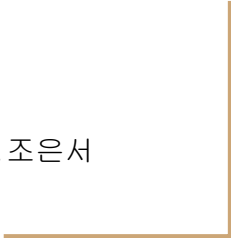


# 가짜뉴스 분류 알고리즘

ESAA 0B - 2조

박성희, 예지혜, 김연휘, 송민경, 조은서



# 목차

1. 배경
  - 1.1. 주제 소개
  - 1.2. 데이터 설명
2. EDA
3. Modeling
  - 3.1. 자연어 처리 과정 소개
  - 3.2. 사용한 모델 소개
  - 3.3. 최종 선정 모델
4. 결론 및 한계점



# 1. 배경

1.1. 주제 소개

1.2. 데이터 설명



# 1.1. 주제 소개

**"데이터, 문화가 되다 : League1"**


**AI야, 진짜 뉴스를 찾아줘!**

금융 | NH투자증권 | 텍스트 분류 | Accuracy + Time | 중복 참가 불가, 대학 재학생만 참가 가능

💰 상금 : 총 5,000만원(League1,2 통합)

🕒 2020.11.23 ~ 2020.12.31 17:59 [+ Google Calendar](#)

👥 551팀 📅 마감





참여중

- 주제 : **가짜 뉴스와 진짜 정보를 구별하는 자연어 처리 알고리즘** 생성
- 내용 : 2020년 1월부터 6월까지의 뉴스 타이틀과 내용을 분석하여 해당 내용이 가짜뉴스인지 아닌지 분류하는 알고리즘을 생성. 전처리부터 모델링까지 자연어 처리의 전 과정을 경험.
- 주제 선정 이유 : 학기 중 세션을 통해 가볍게 학습한 텍스트 분석이라는 새로운 분야에 도전해보고자 선정

# 1.1. 주제 소개

## 기사 내 광고 편집 심의 위반 건 수

※ 출처 : 한국광고자율심의기구 (2019)



광고가 혼재된 뉴스 기사의 증가 추세



뉴스 본문에서 가짜 뉴스를 선별해  
진짜 정보만을 제공하는 알고리즘의 필요성



# 1. 배경

1.1. 주제 소개

1.2. 데이터 설명



## 1.2. 데이터 설명

### 학습 데이터

고객에게 제공되는 뉴스 원문 데이터 (118,745건)

- 제공 기간 : '20년 1월 ~ 6월
- 파일명 : news\_train.csv

No	컬럼명	컬럼 설명	예시
1	N_ID	뉴스 Index 번호	33938
2	DATE	뉴스 발행 날짜	20200518
3	TITLE	뉴스 제목	'디지털 적폐' 공인인증서 퇴출 코앞... 20일 본회의 오늘 듯
4	CONTENT	뉴스 내용	공인인증서가 21년 만에 역사 속으로 사라질 전망이다.
5	ORD	뉴스 내용 순서	1
6	INFO	진짜뉴스유무 1: 가짜정보/ 0: 진짜뉴스	1

Target (Y)

### 테스트 데이터

개발한 알고리즘 검증을 위한 문제지 (142,565건)

- 정답 기록을 위한 질문 항목 (info) 포함
- 파일명 : news\_test.csv

No	컬럼명	컬럼 설명	예시
1	N_ID	뉴스 Index 번호	NEWS04938
2	DATE	뉴스 발행 날짜	20200318
3	TITLE	뉴스 제목	서울 아파트값 2주째 보합... '경기·인천·세종'도 상승폭 축소
4	CONTENT	뉴스 내용	경기도는 전주(0.40%) 대비 0.28% 올라 상승폭이 뚝 떨어 졌다. 신분당선...
5	ORD	뉴스 내용 순서	3
7	ID	N_ID + ORD 로 구성된 고유 ID	NEWS04938_3

