1. 개요

본 프로젝트에서는 고객 리뷰 데이터를 통해 내용이 긍정(부정)인지 파악하는 것 뿐만 아니라, 어떤 요소에 대한 리뷰인지 파악하는 Multi-output 문제를 주제로 선정했다. 리뷰 내용에 대해 감성 분석(Sentiment Analysis)만 하고 그치는 것이 아니라, 제품의 '품질'에 대한 리뷰인지, '디자인'에 대한 리뷰인지 등, 즉 어떤 항목에 대한 리뷰인지도 함께 파악하고자 하였다. 항목을 파악하는 문제와 관련하여, 문장에서 중요 요소를 추출(Aspect Extraction)하는 비지도 학습 방법도 존재하였지만, 깔끔하지 않은 실제 고객 리뷰 데이터에서 비지도 학습을 통해 좋은 성능을 기대하기는 어려울 것으로 판단했다. 따라서, 리뷰 내용들을 참고하여, 4가지 항목(품질/성능, 가격/혜택, 배송/서비스, 디자인/외관)으로 항목 범주를 선택했고, 지도 학습 방법을 이용하고자 하였다.

그런데, 하나의 리뷰가 오직 하나의 항목에 대한 내용만을 담고 있으리라는 법이 없었다. 어떤 리뷰는 품질, 배송, 디자인에 대한 내용을 모두 내포하기도 하였다. 즉, 리뷰마다 그에 해당하는 항목 개수가 다르기 때문에, Multi-label 분류 문제로 간주할 필요가 있었다. 이는, 복수의 클래스 중 하나의 클래스로 분류하는 Multi-class 분류 문제와는 구별된다. Multi-class 분류 문제에서는 각 클래스가 상호 배타적인 관계이지만, Multi-label에서는 더 이상 상호 배타적인 관계가 아니게 된다. 3년 전 캐글(Kaggle)에서 열린 악성 댓글 분류 대회[1]가 바로 Multi-label 분류 문제였다. 특정 악성 댓글에 대해, 외설적인 내용이 포함되면 1 아니면 0, 욕이 포함되면 1 아니면 0 등, 분류해야 할 타겟이 복수인 대회였다. 따라서, 해당 프로젝트에서도, 감정, 품질/성능, 가격/혜택, 배송/서비스, 디자인/외관 총 5가지 경우에 대해, 각각 이항 분류하는 Multi-label 문제를 다루고자 하였다.

데이터로는, 우선 긍정/부정 분류가 되어 있는 네이버 쇼핑 고객 리뷰 데이터[2]를 얻었다.

	rating	text	label
0	5	배공빠르고 굿	1
1	2	택배가 엉망이네용 저희집 밑에층에 말도없이 놔두고가고	0
2	5	아주좋아요 바지 정말 좋아서2개 더 구매했어요 이가격에 대박입니다. 바느질이 조금 ...	1
3	2	선물용으로 빨리 받아서 전달했어야 하는 상품이었는데 머그컵만 와서 당황했습니다. 전...	0
4	5	민트색상 예뻐요. 옆 손잡이는 거는 용도로도 사용되네요 ㅎㅎ	1

이에 추가적으로, 4가지 항목들에 대해 라벨링을 진행해야 했다. 다음 두번째 절에서 데이터 구축 및 탐색에 대한 내용을 언급하고, 세번째 절에서 모델링에 대한 내용을 다루고자 한다. 마지막 네번째 절에서 모델의 결과를 해석하고 한계점을 언급하며 프로젝트 보고서를 마치고자 한다.

[1] Toxic Comment Classification Challenge, https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview

[2] https://github.com/bab2min/corpus/tree/master/sentiment

2. 데이터 구축 및 전처리

2.1. 데이터 구축

우선 학습용 데이터 구축을 위해, 감정, 품질/성능, 가격/혜택, 배송/서비스, 디자인/외관의 4가지 항목에 대해, 해당 항목에 대한 내용이 포함되어 있는지 여부를 라벨링해야 했다. 20만 개에 해당하는 모든 리뷰 내용을 직접 수기로 라벨링 하기에는 시간이 부족했기 때문에, 다음의 과정들을 통해 효율적으로 라벨링을 진행하고자 하였다. 먼저, 띄어쓰기 및 토큰화 등의 전처리를 진행한 후, 주요 키워드 포함 여부로 1차 라벨링을 진행하였다. 이후, 항목들 각각 하나씩만 라벨링된 리뷰들로 간단한 모델링을 통해 2차 라벨링을 진행하였다. 최종적으로 대략적으로 검토하는 시간을 가진 후, 데이터 구축을 완료했다.

	text	sentiment	quality/performance	price/event	delivery/service	design/appearance
0	배공빠르고 굿	1	0	0	1	0
1	택배가 엉망이네용 저희집 밑에층에 말도없이 놔두고가고	0	0	0	1	0
2	아주좋아요 바지 정말 좋아서개 더 구매했어요 이가격에 대박입니다 바느질이 조금 영성...	1	1	1	0	0
3	선물용으로 빨리 받아서 전달했어야 하는 상품이었는데 머그컵만 와서 당황했습니다 전화...	0	0	0	1	0
4	민트색상 예뻐요 옆 손잡이는 거는 용도로도 사용되네요 ㅎㅎ	1	0	0	0	1
5	비추합니다 계란 뒤집을 때 완전 불편해요 ㅠㅠ 코팅도 묻어나고 보기엔 예쁘고 실용적...	0	1	0	0	1
6	주문을 월에 시켰는데 월일에 배송이 왔네요 ㅎㅎㅎ 여기 회사측과는 전화도 안되고 아...	0	0	0	1	0
7	넉넉한 길이로 주문했는데도 안 맞네요 별로예요	0	0	0	0	1
8	보풀이 계속 때처럼 나오다가 지금은 안나네요	0	1	0	0	0
9	인데 전문속옷브랜드 위생팬티보다 작은듯해요 불편해요 밴딩부분이 다 신축성없는 일반실...	0	1	0	0	1

아래 wordcloud에서 각 항목들을 대표하는 주요 키워드들을 한 눈에 확인할 수 있다.

