SAMSUNG

자연어 분석을 통한 마케팅 전략 도출

OUTLINE

ANALYSIS

PREDICT

CONCLUSION

팀 소개, 역할 분담

주제선정이유

데이터 개요

연구 방법

데이터 수집 / 전처리

예측 모델 선택

네이버 카페 / 유투브 댓글 분석

결론해석

시사점, 연구한계

출처

OUTLINE

1-1 팀소개

TEAM | 미래전략실

성호경

데이터 크롤링 데이터 전처리 궁부정 분석 모델링 PPT 제작 노태진

데이터 크롤링 데이터 전처리 궁부정 분석 모델링 과정 총괄 이한경

데이터 크롤링 데이터 전처리 긍부정 분석 모델링 발표



TEAM | 미래전략실

분석 목표

애플, 샤오미, 삼성 전자기기 제품군을 키워드로 SNS채널과 네이버 쇼핑 리뷰에서 수집, 긍부정어 분석을 통해 마케팅 방안 도출

분석대상

- 3사 대표 전자기기 제품군 분석
- 네이버 카페 사용자 후기
- 네이버 쇼핑 리뷰 긍부정어 분석
- 인스타, 유투브 크롤링 데이터

데이터 수집 기간 / 채널

기간: 2020.01.17 ~ 2021.08.15

채널: 네이버 카페 (애플, 삼성, 샤오미 사용자 카페),

인스타그램, 유투브(댓글), 네이버쇼핑 리뷰 (총 85,799건)

주요 키워드 / 수집 방법

키워드: 삼성: 갤럭시, 버즈라이브, 버즈플러스, 갤탭, 갤럭시북, 갤럭시기어

샤오미: 미밴드, 홍미노트, 샤오미워치, 어메이즈핏, 샤오미폰, 샤오미휴대폰

애플: 맥북, M1, 아이폰, 에어팟, 애플워치, 아이패드

방법: Beautiful Soup과 Selenium을 사용해

각 채널 별 Crawler 제작

1-3 주제선정이유

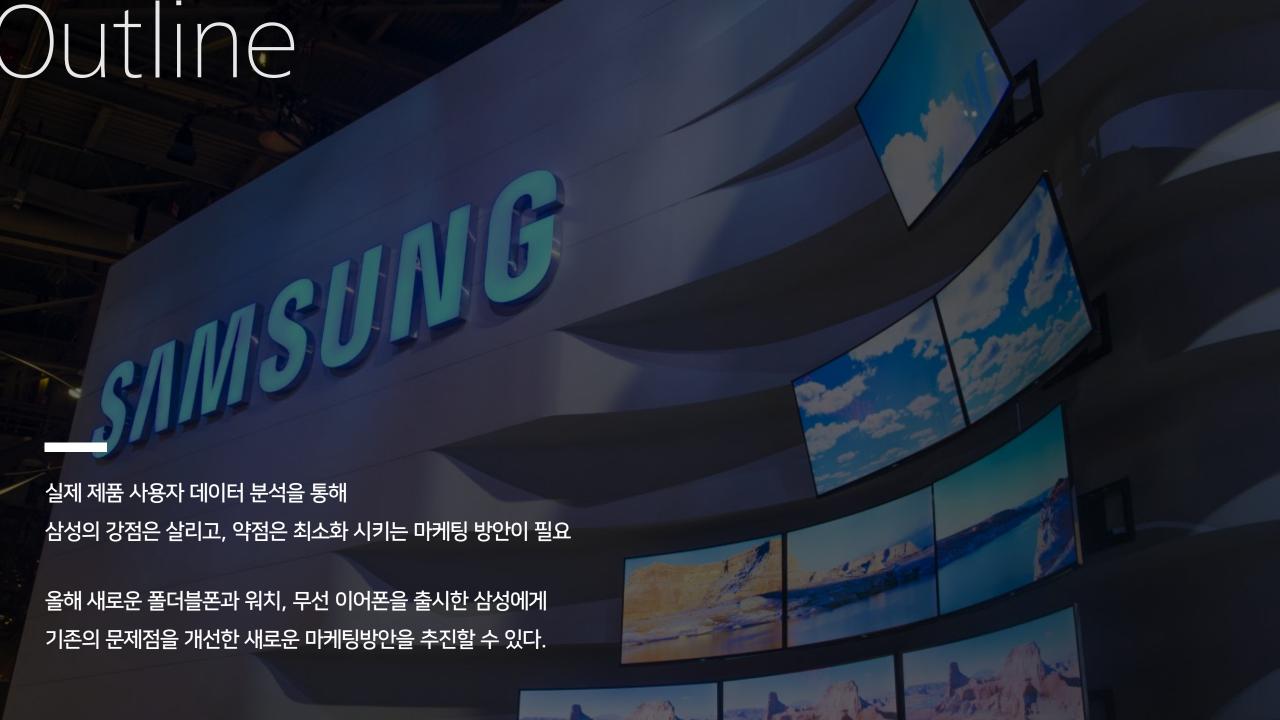
하락하는 삼성, 추격하는 애플과 샤오미



10년간 스마트폰 분야 최대 점유율을 차지하던 삼성은 애플과 샤오미의 증가세에 1위를 위협받고 있다.



