Sentiment Analysis for Financial News

161655 허정화

<INDEX>

- 주제
- 기존 연구 소개
- 데이터
- 모델 & 결과분석
- 기대효과

- 주 제

" 금융뉴스 헤드라인 감정 분석 "

목표

1. 뉴스의 헤드라인에 적합한 **전처리**

2. 감정을 분류할 모델 **학습 및 예측**

- 기존 연구 소개

기존 연구

정기적인
경제 발전에 대한 설문조사

신뢰도 지표 계산

But
시간이 지날 수 록
빠르게 뒤쳐짐

극복

" 미디어를 투자자입장에서 바라보자 "

투자 정보는 주로 미디어를 기반으로 함
이는 투자자(시장참여자)의 행동에 영향을 줌

유럽 시장은 아래로 마감하였습니다 닥스 2.56% 아래로
부터 Investing.com – 29 분전
Investing.com – 수요일 마감종 후에 유럽 주식들이 아래로, 유럽 무역 폐쇄에, 독일 닥스가 2.56% 떨어졌습니다, 영국 런던증권거래소의 F...

대한항공 '코로나 보릿고개' 넘자, 1조원 유상증자
부터 중앙일보 | 경제 – 1 시간 전
대한항공은 13일 오전 서울 중구 서소문 사옥에서 이사회를 열고 1조원 규모의 유상증자와 국책은행을 통한 정부 자금 지원안 1조 2000억원의 실행을 결의했다. ...

코로나에도 영업익 484억…'절치부심' 이마트 흑자로 돌아섰다
부터 중앙일보 | 경제 – 1 시간 전
이마트는 1분기 매출액 5조2108억원, 영업이익 484억원(연결 재무제표 기준)을 기록했다고 13일 코스피 시장에 공시했다. 1분기 매출액은 1년 전과 비교하면...

: 예시 뉴스

- 데 이 터

infomation

이름	Sentiment Analysis for Financial News
특징	소매 투자자 관점에서 금융 뉴스 헤드라인에 대한 감정이 라벨링 되어있음
용도	텍스트 감정 분류 학습 및 예측을 목적으로 사용
출처	kaggle

https://www.kaggle.com/ankurzing/sentiment-analysis-for-financial-news

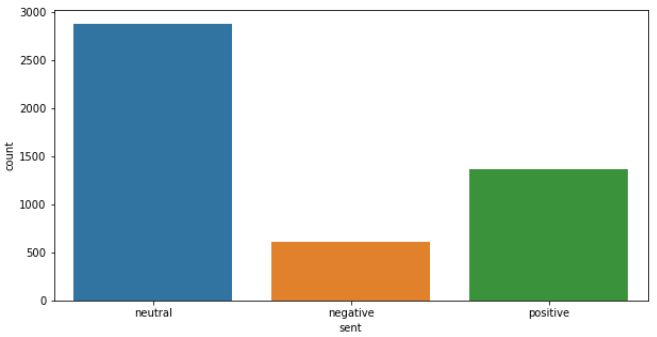
- 데 이 터

statistic

	sentiment	text
0	neutral	According to Gran , the company has no plans t...
1	neutral	Technopolis plans to develop in stages an area...
2	negative	The international electronic industry company ...
3	positive	With the new production plant the company woul...
4	positive	According to the company 's updated strategy f...
...	...	...
4841	negative	LONDON MarketWatch -- Share prices ended lower...
4842	neutral	Rinkuskiai 's beer sales fell by 6.5 per cent ...
4843	negative	Operating profit fell to EUR 35.4 mn from EUR ...
4844	negative	Net sales of the Paper segment decreased to EU...
4845	negative	Sales in Finland decreased by 10.5 % in Januar...