<quality/performance>

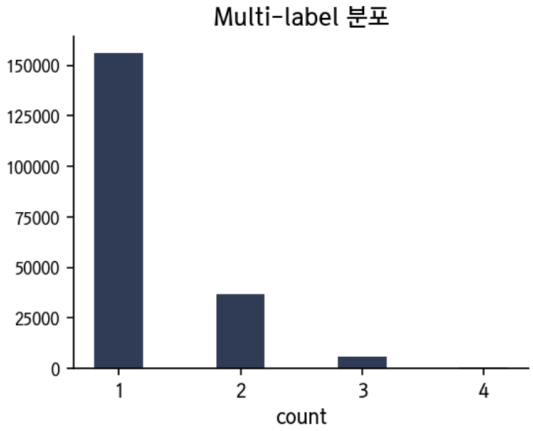
<price/event>

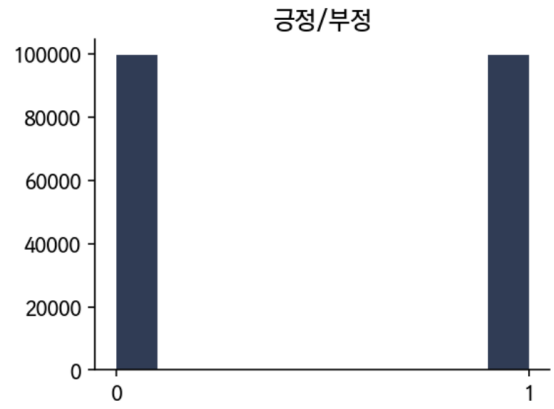
<delivery/service>

<size/appearance>

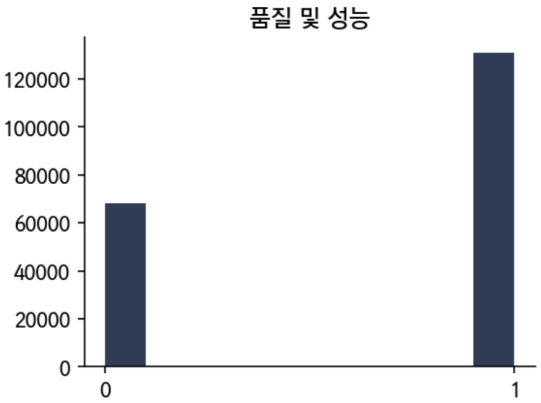
한편, 아래 그림의 왼쪽 <Multi-label 분포>에서 볼 수 있듯이, 샘플별 sentiment를 제외한 label 개수를 파악한 결과, 하나의 항목에 대한 내용만 포함하는 리뷰가 약 75%에 해당하고, 두개의 항목에 해당하는 리뷰가 약 20%에 해당했다.

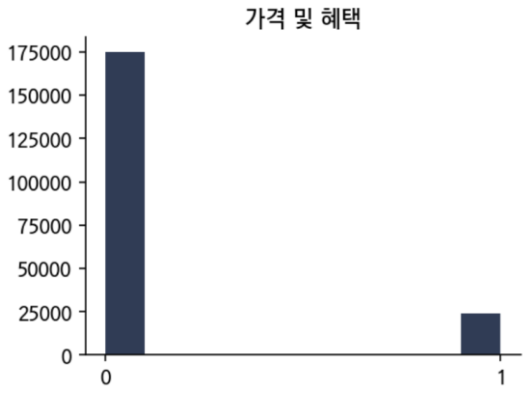
이후, 각 label별 분포를 확인한다. 먼저, 아래 오른쪽 그림인 sentiment에 대한 분포를 보면, 긍정/부정에 대한 리뷰 개수가 거의 동일할 정도로 balanced하다.

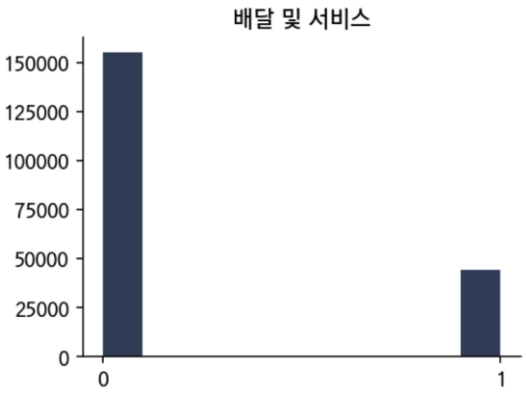


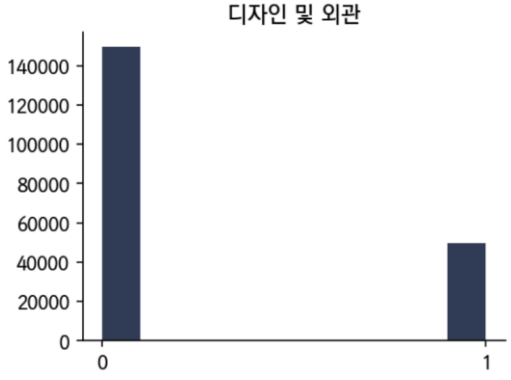


한편, 각 리뷰 항목에 대한 분포를 확인해본 결과, 불균형한 형태를 띄는 것을 볼 수 있다. 사실상 과반수의 리뷰가 제품 자체의 품질 및 성능에 대한 내용을 언급하고 있는 경우가 많고, 그 외의 항목들에 대해서는 언급하고 있지 않는 경우가 많았다. 특히, 가격 및 혜택에 대한 내용은 언급되지 않은 리뷰가 전체의 약 80%에 해당했다. 따라서, 이렇게 label 별로 불균형한 형태를 띄고 있기 때문에, 모델링에서 지표는 'accuracy'보다는 'f1 score'를 이용할 것으로 결정했다.