## 1.2. 데이터 설명

각 문장들에 대한 가짜 정보 / 진짜 정보 여부

02580번 신문기사    02580번 신문기사의 문장들(총 4문장)    02580번 신문기사에 대한 문장들의 순서

	n_id	date	title	content	ord	info
0	NEWS02580	20200605	[마감]코스닥 기관 678억 순매도	[이데일리 MARKETPOINT]15.32 현재 코스닥 기관 678억 순매도	1	0
1	NEWS02580	20200605	[마감]코스닥 기관 678억 순매도	"실적기반" 저가에 매집해야 할 8월 급등유망주 TOP 5 전격공개	2	1
2	NEWS02580	20200605	[마감]코스닥 기관 678억 순매도	하이스탁론, 선취수수료 없는 월 0.4% 최저금리 상품 출시	3	1
3	NEWS02580	20200605	[마감]코스닥 기관 678억 순매도	종합 경제정보 미디어 이데일리 - 무단전재 & 재배포 금지	4	0
4	NEWS09727	20200626	롯데 공영 등 7개 TV 홈쇼핑들, 동행세일 동참	전국적인 소비 품 조성에 기여할 예정	1	0
5	NEWS09727	20200626	롯데 공영 등 7개 TV 홈쇼핑들, 동행세일 동참	[이데일리 권오석 기자] 중소벤처기업부(이하 중기부)는 대한민국 동행세일에 7개 T...	2	0
6	NEWS09727	20200626	롯데 공영 등 7개 TV 홈쇼핑들, 동행세일 동참	대한민국 동행세일은 라이브 커머스, 언택트 콘서트, O2O 행사 연계 등 비대면 이라...	3	0
7	NEWS09727	20200626	롯데 공영 등 7개 TV 홈쇼핑들, 동행세일 동참	6개 권역에서의 현장행사와 온 오프라인 판촉, TV홈쇼핑 등 연계행사를 통해 소비심...	4	0
8	NEWS09727	20200626	롯데 공영 등 7개 TV 홈쇼핑들, 동행세일 동참	이번 동행세일에서는 롯데 공영 CJ 현대 GS NS 홈쇼핑 등 7개 TV 홈쇼핑	5	0

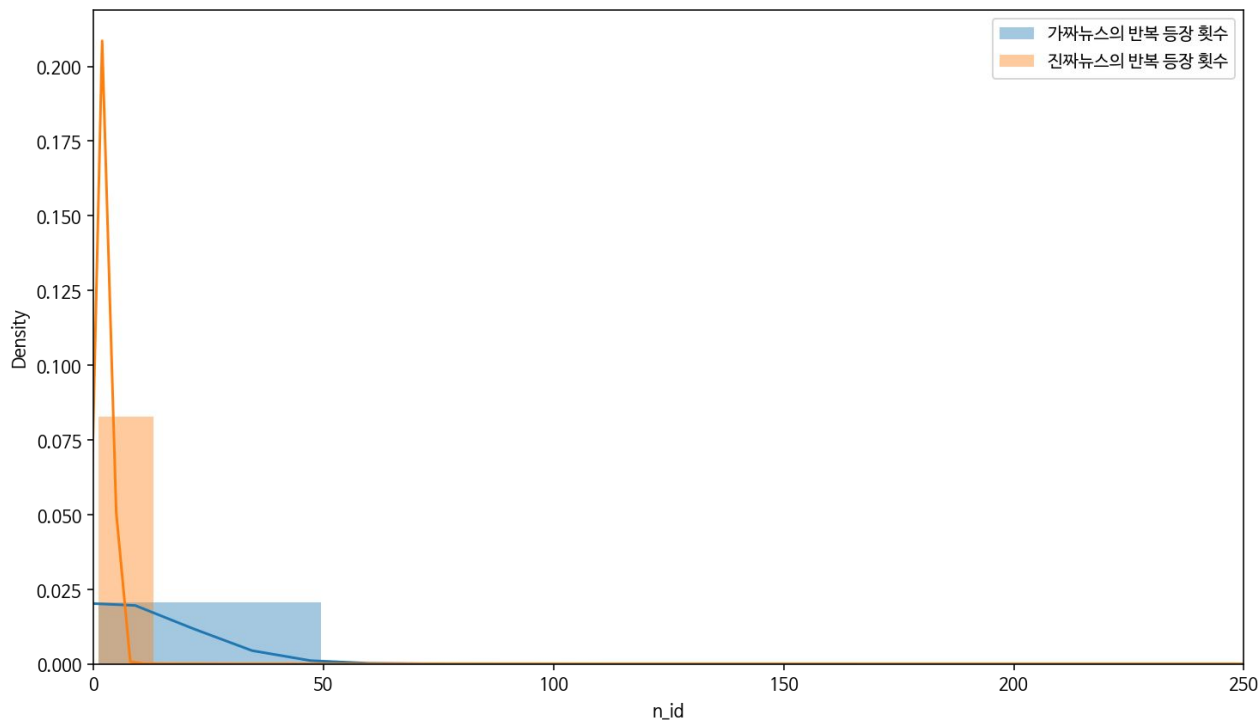
09727번 신문기사    09727번 신문기사의 문장들    09727번 신문기사에 대한 문장들의 순서