4846 rows × 2 columns

- sentiment ratio



neutral	2879 (59%)
positive	1363 (28%)
negative	604 (12%)
total	4846

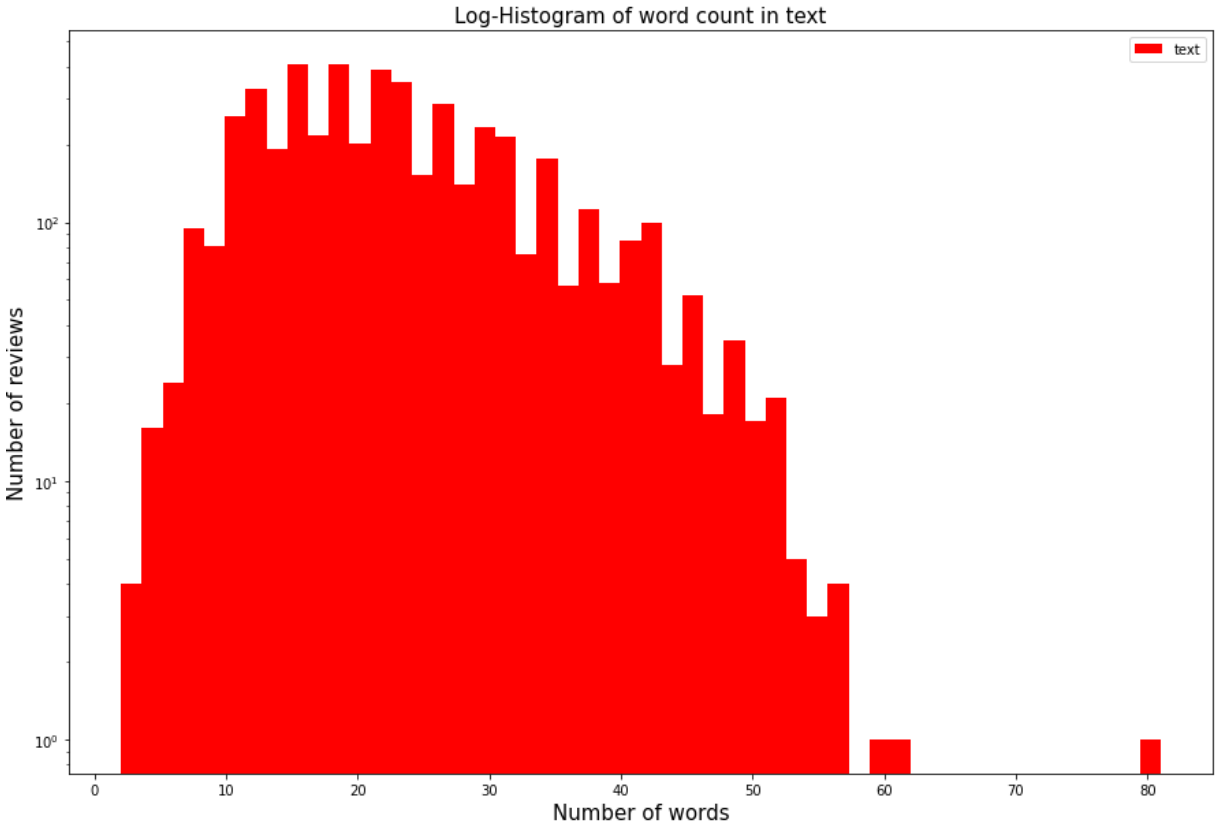
- 데 이 터

wordcloud

가장 많이 사용되는 단어 : EUR, million, company

- 데 이 터

단어 개수 분포



단어 개수 최대 값: 81

단어 개수 최소 값: 2

단어 개수 평균 값: 23.11

단어 개수 표준편차: 9.96

단어 개수 중간 값: 21.0

- 데 이 터

특수문자 및 대, 소문자 비율

```
qmarks = np.mean(data['text'].apply(lambda x: '?' in x)) # 물음표가 구두점으로 쓰임
fullstop = np.mean(data['text'].apply(lambda x: '.' in x)) # 마침표
capital_first = np.mean(data['text'].apply(lambda x: x[0].isupper())) #  첫번째 대문자
capitals = np.mean(data['text'].apply(lambda x: max([y.isupper() for y in x]))) # 대문자가 몇개
numbers = np.mean(data['text'].apply(lambda x: max([y.isdigit() for y in x]))) # 숫자가 몇개

print('물음표: {:.2f}%'.format(qmarks * 100))
print('마침표: {:.2f}%'.format(fullstop * 100))
print('첫 글자가 대문자: {:.2f}%'.format(capital_first * 100))
print('대문자: {:.2f}%'.format(capitals * 100))
print('숫자: {:.2f}%'.format(numbers * 100))
```

```
물음표: 0.17%
마침표: 99.46%
첫 글자가 대문자: 92.88%
대문자: 99.83%
숫자: 53.32%
```

<전처리 해줘야 할 것>

- 특수문자 제거
- 대문자 => 소문자 바꾸기

전처리

```
def preprocessing(text, remove_stopwords = False ):

    #1. 영어가 아닌 특수문자들을 공백(" ")으로 바꾸기
    text = re.sub("[^a-zA-Z]", " ", text)

    #2. 대문자들을 소문자로 바꾸고 공백단위로 텍스트들 나눠서 리스트로 만든다.
    words = text.lower().split()

    if remove_stopwords:
        #3. 불용어들을 제거
        #영어에 관련된 불용어 불러오기
        stops = set(stopwords.words("english"))
        # 불용어가 아닌 단어들로 이루어진 새로운 리스트 생성
        words = [w for w in words if not w in stops]
        # 4. 단어 리스트를 공백을 넣어서 하나의 글로 합친다.
        clean= ' '.join(words)

    else: # 불용어 제거하지 않을 때
        clean = ' '.join(words)

    return clean
```

1. 특수문자 공백으로 바꾸기
2. 대문자 -> 소문자
3. 불용어 제거

불용어란?

문장에서 자주 출현하나
의미에 큰 영향을 주지 않는 데이터
Ex)조사, 관사

" 머신러닝은 여기서 csv파일로 저장 "

전처리(딥러닝 용)

4. 각 단어 인덱스로 벡터화 -> 인덱스변환

텐서플로우의 Tokenizer

```
from tensorflow.python.keras.preprocessing.text import Tokenizer
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(clean)
text_sequences = tokenizer.texts_to_sequences(clean)
```

```
print(text_sequences[0])
```

```
[44, 3135, 2, 229, 524, 39, 84, 2485, 2, 609]
```

전처리(딥러닝 용)

5. 일정 길이로 자르고 부족한 부분을 특정 값으로 채움 -> 패딩과정

텐서플로우의 pad sequences

```
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
```

```
MAX_SEQUENCE_LENGTH = 21 #단어 개수 중간 값 사용
inputs = pad_sequences(text_sequences, maxlen=MAX_SEQUENCE_LENGTH, padding='post')
print('Shape of data: ', inputs.shape)
```

```
Shape of data:  (4846, 21)
```