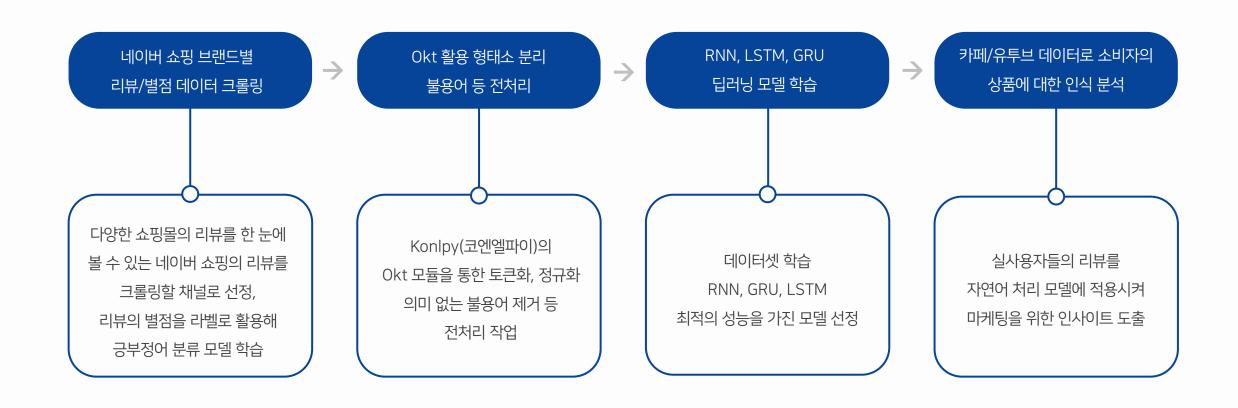
2016년 점유율 21.9%를 차지했던 삼성은 2021년 8.2%가 하락한 15.7%로 하락했다.



02 ANALYSIS

2-1 연구방법

프로젝트 전체 로드맵



2-2 데이터 수집



	▼		
	review	keyword	star
2003	와 레드색이 너무 예뻐요무난한 색상이랑 고민 많이했는데정말 탁월한 선택이였어요진짜	삼성전자 갤럭시 버즈 라이브	5
2004	갤럭서 버즈의 첫 기능이 있는 제품을 기다렸는데 타사 적용 블루투스 이어폰보다	삼성전자 갤럭시 버즈 라이브	5
2005	발송은 빨랐는데 연초라 택배 물량이 많아서 그런지 배송은 일정도 걸린것 같아요 판매	삼성전자 갤럭시 버즈 라이브	5
2006	블랙 색상이다 보니 모든 구성품 검은색으로 깔맞춤 되어있어 더 보기가 좋네요 구입	삼성전자 갤럭시 버즈 라이브	5
2007	무선 이어폰을 처음 사용해봤습니다 유선만 이용해서 무선은 음질이 떨어지지 않을까 걱	삼성전자 갤럭시 버즈 라이브	5
3141	아직 오픈형 하려면 양쪽 안정성좀	삼성전자 갤럭시 버즈 라이브	1
3142	음질은 좋은데 노캔기능이 없다시피 하네요불량같다 느낄정도로그리고 귀에 잘 안맞아요	삼성전자 갤럭시 버즈 라이브	1
3143	인식불가한 고장품이 와서 서비스센터에서 분이나 시간 허비하고 부분교품 받았어요 삼성	삼성전자 갤럭시 버즈 라이브	1
3144	0 0	삼성전자 갤럭시 버즈 라이브	1
3145	택배사가 너무 마음에 안들어요ㅠㅡㅠ 더러운 택배사 선택은 안되나ㅠㅠ	삼성전자 갤럭시 버즈 라이브	1

네이버 쇼핑

삼성 : 13437 애플 : 4586 샤오미 : 6984

총 25007개

******* *****



0 (부정) 라벨링





1 (긍정) 라벨링

전처리

- 1 중복 / 결측치 제거
- 2 정규표현식을 사용해 한글을 제외한 문자열 제거
- 3 특정 문자열(광고성 키워드)이 포함된 행 제거

2-2 데이터 수집

4 C	♥ 아이패드 •	1	2
0	인치 가지고 왔습니다	칩을 품은 아이패드 프로 하나 가지고왔습니다 게임용이 주목적으로 구입했고 휴대성도	2021.08.11. 21:02
1	미니	NaN	2021.08.11. 14:14
2	애플펜슬 교환받고 왔는데요ㅠ	아이패드 프로 세대 인치 사용한지는 년 넘었어요굿노트에서 자꾸 필기 오류가 생겨서	2021.08.11. 13:36
3	패드로 책 읽으시는 분 계신가요	출퇴근할 때 전철에서 아이패드로 전자책 읽는데눈이 좀 아픈 것 같아요 ㅜㅜㅜㅜ아이패	2021.08.11. 09:26
4	아이패드펜촉 팁 글씨필기감	도움말라이센스 동영상 인코딩 중입니다분 이상 소요될 수 있으며	2021.08.11. 01:36

0	# 미밴드 context
829	주말에 직구한 새 휴대폰이 빛의 속도로 도착년간 잘 쓰던 샤오미 홍미노트가 이제 버
830	블랙박스공기청정기 보조배터리샤오미휴대폰 진열제품구경공짜판플렛공짜남양주진접아이나비공식
831	샤오미 미맥스 직구가격대비 가성비 좋네 ㅋㅋ휴대폰 가성비갑 샤오미미맥스 샤오미 데일
832	샤오미휴대폰 카메라 나이성별이 인식댄다 ㅋ신기방기ㅋ 베이징 여행 좋은사람과 샤오미
833	듀얼 샤오미샤오미 샤오미휴대폰 샤오미 휴대폰 핸드폰 이거한국에서 사용가능 굿굿