각 문장들에 대한 가짜 정보 / 진짜 정보 여부



## 2. EDA

## 2. EDA



가짜뉴스에 해당하는 내용을  
읽어보면 **광고성 문구**가 많이  
분포해있는 것을 파악할 수 있음.

해당 기간 동안 동일한 광고가  
등장했을 가능성이 높아 가짜뉴스와  
진짜뉴스의 반복 등장 횟수를 비교해  
본 결과,

가짜뉴스의 반복 등장 횟수가 훨씬  
높은 경향을 발견할 수 있음.

## 2. EDA



'종목': 9140, '가능': 7288, '공개': 5495, '한국': 5483, '추천': 5464, '주': 5061, '목표': 4765, '상한': 4740, '이상': 4468, '무료': 4168, '주식': 4091, '바이오': 3696, '금리': 3504, '테마': 3498, '최저': 3478, '방': 3461, '탁론': 3295, '카톡': 3102, '실적': 3079, '이용': 3029, '투자': 2997, '신용': 2967, '수익': 2962, '미수': 2939, '환': 2926, '혜주': 2912, '급등': 2873, '오늘': 2788, '대장': 2784, '당장': 2778, '매집': 2743, '젠': 2559, '줄': 2526, '로': 2520, '배': 2503, '효과': 2479, '수익률': 2475, '돈': 2474, '재료': 2474, '평가': 2463, '스': 2455, '업계': 2449, '지금': 2437, '시대': 2435, '라면': 2432, '연결': 2428, '인터넷': 2427, '레버리지': 2426, '소비자': 2423, '영웅': 2422, '박주': 2421, '똑똑해진': 2420, '소형차': 2420, '대중': 2420, '역대': 2340, '최종': 2302, '거래': 2296, '내일': 2293, '핵심': 2293, '클릭': 2277, '모집': 2261, '분': 2238, '끝': 2208, '다시': 2207, '책임': 2198, '여기': 2197, '주식시장': 2197, '도전': 2195, '사면': 2190, '정치': 2190, '역사': 2187, '니': 2185, '량': 2183, '가도': 2181, '인맥': 2180, '안집': 2180, '정부': 2179, '정책': 2080, '긴급': 2077, '코로나': 1861, '확인': 1529, '상반기': 1487, '관련': 1416, '수수료': 1375, '바로': 1339, '예상': 1176, '최대': 1147, '공략': 1082, '체험': 973, '바이러스': 934, '추가': 914, '부터': 911, '임박': 855, '것': 842, '직전': 842, '명': 838, '장주': 729, '유망': 721, '수': 698