- 데 이 터

전처리(딥러닝 용)

5. 일정 길이로 자르고 부족한 부분을 특정 값으로 채움 -> **패딩과정**

텐서플로우의 pad sequences

```
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
```

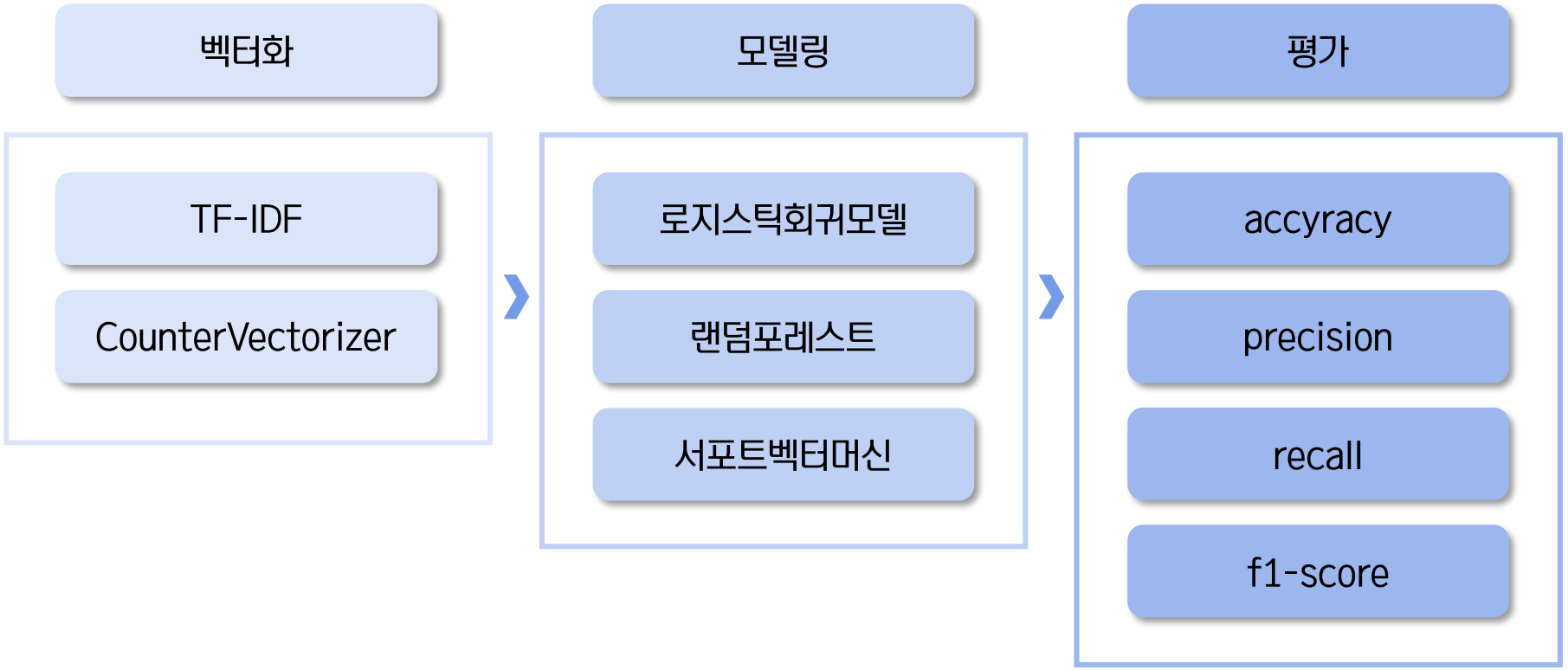
```
MAX_SEQUENCE_LENGTH = 21  #단어 개수 중간 값 사용
ces(text_sequences, maxlen=MAX_SEQUENCE_LENGTH, padding='post')
Shape of data:  (4846, 21)
```

왜 평균 대신 **중간값**?

일부 데이터의 길이가 지나치게 길면 평균이 급격히 올라갈 수 있기 때문에

- 모 델

머신러닝


벡터화
모델링
평가
TF-IDF
CounterVectorizer
로지스틱회귀모델
랜덤포레스트
서포트벡터머신
accyracy
precision
recall
f1-score

- 모 델 (머신러닝)

TF-IDF

TF(특정 단어가 하나의 데이터에서 등장 횟수)
X
I(역수)+DF(특정 단어가 여러 데이터에서 등장 횟수) -> IDF

```
vectorizer = TfidfVectorizer(min_df = 0.0, analyzer="word"
                             , sublinear_tf=True, ngram_range=(1,3), max_features=10000)
```

min_df	df값이 0.0보다 적게 나오면 제거
analyzer	"word" , 단어를 하나의 단위로
sublinear_tf	스무딩여부 TF->1 + ln(TF)
ngram_range(1,3)	단어 묶음을 1~3개
max_features	1000 벡터의 최대 길이

- 모 델 (머신러닝)