2.2. 전처리

한국어 토큰화 패키지로는 customized KoNLPy[3]를 이용했다. Customized KoNLPy에서는 일부 토큰들을 지정할 수 있는 기능이 존재하는데, 이는 빈번하게 사용되는 유행어 등을 지정할 때 주로 사용된다. 예를 들어, '가성비'라는 단어는 '가격 대비 성능'을 의미하는데, 쇼핑 리뷰에서 빈번하게 등장하는 단어이다. 그런데, 일반 KoNLPy 품사 태깅으로 토큰화를 진행할 경우 '가성비'를 '가', '성비'로 분리하는 모습을 보였다. 또는, 정규화(용언을 '-이다' 형태로 바꾸는 과정) 설정까지 하게 되면, '재구매'라는 단어를 '자다', '구매'로 바꾸기도 하였다. 즉, '재'라는 단어를 용언으로 받아들여 '자다'로 바꾸는 것이었다. 워낙 쇼핑 리뷰에서 빈번하게 등장하는 단어였기 때문에, 이에 대한 조치를 취할 필요가 있었다.

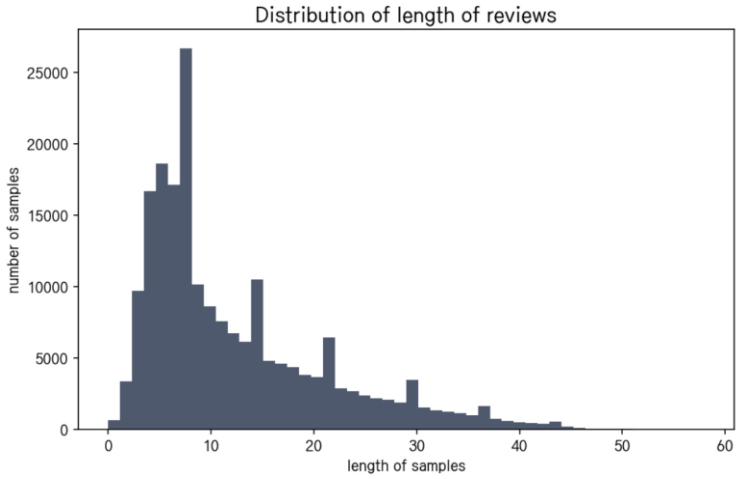
쇼핑 리뷰에서 빈번하게 등장할 것으로 추측되지만 잘 못 토큰화될 가능성이 높은 일부 단어들을 customized KoNLPy를 통해 토큰으로 지정하였다. 아래와 같이, '강추', '비추', '가성비', '재구매', '핏' 등 일부 단어들을 토큰으로 지정한 후, 품사 태깅을 진행하였다.

```
from ckonlpy.tag import Twitter
twi = Twitter()

words = [('강추','Noun'), ('비추','Noun'), ('가성비','Noun'),
         ('재구매','Noun'), ('엉성', 'Noun'), ('핏', 'Noun'), ('타이트','Noun')]

for word in words:
    name, poomsa = word
    twi.add_dictionary(name, poomsa)
```

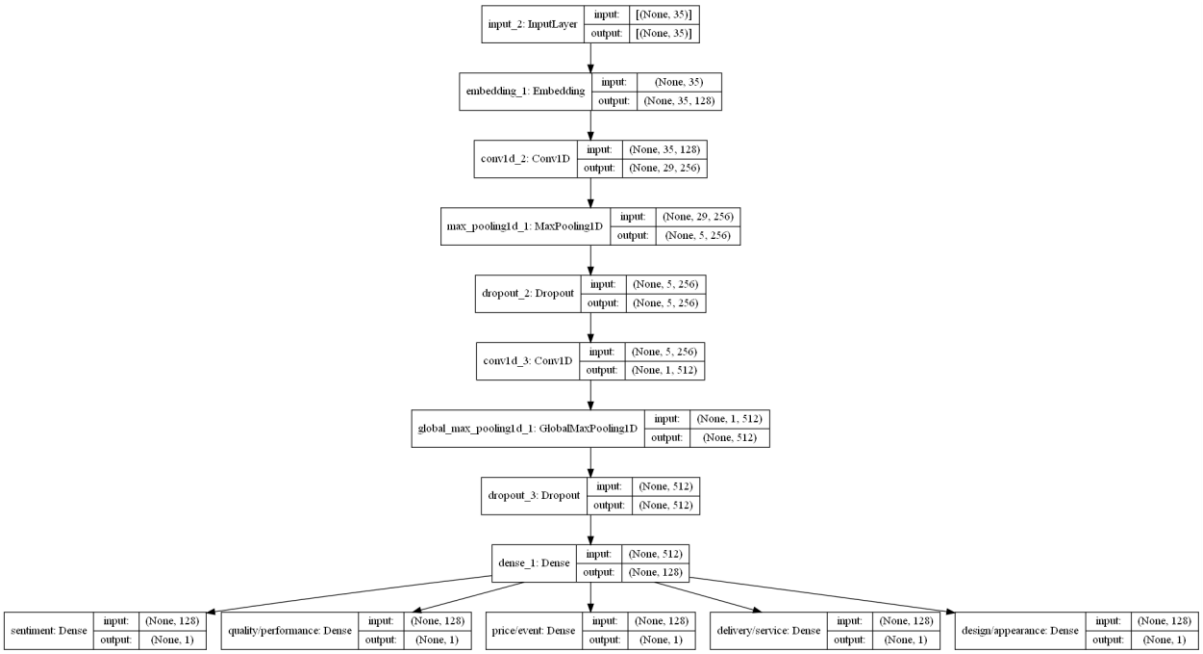
토큰화 진행 후에는, 전체 데이터에서 등장 빈도가 2번 이하인 토큰들을 제거하였고, 따라서 남은 토큰 집합 크기는 17328개가 되었다. 그 후, 각 샘플의 문장 길이들의 분포를 확인하였다. 그 결과, 전체 샘플 중 길이가 35이하인 샘플이 약 97%에 해당했고, 이에 따라 샘플 최대 길이를 35로 두어 패딩을 진행하였다.