가짜 뉴스의 경우, '종목', '주', '상한', '주식', '테마' 등 **주식 관련 단어**, '추천', '무료' 등 **광고성 단어**가 자주 등장

## 2. EDA



코로나: 5513, '기자': 3447, '경제': 3173, '한국': 3032,  
'위해': 2920, '사업': 2791, '기업': 2781, '통해': 2588,  
'지원': 2508, '관련': 2482, '지역': 2477, '투자': 2396,  
'지난': 2334, '대한': 2230, '시장': 2200, '최근': 2174,  
'거래': 2157, '금지': 2151, '정부': 2109, '기준': 2020,  
'이번': 1996, '현재': 1994, '대비': 1992, '배포': 1978,  
'무단': 1945, '서울': 1864, '미국': 1861, '대해': 1829,  
'개발': 1795, '해럴드경제': 1752, '계획': 1722, '진행':  
1657, '제공': 1655, '국내': 1636, '종목': 1636, '이후': 610,  
'금융': 1603, '기술': 1529, '올해': 1524, '지난해': 516,  
'예정': 1499, '대표': 1472, '기록': 1470, '기사': 1459, '경우':  
1445, '상황': 1439, '서비스': 1434, '상승': 1421, '중국':  
1418, '정보': 1408, '확대': 1398, '뉴스': 1377, '증가':  
1369, '산업': 1367, '글로벌': 1351,  
'대상': 1349, '진자': 1332, '종합': 1324, '기관': 1311,  
'운영': 1307, '마스크': 1299, '해당': 1288, '이상': 1279,  
'추가': 1275, '증권': 1269, '확산': 1262, '발생': 1256

진짜 뉴스의 경우, ‘코로나’, ‘기자’, ‘경제’, ‘한국’ 등 **현 이슈에 걸맞는 뉴스에서 자주 등장할 만한 단어들이 많이 포함되어 있음.**



## 3. Modeling

3.1. 자연어 처리 과정 소개

3.2. 사용한 모델 소개

3.3. 최종 선정 모델



## 3.1.1. Tokenizing 및 불용어 제거

주어진 코퍼스(corpus)에서 **토큰(token)**이라 불리는 단위로 나누는 작업을 **토큰화(tokenization)**라고 함  
영어는 띄어쓰기 단위로 토큰화해도 충분히 모델링이 가능하지만, 한국어는 교착어이기 때문에 **형태소 단위로** 토큰화  
이때 **'형태소 분석기'**를 사용

형태소 분석기를 통해 한국어를 형태소 단위로 쪼갤 수 있고, 각 단어의 품사를 태깅 가능  
현재, Mecab, Komoran, Okt 등 다양한 형태소 분석기를 사용

이 단계에서, 분석에 도움이 되지 않는 특수문자, 조사 등은 불용어로 설정하여 제거 => **불용어(stopwords) 제거**

\*이과정은 데이터와 분석자의 재량에 따라 유동적\*

```
print(komoran.morphs(text)) #텍스트에서 형태소 반환  
print(komoran.nouns(text)) #텍스트에서 명사 반환  
print(komoran.pos(text)) #텍스트에서 품사 정보 부착하여 반환
```

```
['ESAA', '2', '조', '프로젝트', '파이팅', '힘내', '요']
```

```
['프로젝트', '파이팅']
```

```
[('ESAA', 'SL'), ('2', 'SN'), ('조', 'NR'), ('프로젝트', 'NNP'), ('파이팅', 'NNP'), ('힘내', 'VY'), ('요', 'EC')]
```