	Q 버즈라이브 review	keyword
0	끝없는 맥북 논란과 이슈이번에는 화면만 열어도 깨지는디스플레이가 도마 위에 올랐습니다	M1
1	맥북 프로 인치가 저렇게 된 적 있는데 이건 만의 문제가 아니라 맥북 의 문제인 것	M1
2	개인적으로 애플은 내구성에 대해 조금더 발전된 모습이 필요해 보이는거 같음 아무리	M1
3	고압적인 정책 강제적인 업그레이드 정책 제한된 자유도 이 가지때문에 전 애플로 못	M1
4	이 영상을 맥북으로 보는 심정이 참 ㅜㅜ	M1

네이버 카페

삼성 : 1881 애플 : 2252 샤오미 : 601

총 4734개

Youtube

삼성 : 20111 애플 : 20117 샤오미 : 11050

총 51278개

인스타그램

삼성 : 1243 애플 : 2704 샤오미 : 833

총 4780개

전처리

1 중복 / 결측치 제거

- 2 정규표현식을 사용해 한글을 제외한 문자열 제거
- 3 특정 문자열(광고성 키워드)이 포함된 행 제거



2-3 자연어처리

1

자연어 처리

- 표현 방법이 다른 단어들을 통합시켜서 같은 단어로 만들어준다.

: 말뭉치를 모아 정제 - 토큰화 - <mark>정규회</mark>를 거쳐야 한다.

말뭉치에서 더 이상 나눌 수 없는 단위로 나누는 작업

2

감성분류

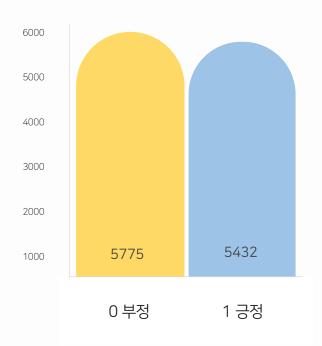
: 텍스트에 들어있는 의견이나 감성, 평가, 태도 등의 주관적인 정보를 컴퓨터를 통해 분석



레이블분포

긍정/부정을 나누기 위해 네이버 평점 1~3점을 0으로, 이외는 1점으로 할당

훈련용 데이터의 라벨 수



토큰화

불용어 (stopwords) 를 지정, 제거하고 Okt를 이용해 토큰화 실행

stopwords = ['가','게','고','과','네','는','다','도','들','듯','를','에','와','으로',
'은','을','의','이','인','임','자','잘','좀','지','하다','한']

```
X_test = []
for sentence in test_data['review']:
    temp_X = okt.morphs(sentence, stem=True) # 토큰화
    temp_X = [word for word in temp_X if not word in stopwords] # 불용어 제거
    X_test.append(temp_X)
print(X_test[:3])
```

[···, ['좋다', '빨르다', '가볍다', '예쁘다'], ['아이', '착용', '가볍다', '좋다'], ['배송', '빠르다', '좋다', '상품', '받다', '매우', '만족하다', '색도', '너무', '이쁘다', '케이스', '마음', '들다', '다만', '개인', '적', '필름', '붙이다', '조금', '힘들다', '저', '실패하다', '전적', '있다', '따로', '돈', '주다', '붙이다'], ···]

2-3 자연어처리

정수 인코딩

텍스트를 숫자로 처리하도록 훈련 데이터와 테스트 데이터에 정수 인코딩을 수행

tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)

print(tokenizer.word_index)

{'좋다': 1, '배송': 2, '받다': 3, '있다': 4, '자다': 5, '제품': 6, '보다': 7, '빠르다': 8, '사용': 9, '너무': 10, '되다': 11, '구매': 12, '로': 13, '같다': 14, '오다': 15, '없다': 16, '이다': 17, '안': 18, '쓰다': 19, '요': 20, '에서': 21, '않다': 22, '사다': 23, '가격': 24, '하고': 25, '만족하다': 26, '것': 27, '주문': 28, '때': 29, '만': 30, …}

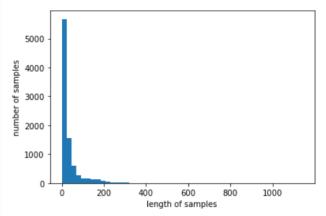
패딩

하나의 행렬로 만들기 위해 서로 다른 길이의 샘플들의 길이를 동일하게 만듦

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

리뷰의 최대 길이 : 1141

리뷰의 평균 길이 : 33.27797026936403



```
max_len = 100
below_threshold_len(max_len, X_train)
```

전체 샘플 중 길이가 100 이하인 샘플의 비율: 91.17022465630937

2-3 자연어 처리

정수 인코딩

텍스트를 숫자로 처리하도록 훈련 데이터와 테스트 데이터에 정수 인코딩을 수행

```
threshold = 3
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
```

```
# 전체 단어 개수 중 빈도수 2 이하인 단어는 제거
# 0번 패딩 토큰을 고려하여 +1
vocab_size = total_cnt - rare_cnt + 1 |
print("단어 집합의 크기: ", vocab_size)
```