TF-IDF

TF(특정 단어가 하나의 데이터에서 등장 횟수)
X
I(역수)+DF(특정 단어가 여러 데이터에서 등장 횟수) -> IDF

```
vectorizer = TfidfVectorizer(min_df = 0.0, analyzer="word"
                             , sublinear_tf=True, ngram_range=(1,3), max_features=10000)
```

```
X = vectorizer.fit_transform(text).toarray()
y = np.array(sentiments)
```

```
X.shape
```

```
(4845, 10000)
```

- 모 델 (머신러닝)

TF-IDF

SVM

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear')
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_eval)
print("Accuracy: %f" % classifier.score(X_eval, y_eval))
print(classification_report(y_eval,y_pred))
```

```
Accuracy: 0.753354
              precision    recall  f1-score   support

    negative       0.78      0.45      0.57       113
     neutral       0.75      0.93      0.83       567
    positive       0.75      0.52      0.61       289

    accuracy                           0.75       969
   macro avg       0.76      0.63      0.67       969
weighted avg       0.75      0.75      0.74       969
```

LogisticRegression

```
lgs = LogisticRegression(class_weight='balanced')
lgs.fit(X_train, y_train)
y_pred = lgs.predict(X_eval)
print("Accuracy: %f" % lgs.score(X_eval, y_eval))
print(classification_report(y_eval,y_pred))
```

```
Accuracy: 0.753354
              precision    recall  f1-score   support

    negative       0.64      0.60      0.62       113
     neutral       0.79      0.86      0.83       567
    positive       0.70      0.60      0.65       289

    accuracy                           0.75       969
   macro avg       0.71      0.69      0.70       969
weighted avg       0.75      0.75      0.75       969
```

둘의 Accuracy는 비슷하나,

SVM의 neurtral recall이 높으며

LogisticRegression은 negative와 positive를 SVM보다 잘 맞춘다(f1-score 높음)

- 모 델 (머신러닝)

TF-IDF

RandomForest

```
from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier(n_estimators = 1000)

forest.fit( X_train, y_train )
y_pred = forest.predict(X_eval)
print("Accuracy: %f" % forest.score(X_eval, y_eval))
print(classification_report(y_eval,y_pred))
```

```
Accuracy: 0.734778
              precision    recall  f1-score   support

    negative       0.73      0.32      0.44       113
     neutral       0.73      0.96      0.83       567
    positive       0.76      0.45      0.57       289

    accuracy                           0.73       969
   macro avg       0.74      0.58      0.61       969
weighted avg       0.74      0.73      0.71       969
```

- 모 델 (머신러닝)

CounterVectorizer

텍스트에서 횟수를 측정한 뒤 벡터로 만듬

이 때 횟수 기준을 word 로 잡음

```
vectorizer = CountVectorizer(analyzer = "word", max_features = 1000)

train_data_features = vectorizer.fit_transform(text)
```

```
train_data_features.shape
```

```
(4845, 1000)
```

- 모 델 (머신러닝)

CounterVectorizer

RandomForest

```
from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier(n_estimators = 1000)

forest.fit( train_input, train_label )
y_pred = forest.predict(eval_input)
print("Accuracy: %f" % forest.score(eval_input, eval_label))
print(classification_report(eval_label,y_pred))
```

```
Accuracy: 0.730650
              precision    recall  f1-score   support

    negative       0.63      0.42      0.51       113
     neutral       0.75      0.89      0.82       567
    positive       0.69      0.54      0.61       289

    accuracy                           0.73       969
   macro avg       0.69      0.62      0.64       969
weighted avg       0.72      0.73      0.72       969
```

LogisticRegression

```
lgs = LogisticRegression(class_weight='balanced')
lgs.fit(train_input, train_label)
y_pred = lgs.predict(eval_input)
print("Accuracy: %f" % lgs.score(eval_input, eval_label))
print(classification_report(eval_label,y_pred))
```

```
Accuracy: 0.710010
              precision    recall  f1-score   support

    negative       0.49      0.62      0.55       113
     neutral       0.81      0.77      0.78       567
    positive       0.64      0.64      0.64       289

    accuracy                           0.71       969
   macro avg       0.65      0.67      0.66       969
weighted avg       0.72      0.71      0.71       969
```

- 모 델 (머신러닝)

CounterVectorizer

SVM

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear')
classifier.fit(train_input, train_label)
y_pred = classifier.predict(eval_input)
print("Accuracy: %f" % classifier.score(eval_input, eval_label))
print(classification_report(eval_label,y_pred))
```

```
Accuracy: 0.706914
              precision    recall  f1-score   support

    negative       0.53      0.46      0.49       113
     neutral       0.75      0.84      0.79       567
    positive       0.66      0.55      0.60       289

    accuracy                           0.71       969
   macro avg       0.65      0.62      0.63       969
weighted avg       0.70      0.71      0.70       969
```

- 모 델

딥러닝

전처리

label 원핫인코딩

```
array([[0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       ...,
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.]]
```

0 -> negative

1 -> neutral

2 -> positive

모델링

RNN

CNN+RNN

CNN

output = softmax 사용

평가

predict 결과

[0.09909477, 0.8757712 , 0.02513401]

accuracy

loss

- 모 델 (딥러닝)