## 3.1.2. Vectorization

NLP를 컴퓨터가 이해할 수 있게 **수치로 바꾸는 작업**을 의미

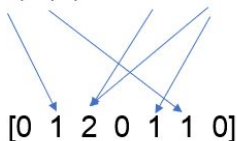
- **One Hot Encoding** : 해당 단어가 존재하면 1, 그렇지 않으면 모두 0으로 표시하는 방법
- **Count vectorization** : 각 문장이 갖고 있는 토큰의 **count**를 **기반**으로 문장을 벡터화하는 방법
- **Tfidf** : 등장 횟수도 많고 문서 분별력 있는 단어들을 점수화하여 벡터화하는 방법

vocabulary

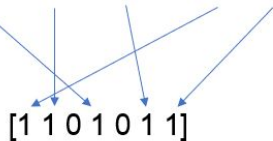
['같이', '자연어', '정말', '즐거운', '즐거워', '처리', '해보자']

Count vectorization

'자연어 처리 는 정말 정말 즐거워'



'즐거운 자연어 처리 다 같이 해보자 '

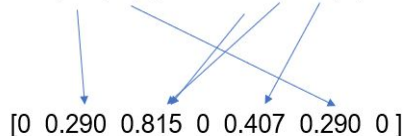


vocabulary

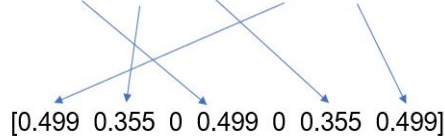
['같이', '자연어', '정말', '즐거운', '즐거워', '처리', '해보자']

Tfidf

'자연어 처리 는 정말 정말 즐거워'



'즐거운 자연어 처리 다 같이 해보자 '



# 3.1.3. Embedding

비슷한 의미를 내포하고 있는 토큰들은 서로 **가깝게**, 그렇지 않은 토큰들은 서로 멀리 뿌리도록 하는 것이 임베딩의 목적

임베딩 또한 **하나의 모델**을 의미하며 훈련이 필요  
데이터가 충분하고 시간이 많으면 소지한 데이터에 특화된 임베딩 모델을 학습 가능

word2vec, glove, fasttext을 비롯한 다양한 임베딩 기법들이 존재

→ 생성된 **embedding matrix**는 사용하는 모델의 **가중치로 사용**



1 embedding\_matrix

```
array([[ -0.046875,  -0.03637695,  0.09423828, ..., -0.00408936,
         0.1328125,   0.17773438],
       [-0.06494141, -0.04272461,  0.16601562, ...,  0.02539062,
         0.10986328,  0.29882812],
       [ 0.0037384,  -0.09228516,  0.13867188, ...,  0.01037598,
         0.14160156,  0.24609375],
       ...,
       [ 0.,          0.,          0.,          ...,  0.,          ],
       [ 0.,          0.,          ],
       [ 0.,          0.,          0.,          ...,  0.,          ],
       [ 0.,          0.,          ],
       [ 0.,          0.,          0.,          ...,  0.,          ],
       [ 0.,          0.,          ]])
```