[3] Customized KoNLPy, https://github.com/lovit/customized_konlpy

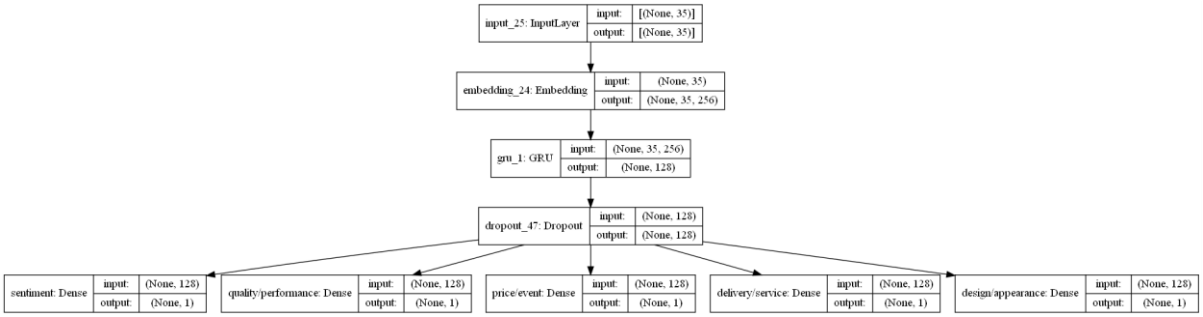
3. 모델링

시도해본 아키텍처는 크게 4가지인데, 각각 **(1)1D Convolution, (2)GRU, (3)Bidirectional GRU, (4)Bidirectional LSTM** 레이어를 주로 사용한 아키텍처이다. padding된 token index 데이터를 input으로 하고, Sentiment, Quality/Performance, Price/Event, Delivery/Service, Design/Appearance에 대해 각각 이항 분류하는 multi-output 구조이다. 아래 플롯은 1D convolution layer를 이용한 아키텍처다.



〈fig 3-1. 1D convolution을 이용한 아키텍처〉

다음은 GRU 레이어를 이용한 아키텍처이다. BiGRU, BiLSTM 이용한 아키텍처 모두 아래 구조에서 GRU레이어 부분만 바꾼 것이므로, 구조 플롯은 생략한다.



〈fig 3-2. GRU를 이용한 아키텍처〉

각각의 출력층 손실함수는 binary_crossentropy로 하였고, optimizer는 'rmsprop'으로 하였다. 지표는 'F1 score'로 하였는데, 내장되어 있는 것이 없었기 때문에 직접 정의해야 했다.

```
def Model3():
    input_layer = Input(shape=(maxlen,))
    embedding_text = Embedding(vocab_size, 256)(input_layer)
    x = Bidirectional(GRU(128))(embedding_text)
    x = Dropout(0.5)(x)
    x = Dense(64, activation="relu")(x)

    out_list = []
    for i in range(y.shape[1]):
        out_list.append(Dense(1, activation="sigmoid", name=label_names[i])(x))

    model = Model(inputs=input_layer, outputs=out_list)
    model.compile(loss=['binary_crossentropy']*n_label, optimizer='rmsprop', metrics=[F1score])

    return model
```

<fig 3-5. Model 코드>

```
import tensorflow.keras.backend as K

def F1score(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    recall = true_positives / (possible_positives + K.epsilon())
    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val
```