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
hist=model.fit(X_train, y_train, batch_size=128, epochs=100, validation_data=(X_test, y_test),callbacks=[mc])
```

" 모든 모델의 loss, optimizer, batch_size, epochs value는 같음 "

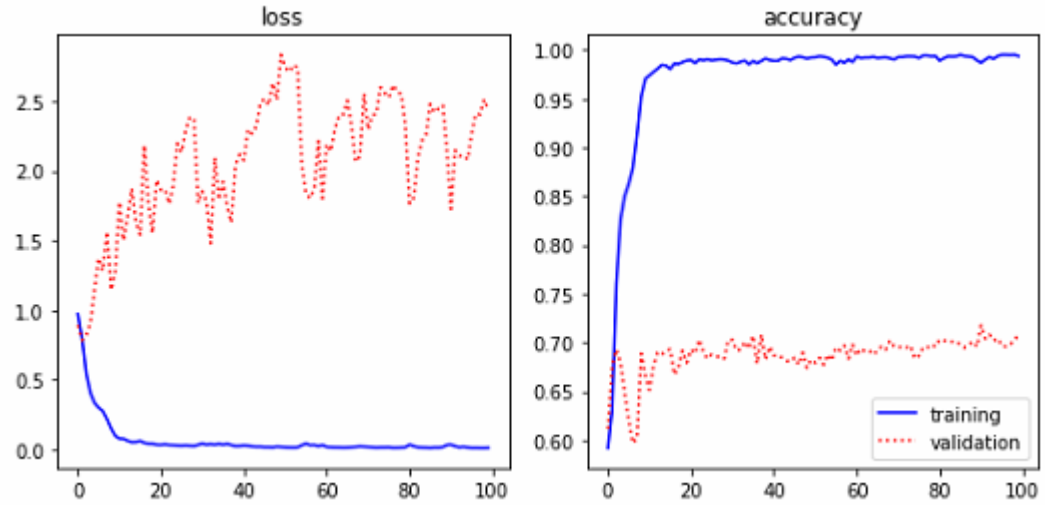
RNN

```
model = Sequential()
model.add(Embedding(VOCAB_SIZE, 100))
model.add(LSTM(100,activation='tanh'))
model.add(Dense(3, activation='softmax'))
```

Word Embedding

LSTM

Dense(Softmax함수사용)

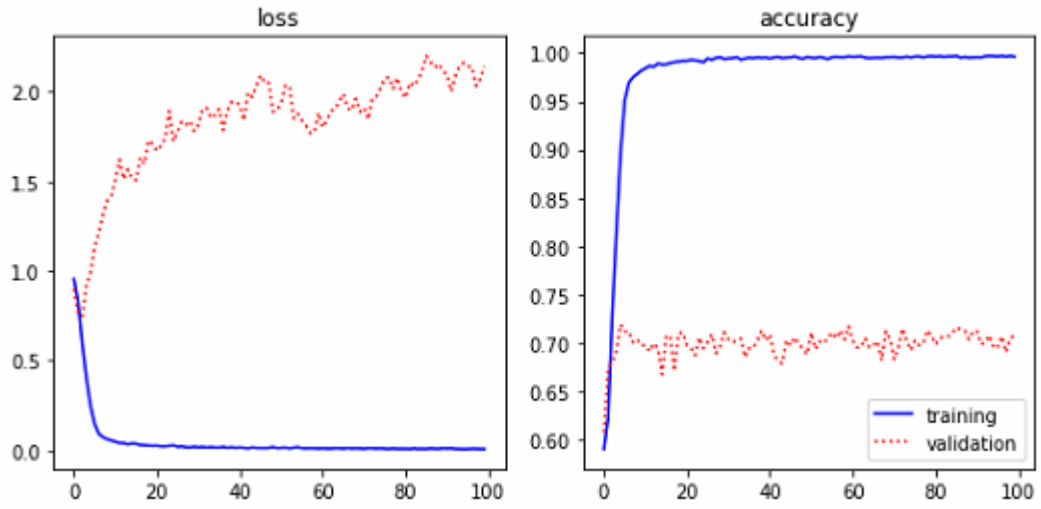



```
970/970 [==============================] - 1s 1ms/step

테스트 loss: 1.7265
테스트 정확도: 0.7175
```

- 모 델 (딥러닝)

RNN+CNN

```
model = Sequential()
model.add(Embedding(VOCAB_SIZE, 100))
model.add(Dropout(0.3))
model.add(Conv1D(128,3, padding= 'valid', activation='relu'))
model.add(MaxPooling1D(pool_size=4))
model.add(LSTM(56,activation='tanh'))
model.add(Dense(3,activation='softmax'))
```



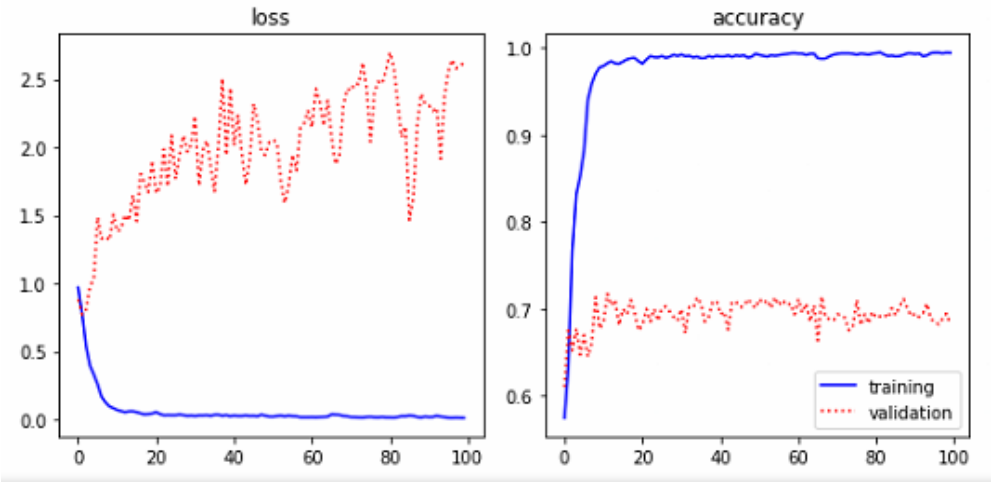