## 3. Modeling

3.1. 자연어 처리 과정 소개

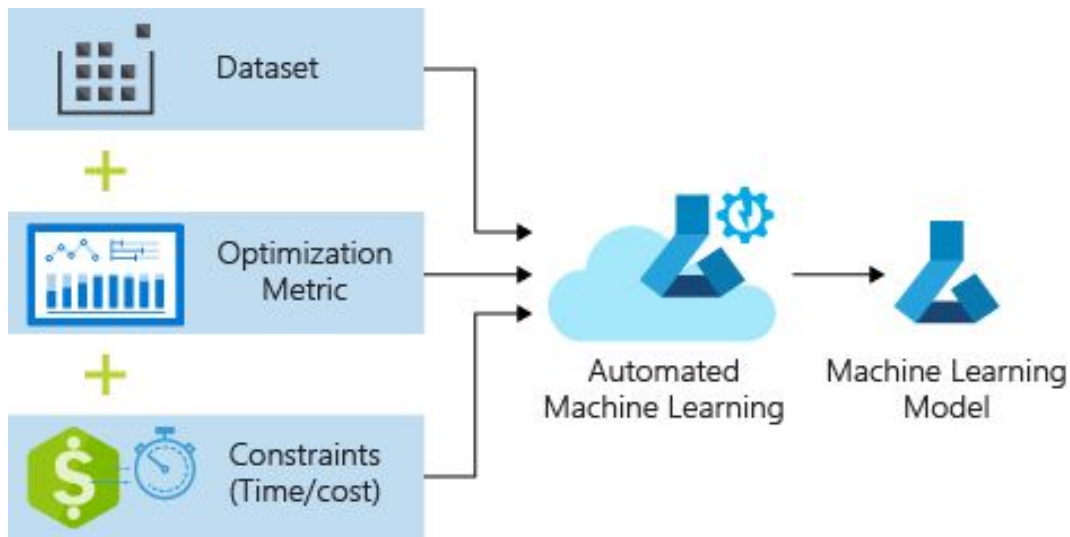
3.2. 사용한 모델 소개

3.3. 최종 선정 모델



## 3.2.1. AutoML (Auto Machine Learning)

정의 : 자동화된 머신러닝을 통해 데이터 전처리부터 알고리즘 선택 및 튜닝까지 모델 개발자의 개입을 최소화한 모델



## 3.2.1. AutoML 적용 모델링

```
input_node = ak.TextInput()  
output_node = ak.TextToIntSequence()(input_node)  
output_node = ak.Embedding()(output_node)  
output_node = ak.ConvBlock(separable=True)(output_node)  
output_node = ak.ClassificationHead()(output_node)  
clf = ak.AutoModel(  
    inputs=input_node,  
    outputs=output_node,  
    overwrite=True,  
    max_trials=5  
)  
clf.fit(X, Y, epochs=5)  
model = clf.export_model()  
model.summary()
```