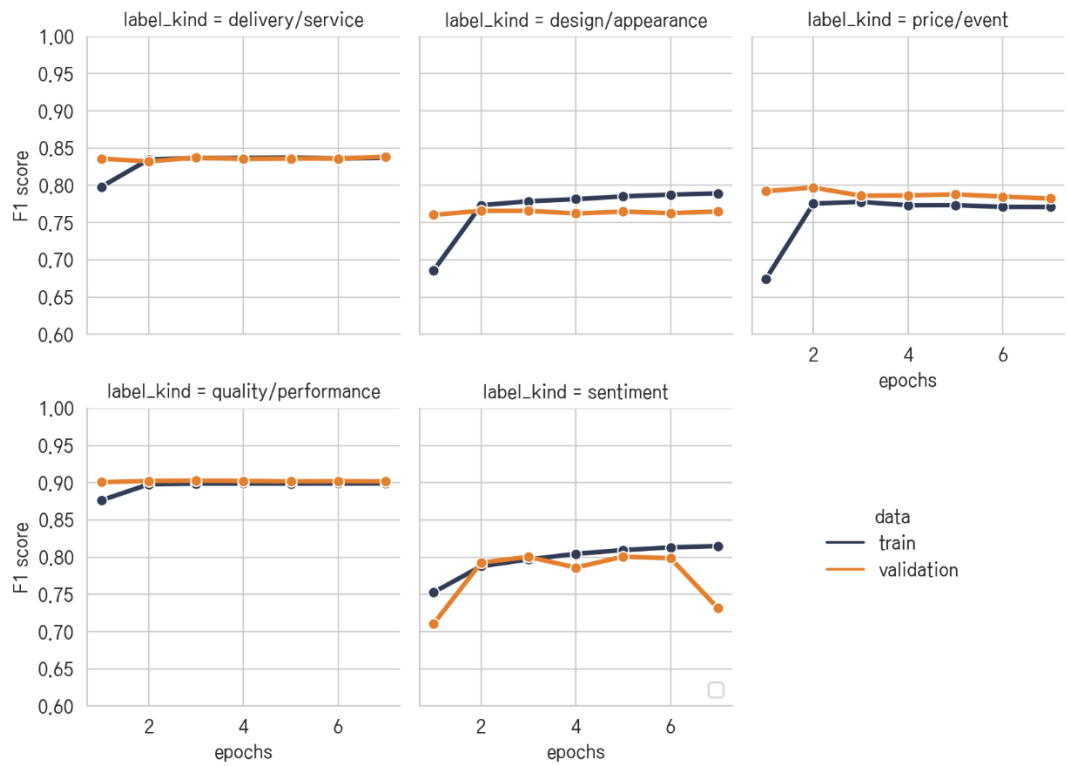
<fig 3-6. F1 score 정의>

이후, 다음과 같이 모델을 적합하였다. 전체 에폭은 20번이고, 4번 이상 validation loss가 감소하지 않을 경우 학습을 중단하도록 했다. Validation loss가 가장 적을 때의 모델을 저장하였다.

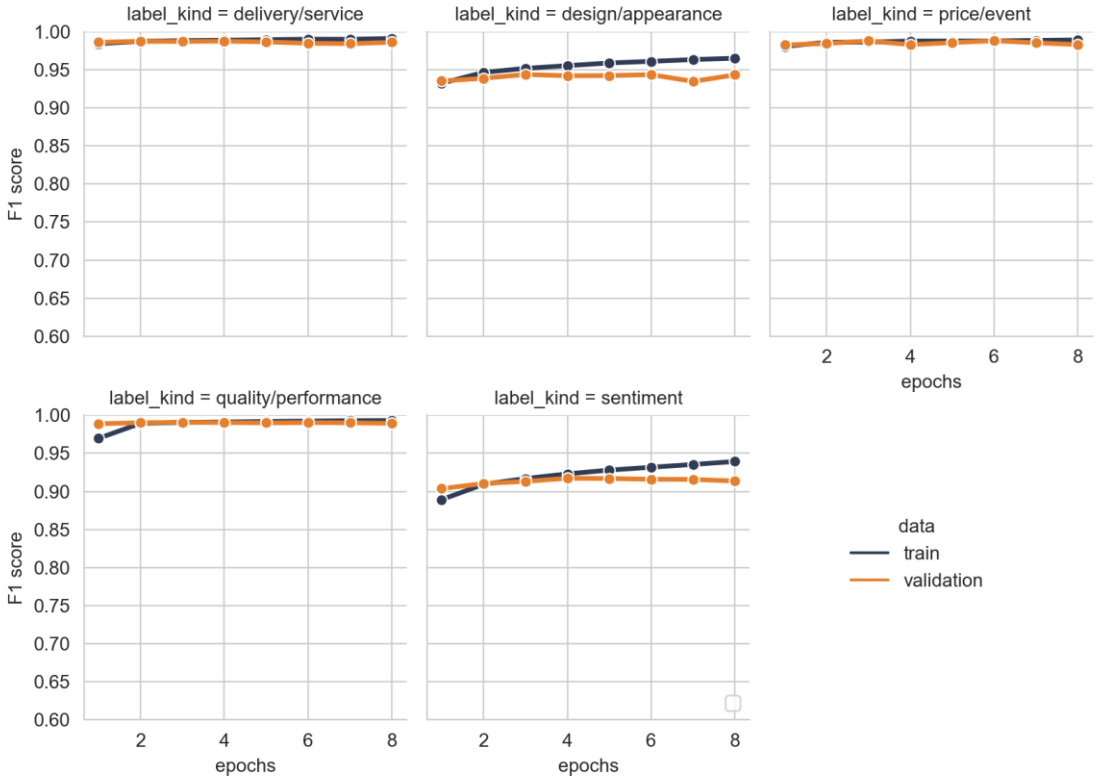
```
callback_list = [EarlyStopping(monitor='val_loss', patience=4),
                 ModelCheckpoint(filepath='model3.h5', monitor='val_loss', save_best_only=True)]
history3 = model3.fit(X_train, y_train_list, epochs=20, batch_size=60, validation_split=0.1, callbacks=callback_list)
```

```
Epoch 1/20
2392/2392 [==============================] - 188s 79ms/step - loss: 0.6965 - sentiment_loss: 0.3050 - quality/performance_loss: 0.1492
- price/event_loss: 0.0458 - delivery/service_loss: 0.0542 - design/appearance_loss: 0.1423 - sentiment_F1score: 0.8797 - quality/perf
ormance_F1score: 0.9583 - price/event_F1score: 0.9217 - delivery/service_F1score: 0.9543 - design/appearance_F1score: 0.8765 - val_los
s: 0.4615 - val_sentiment_loss: 0.2571 - val_quality/performance_loss: 0.0642 - val_price/event_loss: 0.0207 - val_delivery/service_lo
ss: 0.0301 - val_design/appearance_loss: 0.0894 - val_sentiment_F1score: 0.9024 - val_quality/performance_F1score: 0.9892 - val_price/
event_F1score: 0.9852 - val_delivery/service_F1score: 0.9859 - val_design/appearance_F1score: 0.9364
```

