
    model.save('best_model.keras')
    loaded_model = load_model('best_model.keras')
    print("\n 테스트 정확도: %.4f" % (loaded_model.evaluate(X_test, y_test)[1]))
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - acc: 0.8063 - loss: 0.6933

테스트 정확도: 0.8295
```

7. 저장된 모델로 리뷰 예측하기

```python
#저장된 모델로 리뷰예측 함수 정의
import re

def sentiment_predict(new_sentence, tokenizer, model, max_len=7000):
    new_sentence = re.sub(r'[^a-zA-Z ]', '', new_sentence)
    new_sentence = spacy_tokenize(new_sentence)
    new_sentence = [word for word in new_sentence if not word in stopwords]
    sequence = tokenizer.texts_to_sequences([new_sentence])
    padded_sequence = pad_sequences(sequence, maxlen=max_len)
    prediction = model.predict(padded_sequence)
    sentiment = "Positive" if prediction > 0.5 else "Negative"
    sentiment_score = prediction[0][0]
    return sentiment, sentiment_score
```

```python
#스타벅스 데이터 중 무작위로 6개 리뷰 추출
import random

random.seed(12)

random_reviews = random.sample(list(total_data[['Review', 'label']].values), 6)

for review in random_reviews:
  print('review:',review)
```

```
review: ['Know I frequent 3 different Starbucks in my area, the closest one does not have a drive-thro
 0]
review: ['The customer service is also great. Will make the coffee as you wish. Even willing to help y
 1]
review: ["I work as a Starbucks, Safeway employee; and I have recently gotten the job. There are no be
 0]
review: ["December 3rd 2014 at 12:49:35 pm, I went to my usual Starbucks and got my usual 2 Venti suga
 0]
review: ["I'm currently a college student and I make it a routine to visit Starbucks whenever I have a
 0]
review: ['Better coffee made at home. Unknown why people pay high prices for high fat drinks with poor
 0]
```

```python
#함수를 사용하여 리뷰 긍정/부정 예측
for review in random_reviews:
  sentiment, sentiment_score = sentiment_predict(review[0], tokenizer, model)
  print(f"Sentiment: {sentiment}")
  print(f"Sentiment Score: {sentiment_score:.4f}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Sentiment: Negative
Sentiment Score: 0.0001
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
Sentiment: Positive
Sentiment Score: 0.9130
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 730ms/step
Sentiment: Negative
Sentiment Score: 0.0002
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 787ms/step
Sentiment: Negative
Sentiment Score: 0.0001
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 726ms/step
Sentiment: Negative
Sentiment Score: 0.0011
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 725ms/step
Sentiment: Negative
Sentiment Score: 0.0071
```

참고 문헌

- https://wikidocs.net/94600

딥 러닝을 이용한 자연어 처리 입문

->10. RNN을 이용한 텍스트 분류(Text Classification)

->10-07 네이버 쇼핑 리뷰 감성 분류하기(Naver Shopping Review Sentiment Analysis)