```
970/970 [==============================] - 1s 1ms/step

테스트 loss: 0.9917
테스트 정확도: 0.7196
```

- Word Embedding
- Dropout
- Convolution1D
- MaxPooling
- LSTM
- Dense(Softmax함수사용)

- 모 델 (딥러닝)

CNN(Conv1D)

```
model = Sequential()
model.add(Embedding(VOCAB_SIZE, 100))
model.add(Dropout(0.3))
model.add(Conv1D(256,3, padding= 'valid', activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
```




```
970/970 [==============================] - 1s 1ms/step

테스트 loss: 1.2098
테스트 정확도: 0.7402
```

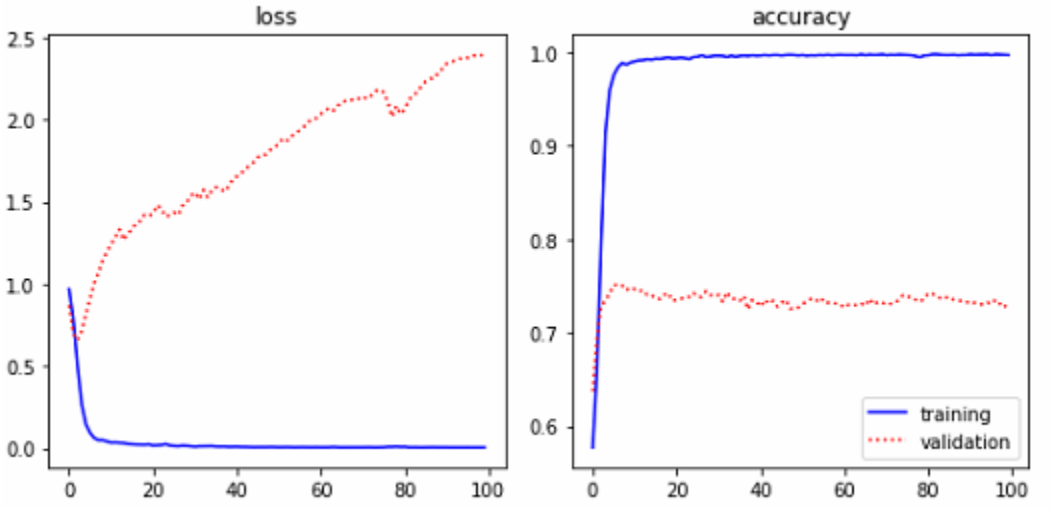
CNN만 사용 " acc 증가 "

- Word Embedding
- Dropout
- Convolution1D
- GlobalMaxPooling1D
- Dense(relu 함수사용)
- Dropout
- Dense(Softmax함수사용)

- 모 델 (딥러닝)

CNN(Conv2D)

```
model = Sequential()