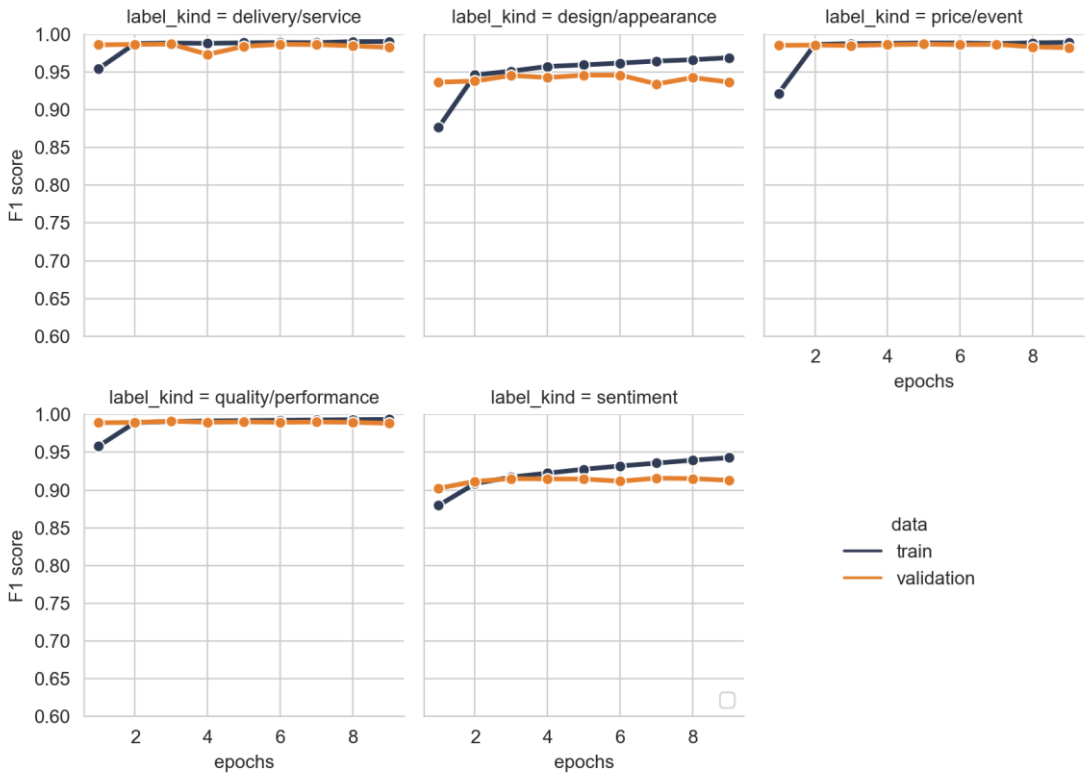
아키텍처 별로 진행된 에폭동안 train/validation set에서의 F1score를 그림으로 나타내면 다음과 같다.



<fig 3-7. Conv1D 이용한 아키텍처 적합 결과>



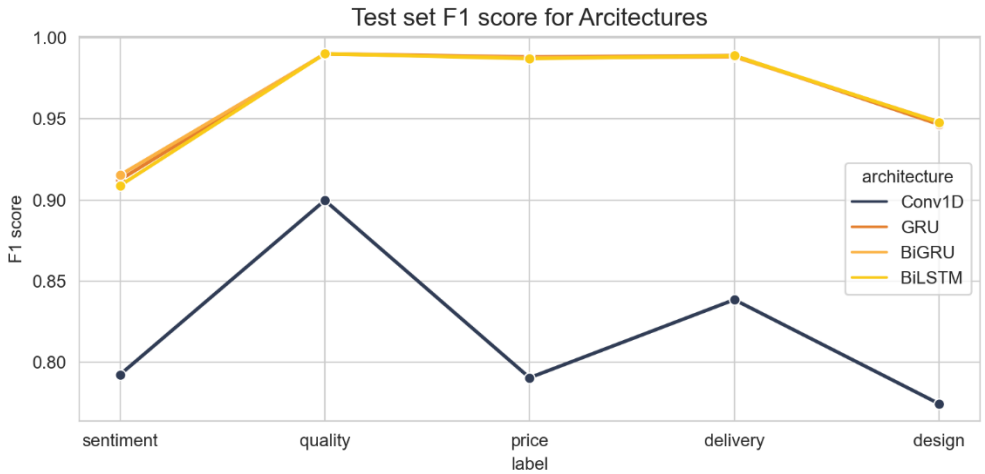
<fig 3-8. GRU 이용한 아키텍처 적합 결과>



<fig 3-9. BiGRU 이용한 아키텍처 적합 결과>

BiLSTM을 이용한 아키텍처는 GRU/BiGRU 이용한 아키텍처의 결과와 매우 유사하였기 때문에 그림은 생략하였다. 전반적으로, Convolution 1D 레이어를 제외한, 나머지 RNN 계열의 아키텍처는 각 target에 대해 높은 성능을 보이며 유사한 양상을 띄었다. Sentiment의 validation F1 score가 약 0.92 대로 가장 낮은 모습을 보였고, 나머지 항목에 대해서는 약 0.95 ~ 0.99 정도의 F1 score를 기록했다.