단어 집합의 크기 : 5092

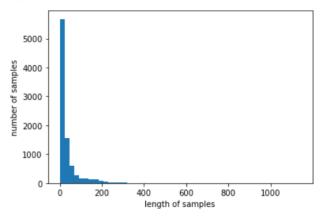
패딩

하나의 행렬로 만들기 위해 서로 다른 길이의 샘플들의 길이를 동일하게 만듦

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

리뷰의 최대 길이 : 1141

리뷰의 평균 길이 : 33.27797026936403

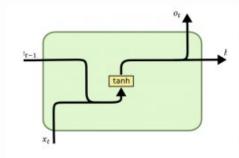


```
max_len = 100
below_threshold_len(max_len, X_train)
```

전체 샘플 중 길이가 100 이하인 샘플의 비율: 91.17022465630937

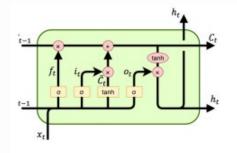
O3 PREDICT

RNN



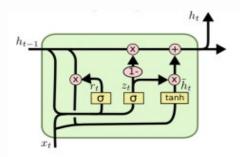
순서가 있는 데이터를 처리하기 위한 인공 신경망

LSTM



RNN의 Vanishing gradient problem을 완화한 방법

GRU



LSTM에 비해서 단순한 구조이면서도 긴 데이터를 잘 처리하는 방법

3-2 RNN

Patience: 검증 데이터 손실이 4회 증가하면 학습을 조기종료

Epoch: 20번 수행

batch_size : 각 데이터의 사이즈 50

Validation_split: 훈련데이터 중 20프로를 검증데이터로 사용

정확도 약 82%, loss 약 38%인 4번째 모델 확정

```
from sklearn.metrics import confusion_matrix
y pred = model rnn.predict(X test rnn)
print(confusion_matrix(y_test, y_pred))
from sklearn.metrics import classification_report
print(classification report(y test, y pred))
3202/3202 [======== ] - 5s 2ms/step
[[ 308 1331]
[ 203 1360]]
                         recall f1-score
             precision
                                           support
                  0.60
                           0.19
                                    0.29
                                              1639
                           0.87
                                    0.64
                 0.51
                                              1563
                                     0.52
                                              3202
   accuracy
                 0.55
                           0.53
                                    0.46
                                              3202
   macro avg
weighted avg
                 0.56
                           0.52
                                    0.46
                                              3202
```

Train 정확도 56% | Test 정확도 52% F1-score (label 0) 29%

RNN은 부정 리뷰 학습에서 성능 미달

3-3 GRU

```
model_GRU = Sequential()
model_GRU.add(Embedding(vocab_size, 100))
model_GRU.add(GRU(128))
model_GRU.add(Dense(1, activation='sigmoid'))

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
mc = ModelCheckpoint('GRU_model.h5', monitor='val_acc', mode='max', verbose=1, save_bes

model_GRU.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history_GRU = model_GRU.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size
```

```
Epoch 1/15
6020/6020 [=============] - 85s 14ms/step - loss: 0.5650 - acc: 0.712
1 - val_loss: 0.4860 - val_acc: 0.7669

Epoch 00001: val_acc improved from -inf to 0.76693, saving model to GRU_model.h5
```

Patience: 검증 데이터 손실이 4회 증가하면 학습을 조기종료

Epoch : 15번 수행

batch_size : 각 데이터의 사이즈 60

Validation_split: 훈련데이터 중 20프로를 검증데이터로 사용

정확도 약 71%, loss 약 56%인 **1번째 모델** 확정

```
from sklearn.metrics import confusion_matrix
y_pred = model_GRU.predict(X_test)
y_pred_ = []
for i in range(len(y pred)):
    if y pred[i][0] > 0.5:
       y_pred_.append(1)
    else:
        y_pred_.append(0)
y pred
print(confusion_matrix(y_test, y_pred_))
from sklearn.metrics import classification_report
print(classification report(y test, y pred ))
[[1069 570]
 [ 283 1280]]
              precision
                          recall f1-score support
           0
                   0.79
                            0.65
                                                1639
                   0.69
                            0.82
                                      0.75
                                                1563
                                      0.73
                                                3202
    accuracy
                            0.74
                                      0.73
                                                3202
   macro avg
                   0.74
weighted avg
                   0.74
                            0.73
                                      0.73
                                                3202
```

Train 정확도 71% | Test 정확도 73% F1-score 71% RNN모델에 비해 label 0을 더 잘 분류

3-4 LSTM

model_lstm = Sequential()

Patience: 검증 데이터 손실이 4회 증가하면 학습을 조기종료

Epoch: 15번 수행

batch_size : 각 데이터의 사이즈 60

Validation_split: 훈련데이터 중 20프로를 검증데이터로 사용

정확도 약 82%, loss 약 38%인 **4번째 모델** 확정

```
from sklearn.metrics import confusion_matrix
y pred = model lstm.predict(X test)
y pred = []
for i in range(len(y_pred)):
    if y_pred[i][0] > 0.5:
       y_pred_.append(1)
    else:
       y_pred_.append(0)
y_pred_
print(confusion_matrix(y_test, y_pred_))
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_))
[[1171 468]
 [ 367 1196]]
                          recall f1-score support
              precision
                                      0.74
                                                1639
                   0.76
                            0.71
                            0.77
                  0.72
                                      0.74
                                                1563
                                      0.74
                                                3202
    accuracy
   macro avg
                  0.74
                            0.74
                                      0.74
                                                3202
weighted avg
                  0.74
                            0.74
                                      0.74
                                                3202
```