model.add(Embedding(input_dim = num_features, output_dim = embedding_dimension,input_length = sequence_length))
model.add(Reshape((sequence_length, embedding_dimension, 1), input_shape = (sequence_length, embedding_dimension)))
model.add(Conv2D(filters = 128, kernel_size = (5, embedding_dimension), strides = (1,1), padding = 'valid'))
model.add(GlobalMaxPooling2D())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(3))
model.add(Activation('softmax'))
```



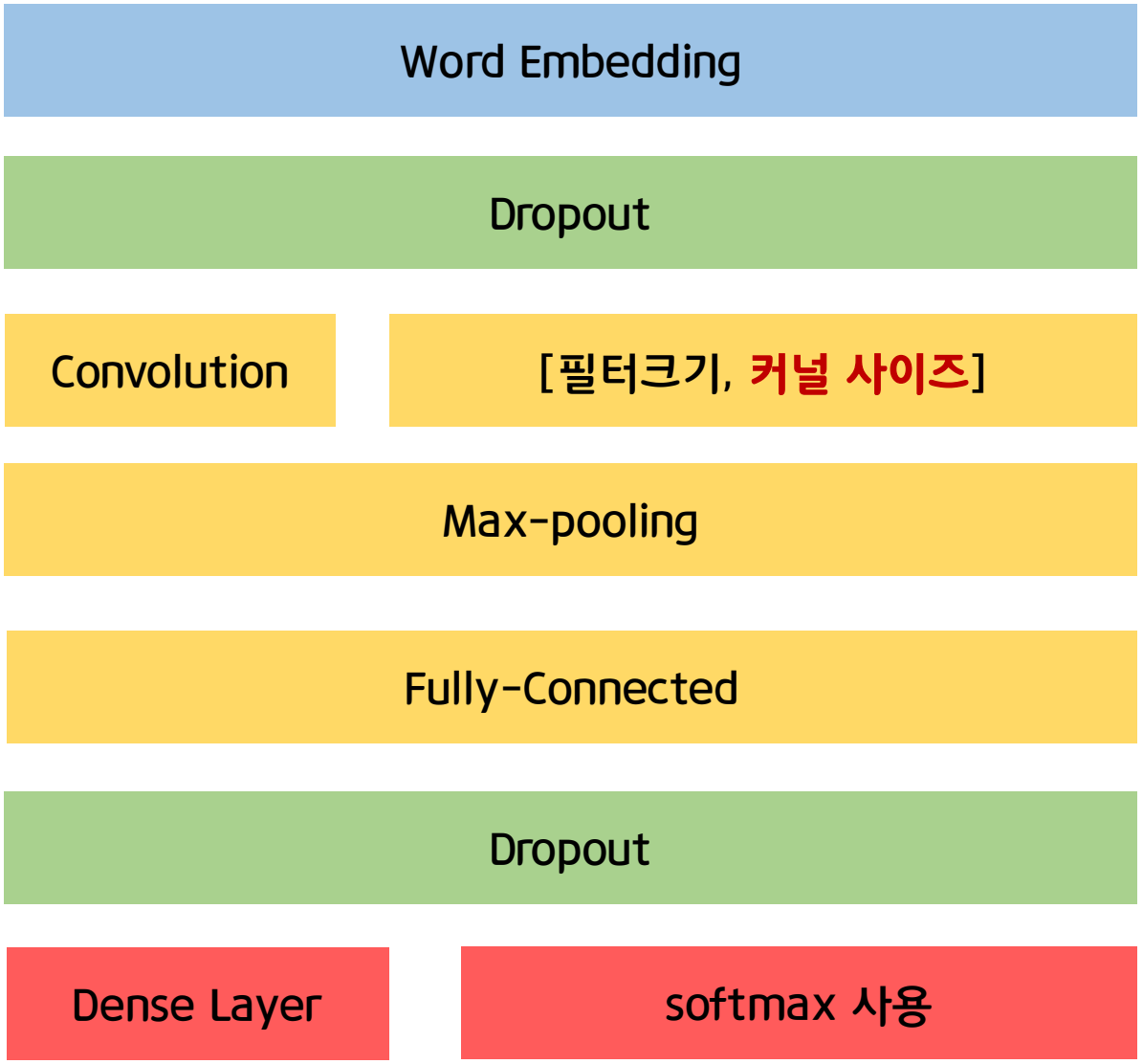

```
970/970 [==============================] - 1s 1ms/step

테스트 loss: 1.0050
테스트 정확도: 0.7515
```

같은 구조에 Conv1D대신 2를 사용

" loss 감소, acc 증가 "

- 모 델 (딥러닝)

CNN(Yoonkim박사의 CNN for sentence classification)



합성곱 **레이어 3개**

커널사이즈= [3,4,5]

이는 3,4,5 gram처럼

다양한 각도에서 문장을 보는 효과!

- 모 델 (딥러닝)

CNN(Yoonkim박사의 CNN for sentence classification)

" Conv1D를 3개 쌓음 "

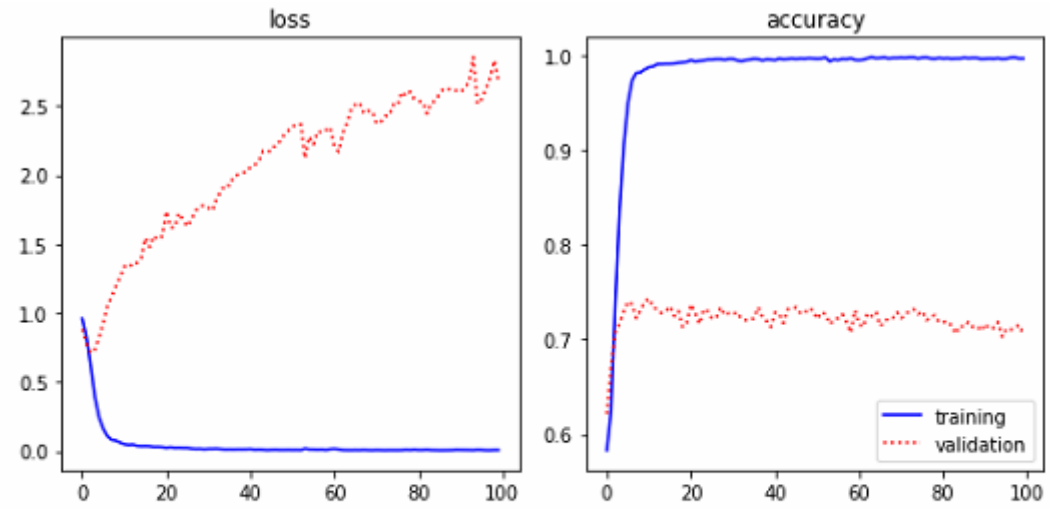
```
filter_sizes = [3, 4, 5]

def convolution():
    inn = Input(shape = (sequence_length, embedding_dimension))
    convolutions = []
    # we conduct three convolutions & poolings then concatenate them.
    for fs in filter_sizes:
        conv = Conv1D(filters = 128, kernel_size = fs, activation=tf.nn.relu , padding = "valid")(inn)
        maxpool = GlobalMaxPooling1D()(conv)
        convolutions.append(maxpool)