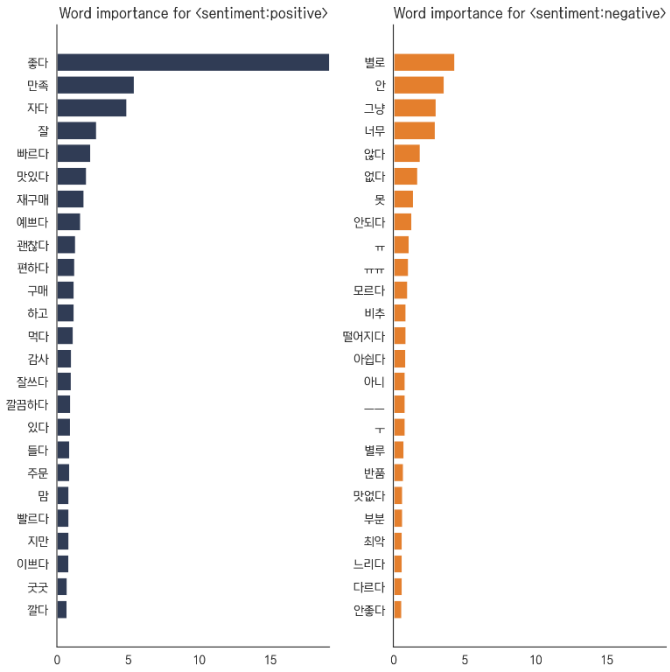
각각 best model로 Target들에 대해 Test F1score를 그림으로 그려보면 다음과 같다. RNN 계열의 아키텍처들은 거의 동일하게 높은 성능을 보였다. 반면, 해당 텍스트 데이터에서는, Convolution을 이용한 아키텍처가 매우 저조한 퍼포먼스를 보였다.



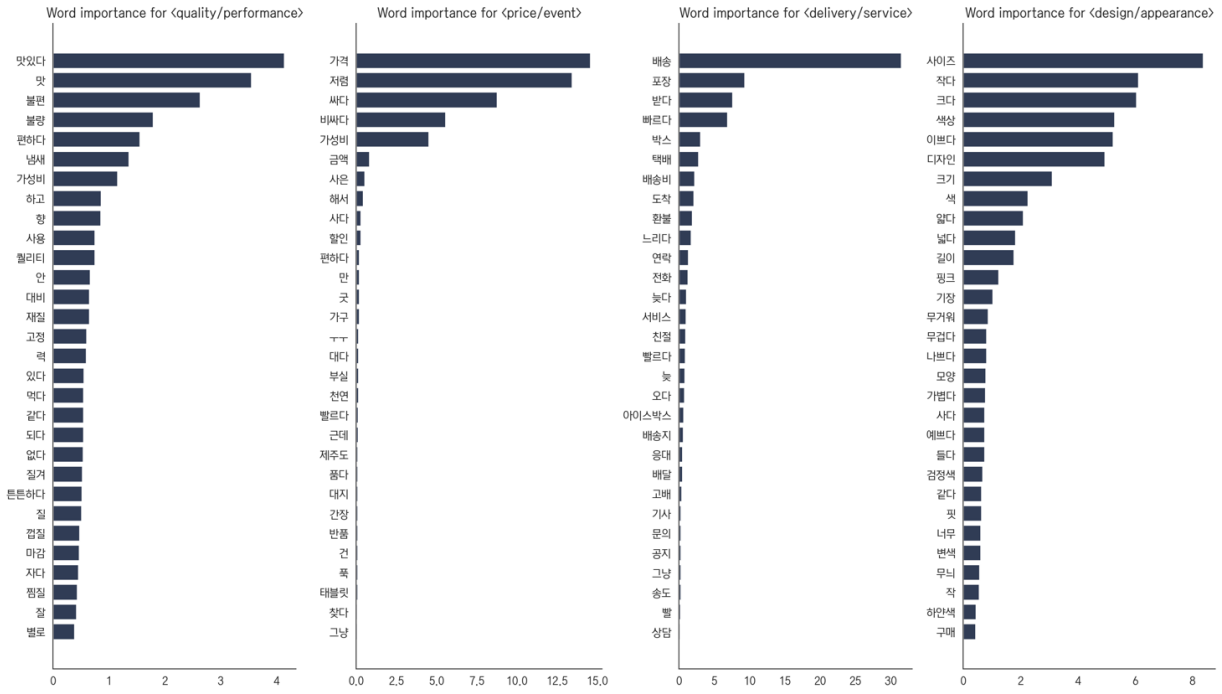
4. 결론

4.1. 해석

학습한 두번째 모델에 대해 shap 라이브러리를 이용하여 각각의 target에 대해 중요한 역할을 한 단어를 파악하고자 하였다. 그 이유로는, 딥러닝 모델을 학습한 후에 변수 중요도까지 확인할 수 있다면, 모델에 대한 신뢰도를 더욱 얻을 수 있을 뿐 아니라, 인사이트 또한 얻을 수 있을 것이라 생각했기 때문이다. 평상시 shap을 이용하면, 변수 중요도를 바로 파악할 수 있었지만, 해당 텍스트 분석에서는 바로 파악할 수는 없었다. 따라서, 각각의 자리(feature)에 대한 shap value를 뽑은 뒤, 그 자리의 인덱스 값에 해당하는 단어에 대해 value들을 더하는 과정을 구상하여 계산했다. 먼저, sentiment에 대해 단어 중요도를 파악한 결과 오른쪽 플롯과 같다. 왼쪽은 긍정 리뷰, 오른쪽은 부정 리뷰에 대한 단어 중요도인데, 예상했던 결과와 비슷하다. 즉, 모델이 의도에 맞게 잘 학습되었다고 볼 수 있다.