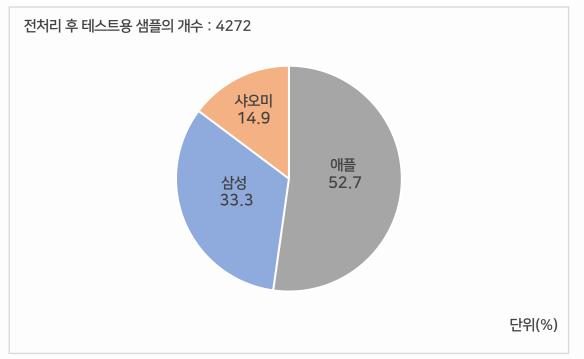
Train 정확도 82% | Test 정확도 74% F1-score 74%

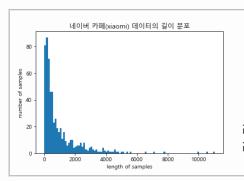
RNN / GRU 모델에 비해 높은 예측률 >>> 모델 선정

3-5 네이버 카페 분석 💍

```
plt.figure(figsize=(12, 10))
ratio = [len(df_apple), len(df_samsung), len(df_xiaomi) ]
labels = ["apple", "samsung", "xiaomi"]

plt.pie(ratio, labels=labels, autopct='%.1f%%', startangle=260, counterclock=False)
plt.show()
print('전처리 후 테스트용 샘플의 개수:',len(df_apple)+len(df_samsung)+len(df_xiaomi))
```

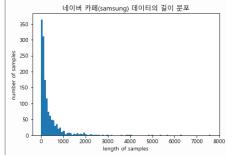




리뷰의 최대 길이 : 23341 리뷰의 평균 길이 : 335.18

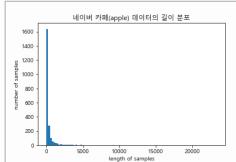


SAMSUNG



리뷰의 최대 길이 : 7624

리뷰의 평균 길이: 367.8029556650246



리뷰의 최대 길이: 11025

리뷰의 평균 길이: 984.2993311036789

MI

3-5 네이버 카페 분석 🥌



```
label = []
   score_list = []
   for review in df_apple["review"]:
       new_sentence = okt.morphs(review) # 토근화
       new_sentence = [word for word in new_sentence if not word in stopwords] # 물용어 제거
       encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
       pad_new = pad_sequences(encoded, maxlen = 100)
       score = float(model test.predict(pad new))
       if(score > 0.5):
           label.append(1)
           score list.append(score)
       else:
           label.append(0)
           score_list.append(score)
   xiaomi_score = pd.DataFrame([label, score_list]).T
   xiaomi_score.columns = ["label", "score"]
   print(xiaomi_score[xiaomi_score["label"]==0].mean())
   print(xiaomi_score[xiaomi_score["label"]==1].mean())
   print(xiaomi_score.describe())
label
        0.000000
score
        0.207032
dtype: float64
        1.0000000
label
        0.863782
score
dtype: float64
             label
                         score
count 2250.000000 2250.000000
         0.658222
                      0.639320
mean
std
         0.474411
                      0.344702
min
         0.000000
                      0.002578
         0.000000
                      0.318488
25%
                      0.768040
         1.000000
                      0.961851
75%
         1.000000
         1.000000
                      0.999201
```



SAMSUNG



0.000000

label	0.000000	
score	0.207032	
dtype:	float64	
label	1.000000	
score	0.863782	
dtype:	float64	
	label	score
count	2250.000000	2250.000000
mean	0.658222	0.639320
std	0.474411	0.344702

0.000000

0.000000

1.000000

1.000000

1.000000

min

25%

50%

75%

max

label	0.000000	
score	0.206288	
dtype:	float64	
label	1.000000	
score	0.874746	
dtype:	float64	
	label	score
count	1421.000000	1421.000000
mean	0.695285	0.671057
std	0.460449	0.338299
min	0.000000	0.002628
25%	0.000000	0.372983
50%	1.000000	0.823559
75%	1.000000	0.966880
max	1.000000	0.998767

score	0.198157		
	float64		
label	1.000000		
score	0.901951		
dtype:	float64		
	label	score	
count	598.000000	598.000000	
mean	0.789298	0.753660	7
std	0.408149	0.313595	
min	0.000000	0.002463	
25%	1.000000	0.605136	
50%	1.000000	0.918632	
75%	1.000000	0.974123	
max	1.000000	0.998602	

부정 (score < 0.5)

0.002578

0.318488

0.768040

0.961851

0.999201

긍정 (score > 0.5)

label

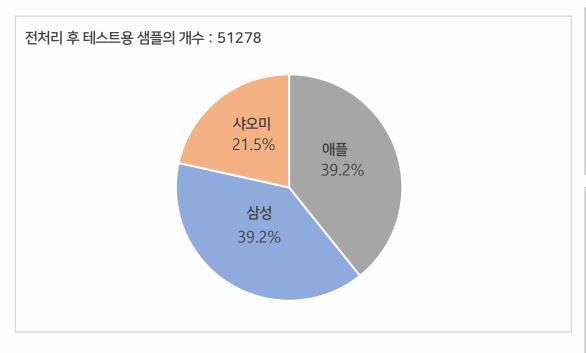
	애플	삼성	샤오미
긍부정 비율	긍정 65 : 35 부정	긍정 70 : 30 부정	긍정 78 : 22 부정
Score 평균	긍정 0.86 : 0.2 부정	긍정 0.87 : 0.2 부정	긍정 0.9 : 0.19 부정