    outt = concatenate(convolutions)
    model = Model(inputs = inn, outputs = outt)

    return model

def imdb_cnn_4():

    model = Sequential()
    model.add(Embedding(input_dim = VOCAB_SIZE, output_dim = embedding_dimension, input_length = sequence_length))
    # call convolution method defined above
    model.add(convolution())

    model.add(Flatten())
    model.add(Dense(50))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(3))
    model.add(Activation('softmax'))
```

<모델 구현 코드>

- 모 델 (딥러닝)

CNN(Yoonkim박사의 CNN for sentence classification)

<결과>



Conv1D 1개 사용했을 때 보다 정확도 상승

- 모 델 (딥러닝)

CNN(Yoonkim박사의 CNN for sentence classification)

" Conv2D를 3개 쌓음 "

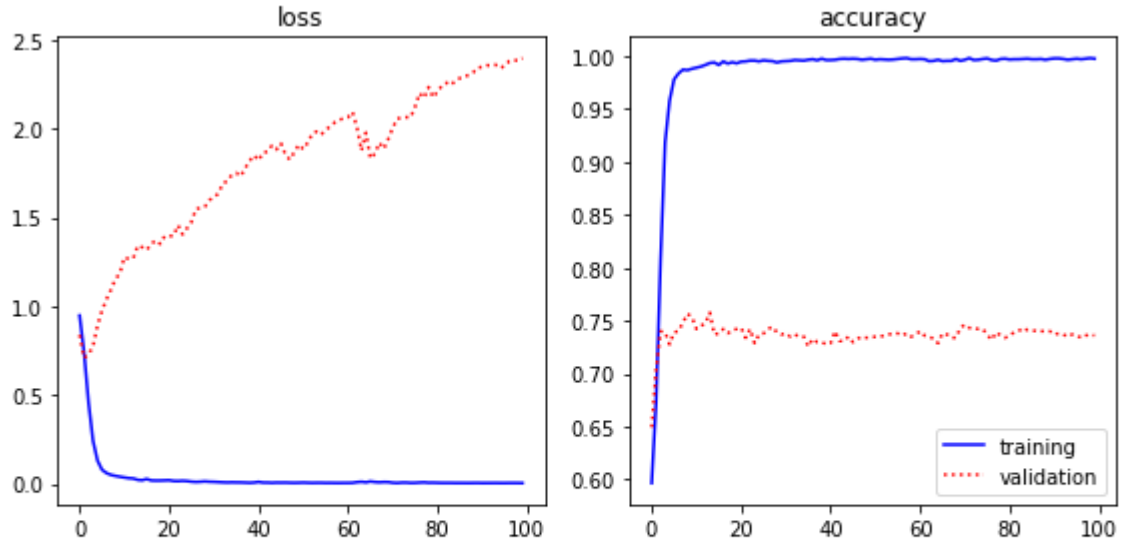
```
filter_sizes = [3, 4, 5]

def convolution():
    inn = Input(shape = (sequence_length, embedding_dimension, 1))
    convolutions = []
    # we conduct three convolutions & poolings then concatenate them.
    for fs in filter_sizes:
        conv = Conv2D(filters = 128, kernel_size = (fs, embedding_dimension), strides = 1, padding = "valid")(inn)
        maxpool = GlobalMaxPooling2D()(conv)
        convolutions.append(maxpool)

    outt = concatenate(convolutions)
    model = Model(inputs = inn, outputs = outt)

    return model

def imdb_cnn_3():

    model = Sequential()
    model.add(Embedding(input_dim = VOCAB_SIZE, output_dim = embedding_dimension, input_length = sequence_length))
    model.add(Reshape((sequence_length, embedding_dimension, 1), input_shape = (sequence_length, embedding_dimension)))

    # call convolution method defined above
    model.add(convolution())

    model.add(Flatten())
    model.add(Dense(50))
    model.add(Activation('relu'))
    model.add(Dropout(0.3))
    model.add(Dense(3))
    model.add(Activation('softmax'))
```

<모델 구현 코드>

- 모 델 (딥러닝)

CNN(Yoonkim박사의 CNN for sentence classification)

<결과>



Conv2D 3개의 경우

" acc가 제일 높음 "

- 결과분석

머신러닝

가장 좋은 성능을 내는 모델은

TF-IDF로 벡터화 후 학습한 LogisticRegression

모델	정확도
로지스틱회귀(TF-IDF)	0.75335
서포트벡터머신(TF-IDF)	0.75335
랜덤포레스트(TF-IDF)	0.73478
로지스틱회귀(CountVt)	0.71001
서포트벡터머신(CountVt)	0.70691
랜덤포레스트(CountVt)	0.73065

- 결과분석

딥러닝

가장 좋은 성능을 내는 모델은

Conv2D를 3개 쌓은 Neural Network

모델	정확도
CNN2D 3개	0.7577
CNN1D 3개	0.7433
CNN2D 1개	0.7515
CNN1D 1개	0.7402
CNN1D+RNN	0.7196
RNN	0.7175

- 기대 효과

'기대효과'  가지

실시간 분류

경제 뉴스의
실시간 감정 분류

경제 전망 예측

경제 위기
경제 호황 예측

주가 예측

상승 종목
하락 종목 예측

Thank you