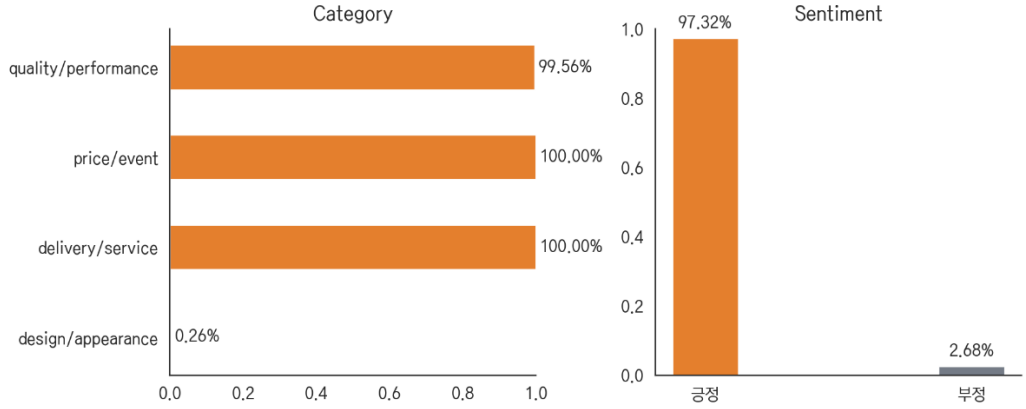


마찬가지로, 아래 플롯들은 각 항목들에 대한 단어 중요도이다. 해당 쇼핑 데이터에서는, 식품, 의류, 기기 등 다양한 종류의 상품이 있었기 때문에, <품질/성능> 카테고리에서는 '맛', '불편', '불량' 등의 다양한 특성의 단어가 존재하며, 값들이 비교적 비슷한 것을 볼 때 많은 단어들이 비슷하게 중요한 역할을 한 것으로 볼 수 있다. 반면, <가격/이벤트>와 <배달/서비스>의 경우, 주요 1~5개의 단어들이 독보적으로 중요한 역할을 하고 있다. 이는, 가격, 배송 등의 주요 키워드들을 가지고 라벨링을 했기 때문에 자명한 결과이기도 하다.

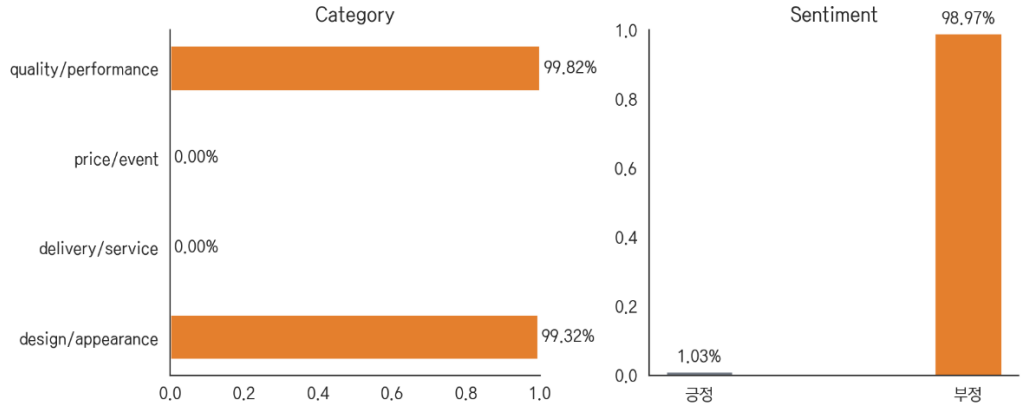


추가적으로, 하나의 리뷰를 입력 받았을 때, 해당 리뷰에 대한 긍정/부정 분석과 어떤 항목에 대한 내용을 포함하는지에 대해 예측하는 함수를 구성했다. 리뷰를 입력했을 때, 그 리뷰가 어떤 항목에 대한 내용이고 긍정 비율이 어떻게 되는지 알려주는 일종의 프로그램으로 직접 정리하였다. 아래 플롯들은 그의 예시이다.

✓ Review1: 내구성이 아쉽긴 한데, 그래도 가격 대비 만족합니다 배송도 빨랐고 포장 상태도 괜찮았어요



✓ Review2: 제품 실밥도 많이 풀려있고 허접합니다. 크기도 대형인가 싶을 정도로 크진 않네요



4.2. 한계점

해당 프로젝트의 한계점으로는, 다음의 사항들이 있다.

(1) 라벨링의 정확성 부족: 라벨링을 효율적으로 진행하다 보니, 직접 내용을 파악하며 라벨링을 하는 것에 비해 정확성이 부족하였다. 이는 모델 학습의 한계라기보다, 데이터 자체의 한계였기 때문에, 정확하게 라벨링이 되어 있는 다른 데이터가 존재할 경우 문제가 되지 않는다.

(2) Target간 상호 의존성 무시: 4가지 항목 간 상호 배타적인 관계가 아니었기 때문에, 이들의 관계를 무시하고 각각 이항 분류를 하는 것은 엄밀하게는 허점이 존재하는 분석 방법이었다. 이에 대해 더욱 조사해본 결과, 모든 분류 조합에 대해 학습시키는 방법, 이전 label에 대한 분류기 예측값을 그 다음 label 분류에 변수로서 사용하는 방법인 Classifier Chains 등 여러가지 방법들이 존재했다. 따라서, 이들의 의존 관계를 반영할 수 있는 분석 방법을 찾아서 재시도할 필요가 있었다.