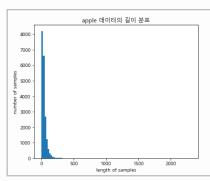
애플 샤오미 삼성

3-6 유투브 댓글 분석



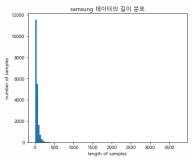
```
plt.figure(figsize=(12, 10))
ratio = []
labels = []
for word in df["company"].unique():
    ratio.append(len(df[df["company"]==word]))
   labels.append(word)
plt.pie(ratio, labels=labels, autopct='%.1f%%', startangle=260, counterclock=False)
plt.show()
print('전처리 후 테스트용 샘플의 개수 :',sum(ratio))
```





리뷰의 최대 길이 : 2305

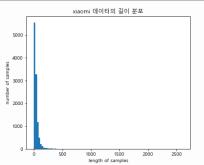
리뷰의 평균 길이: 41.45414326191778



SAMSUNG

리뷰의 최대 길이: 3778

리뷰의 평균 길이: 47.91939734473671



리뷰의 최대 길이: 2609

리뷰의 평균 길이: 39.46850678733032

mı

3-6 유투브 댓글 분석

```
from keras.models import load_model
model_test = load_model('model_lstm.h5')

def sentiment_predict(new_sentence):
    new_sentence = okt.morphs(new_sentence) # 토근화
    new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
    encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
    pad_new = pad_sequences(encoded, maxlen = 100) # 패딩
    score = float(model_test.predict(pad_new)) # 예측
    if(score > 0.5):
        print("{:.2f}% 확률로 긍정 리뷰입니다.".format(score * 100))
    else:
        print("{:.2f}% 확률로 부정 리뷰입니다.".format((1 - score) * 100))

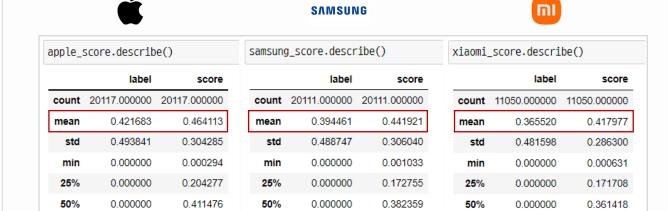
df["review"][0]

'끝없는 맥북 논란과 이슈이번에는 화면만 열어도 깨지는디스플레이가 도마 위에 올랐습니다'

sentiment_predict(df["review"][0])

53.79% 확률로 부정 리뷰입니다.
```

```
label = []
score_list = []
for review in df[df["company"]=="apple"]["review"]:
   new_sentence = okt.morphs(review) # 토큰화
   new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
   encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
   pad_new = pad_sequences(encoded, maxlen = 100)
   score = float(model_test.predict(pad_new))
   if(score > 0.5):
       label.append(1)
       score_list.append(score)
   else:
       label.append(0)
       score_list.append(score)
apple_score = pd.DataFrame([label, score_list]).T
apple_score.columns = ["label", "score"]
```



1.000000

1.000000

75%

max

0.708176

0.999583

75%

max

1.000000

1.000000

0.644604

0.998846

75%

max

1.000000

1.000000

0.736177

0.999599

 애플
 삼성
 샤오미

 공부정 비율
 긍정 42 : 58 부정
 긍정 39 : 61 부정
 긍정 36 : 64 부정

애플 > 삼성 > 샤오미

제품군 별 분석 결과 | 애플 > 삼성 > 샤오미

M1 apple label 0.000000 score 0.239886 dtype: float64 label 1.0000000 score 0.756351 dtype: float64 label Score count 2664.0000000 mean 0.394895 0.443835 std 0.488920 0.290931 min 0.000000 0.3088998 75% 1.0000000 0.388998 75% 1.0000000 0.6889971 max 1.0000000 0.997786	역복 apple label 0.000000 score 0.242141 dtype: float64 label 1.000000 score 0.792609 dtype: float64 label 1.000000 score 0.792609 dtype: float64 label 5.507e label 5.516365 std 0.500099 0.311318 min 0.000000 0.006728 25% 0.000000 0.2440602 50% 0.000000 0.496348 75% 1.000000 0.917271 max 1.000000 0.998983	O O → apple apple 0.000000 Score 0.219999 dtype: float64 abel 1.000000 Score 0.736549 dtype: float64 abel 1.000000 3775.000000 3775.000000 3775.000000 3775.000000 3775.000000 3775.000000 3775.000000 3775.000000 3775.000000 3775.000000 3775.000000 3775.00000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.00000000 3775.0000000 3775.0000000 3775.0000000 3775.00000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.00000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.00000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.0000000 3775.00000000 3775.00000000 3775.0000000 3775.00000000 3775.00000000 3775.00000000 3775.000000000 3775.000000	O O III	OFO[AZ apple label 0.000000 score 0.238112 dtype: float64 label 1.0000000 score 0.752589 dtype: float64 label 1.0000000 score 0.752589 dtype: float64 label score label score 0.375503 0.431300 std 0.484313 0.288270 min 0.000000 0.001541 25% 0.000000 0.197092 50% 0.000000 0.372581 75% 1.0000000 0.372581 75% 1.0000000 0.655108 max 1.0000000 0.998580	애플워치 apple label 0.000000 score 0.240979 dtype: float64 label 1.000000 score 0.761864 dtype: float64 count 3688.000000 mean 0.394794 0.446621 std 0.488873 0.792009 min 0.000000 0.001258 25% 0.000000 0.198009 50% 0.000000 0.390847 75% 1.000000 0.998416
選号	갤럭시복 samsung label 0.000000 score 0.231957 dtype: float64 label 1.000000 score 0.78557 dtype: float64 Label Score count 3715.000000 3715.000000 mean 0.433109 0.471726 std 0.495572 0.310139 min 0.000000 0.002556 25% 0.0000000 0.201500 56% 0.0000000 0.416459 75% 1.000000 0.758837 max 1.000000 0.999408	결복시 Samsung label 0.000000 score 0.227785 dtype: float64 label 1.000000 score 0.766739 dtype: float64 label 1.000000 score 0.766739 dtype: float64 label Score label Score label Score label Score label Score label Score 0.399817 0.443268 std 0.489935 0.301254 min 0.000000 0.02040 25% 0.000000 0.179190 50% 0.000000 0.179190 50% 0.000000 0.391075 75% 1.0000000 0.391075 75% 1.0000000 0.998436	필법 samsung label 0.000000 score 0.222832 dtype: float64 label 1.000000 score 0.815007 dtype: float64	버즈라이브 samsung label 0.000000 score 0.230694 dtype: float64 label 1.000000 score 0.743343 dtype: float64 label score count 4143.0000000 4143.0000000 mean 0.336230 0.403062 std 0.472476 0.282337 min 0.000000 0.163200 50% 0.0000000 0.163200 50% 0.0000000 0.606500 max 1.000000 0.997306	サス亜日本 samsung label 0.0000000 score 0.210754 dtype: float64 label 1.00000 score 0.74223 dtype: float64 label 1.00000 score 0.74223 dtype: float64 label 2.000000 count 3157.000000 mean 0.299018 0.369675 std 0.457900 0.282711 min 0.000000 0.124701 50% 0.000000 0.124701 50% 0.000000 0.386741 75% 1.000000 0.572858 max 1.000000 0.998553
사오미워치 xiaomi label 0.0000000 score 0.223597 dtype: float64 label 1.000000 score 0.733945 dtype: float64 label score count 1499.000000 1499.000000 mean 0.346898 0.400635 std 0.476142 0.281695 min 0.0000000 0.01102 25% 0.0000000 0.162116 50% 0.000000 0.343649 75% 1.0000000 0.93839	사오미폰 xiaomi label 0.000000 score 0.235623 dtype: float64 label 1.000000 score 0.749864 dtype: float64 label score count 3592.000000 3592.000000 mean 0.394209 0.438342 std 0.488748 0.290123 min 0.000000 0.001124 25% 0.000000 0.392899 75% 1.0000000 0.392899 75% 1.0000000 0.996894	샤오미휴대폰 xiaomi label 0.000000 score 0.243046 dtype: float64 label 1.00000 score 0.75153 dtype: float64 label score count 1467.000000 mean 0.405590 0.449282 std 0.491173 0.288468 min 0.000000 0.002429 25% 0.000000 0.002429 50% 0.000000 0.204926 50% 0.000000 0.204926 50% 0.000000 0.204926 50% 0.000000 0.204926	O 마 이골핏 xiaomi label 0.000000 score 0.212903 dtype: float64 label 1.00000 score 0.73134 dtype: float64 Label score count 721.000000 721.000000 mean 0.317614 0.377566 std 0.465872 0.280211 min 0.000000 0.002888 25% 0.000000 0.139912 50% 0.000000 0.389741 75% 1.000000 0.581002 max 1.000000 0.996066	□년번도 xiaomi label 0.000000 score 0.221195 dtype: float64 label 1.000000 score 0.725211 dtype: float64 label 2235.000000 mean 0.286353 0.365522 std 0.452158 0.268025 min 0.000000 0.308741 50% 0.000000 0.308741 75% 1.000000 0.996175	\$\text{8}\text{I} \text{LE} \text{ xiaomi} \\ \text{label} \ \ 0.000000 \\ \text{score} \ 0.236552 \\ \text{dtype: float64} \\ \text{label} \ 1.000000 \\ \text{score} \ 0.756657 \\ \text{dtype: float64} \\ \text{label} \ \ \text{label} \ \text{Score} \\ \text{count 1556.000000} \\ \text{mean} \ 0.416016 \ 0.432674 \\ \text{std} \ 0.493657 \ \text{std} \ 0.493657 \ \text{olomono} \\ \text{min} \ \ 0.000000 \ 0.006631 \\ 25% \ 0.000000 \ 0.0006631 \\ 50% \ 0.000000 \ 0.207641 \\ 50% \ 0.000000 \ 0.207641 \\ 50% \ 0.000000 \ 0.7076432 \\ \text{max} \ 1.000000 \ 0.798846 \\ \text{max} \ \ 1.000000 \ 0.998846 \\ \end{argument}

CONCLUSION





디자인 하나만보고 넘어가야지 했는데 이 영상보고 더 좋은 선택할 수 있을 거 같아요



앱등이 갈아탔습니다.. 제품은 받았는데 아직 미개통 ㅠ 디자인은 만족



YouTube 데이터는 제품 미사용자의 의견을 반영 **브랜드에 대한 이미지**를 확인할 수 있는 지표



으아 … 아이폰 미니 배터리

아침에 100프로 충전하고 일하면서 노래듣고 점심 때 잠깐 유투브 봤는데 지금 방전됐네요 … ㅜㅜ 6s 처럼 보조배터리 들고 다녀야겠네요 배터리 문제 해결하신 분 계신가요?



🌃 🏻 드디어 패드병 완치 ㅠㅠ

아이클라우드에 심하게 데인 적이 있어서 아이클라우드는 단순 백업용도로만!! 그러다 보니 기본용량 안에서만 사용해요 M1칩 빠르고 아주 만족입니다! 필기할 때 종이 안 들고 다니니 편하네요



네이버 카페는 다수의 제품 사용자가 모여 있음 성능/디자인/이슈가 되는 문제 등 **제품에 대한 후기** 확인 가능 4-1 결론

모델 결과 데이터 요약







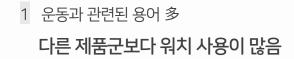
삼성의 마케팅 약점 = 브랜드 이미지 약화

4-1 결론

◎ 인스타 데이터를 활용한 워드클라우드와 빈도수 분석







2 애플 관련 키워드와 가성비 키워드는 두 제품 간의 차이가 보임 가성비라는 이미지가 강함





1 애플은 운동, 드로잉, 공부 등 다양한 키워드 존재 다양한 제품군 사용

* 애플에는 다른 제품군에 대한 언급이 없음

4-1 <u>a</u>Z

◎ 인스타 데이터를 활용한 워드클라우드와 빈도수 분석





- 1 애플 검색결과에는 삼성이 없지만 삼성 검색결과에는 애플 관련 키워드가 존재 삼성은 애플과 비교선상에 있다
- 2 스꾸, 스티커 등 제품을 꾸미는데 사용하는 키워드가 같이 검색 자신이 원하는 대로 커스터마이징하는 갤럭시 사용자 내부의 한정된 문화는 타 브랜드와 다른 양상

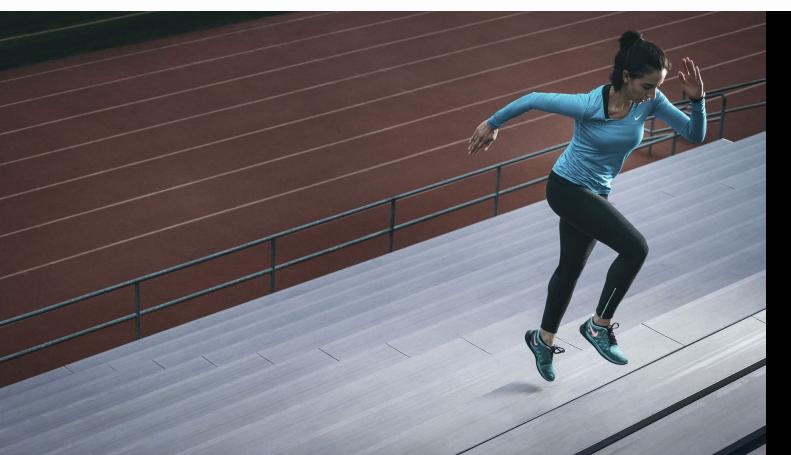
사용자의 일상을 공유할 수 있는 브랜드로서 이미지 메이킹이 필요하다

4-1 결론



NEW LOGO TYPE

다양성과 어우러지는 모습을 도형으로 표현하고, 삼성의 다양한 제품군을 한 곳에 모아 강한 연동성을 표현하였다.



포스트 코로나를 맞이하는 새로운 시대가 열립니다



연구한계

- 텍스트 전처리

수집 대상 텍스트가 제품평 외의 정보를 포함, 섬세한 전처리를 요구

- 자연어 처리 모델링

모델링에 사용한 Okt는 비교적 최근 단어와 같이 라이브러리 사전에 등록되지 않은 형태소를 정확하게 분류할 수 없어 모델의 성능을 저하

시사점

더 이상 기업과 기업의 영업활동은 단순히 제품의 성능과 가격으로만 판단되지 않는다. ESG가 그 대표적인 예로, 환경적으로 지속 가능한지, 사회적이고 윤리적인지, 지배구조가 투명한지 등의 비재무적 요소를 고려

전자기기에서도 성능과 별개로 얼마나 친근한지, 무엇이 연상되는지, 자신과 동질감을 느끼는지 등의 요소가 소비에 영향을 끼치므로 기업에서는 장기적으로 소비자들이 가질 브랜드 이미지 또한 충분히 고려해야 한다.

4-3 출처

[조선일보] 애플·삼성 위협하는 샤오미, 스마트폰 이어 3년 만에 '가성비 태블릿'도 출시 https://biz.chosun.com/it-science/ict/2021/08/15/SG5F56VFKBAJHK6TMQR7KTZPI4/

[아시아투데이] 삼성전자 스마트폰 점유율 20% 또 깨졌다···옥 죄어오는 애플·샤오미 https://www.asiatoday.co.kr/view.php?key=20210730010018273

[중앙일보] 애플이 또 애플했다···'아이폰12 효과'에 2분기 역대 최대 매출 https://news.joins.com/article/24115641

글로벌 스마트폰 제조업체별 월 판매량 점유율. 사진=카운터포인트리서치 https://www.etnews.com/20210806000061

[위키독스] 딥 러닝을 이용한 자연어 처리 입문 https://wikidocs.net/94600

텍스트 마이닝 프로젝트 (2020.6~) Data Crawling (휴먼지능정보공학과 류지혜, 이지민) https://github.com/Jimin980921/Text_mining/blob/master/Project/textmining_project_crawling.ipynb

감사합니다

Q&A