tokenizer를 사용하지 않고 텍스트 통째로 모델링에 사용

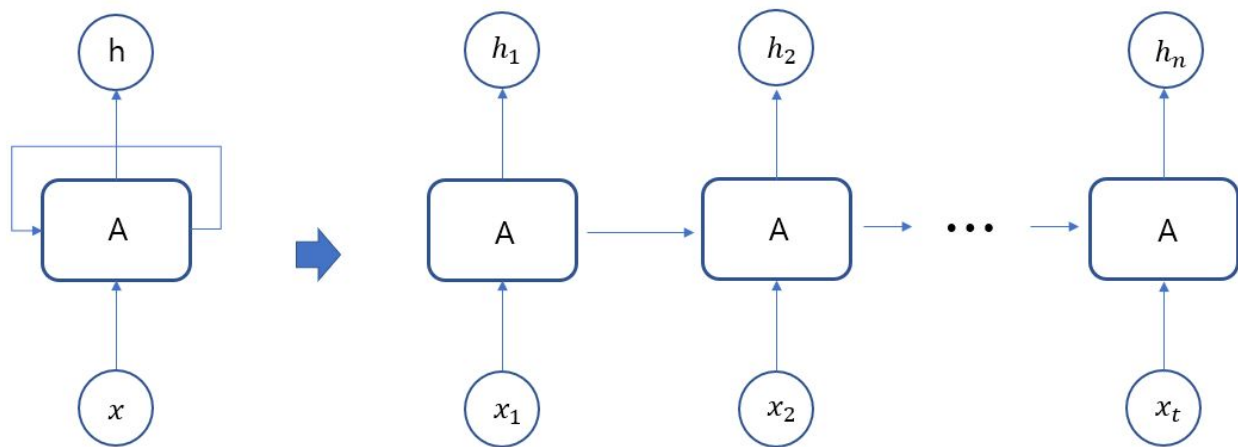
텍스트를 정수형으로

정확도: 0.9458

## 3.2.2. RNN(Recurrent Neural Network, 순환신경망)

정의: 은닉 계층 안에 하나 이상의 순환 계층을 갖는 신경망

순서가 있는 데이터에 주로 사용되며, 해당 데이터를 입력으로, 하나의 네트워크를 통해서 순서대로 출력을 얻음



## 3.2.2. RNN 적용 모델링

```
X_data = train['content']
y_train = train['info']
tokenizer = Tokenizer(num_words = total_cnt - rare_cnt + 1)
tokenizer.fit_on_texts(X_data) # 5169개의 행을 가진 x의 각 행에 토큰화를
수행
sequences = tokenizer.texts_to_sequences(X_data) # 단어를 숫자값, 인덱스로
변환하여 저장
word_to_index = tokenizer.word_index
vocab_size = len(word_to_index) + 1

from tensorflow.keras.layers import SimpleRNN, Embedding, Dense
from tensorflow.keras.models import Sequential

model = Sequential()
model.add(Embedding(vocab_size, 32)) # 임베딩 벡터의 차원은 32
model.add(SimpleRNN(32)) # RNN 셀의 hidden_size는 32
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['acc'])

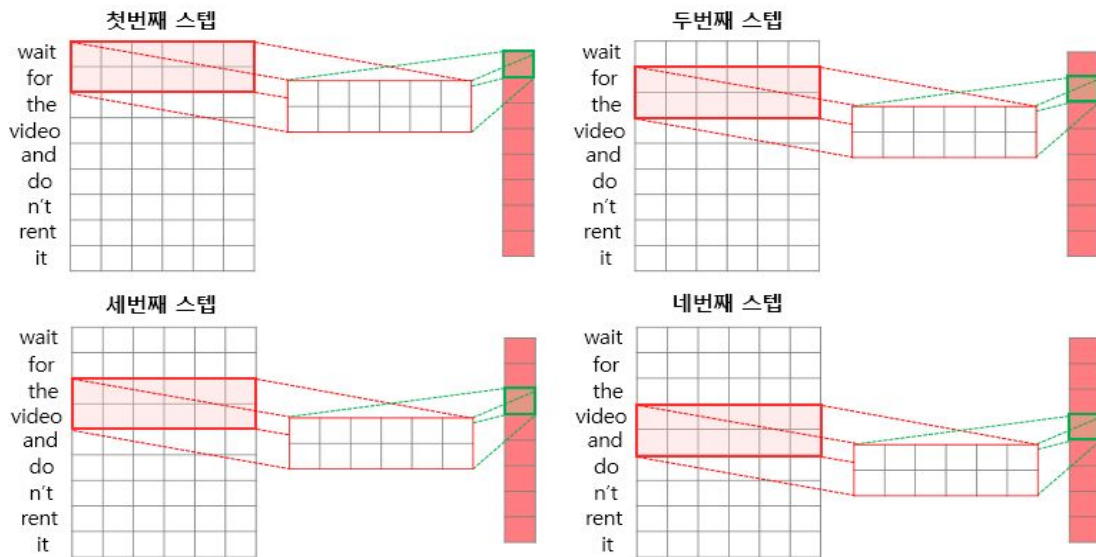
history = model.fit(X_train, y_train, epochs=3, batch_size=64,
validation_split=0.2)
```

정확도: 0.9626

## 3.2.3. 1D CNN (1D Convolutional Neural Networks)

정의: 1차원 합성곱

1D CNN에서 커널의 너비는 문장 행렬에서의 임베딩 벡터의 차원과 동일하게 설정됨  
따라서, 높이 사이즈만을 명명하여 해당 커널의 사이즈라고 간주



## 3.2.3. 1D CNN 적용 모델링

```
def text_preprocessing(text_list):
```

```
    stopwords = ['을', '를', '이', '가', '은', '는', 'null']
```

```
    tokenizer = Mecab()
```

→ 형태소 분석기 Mecab 사용 → 시간 단축

```
    token_list = []
```

```
    for text in tqdm.tqdm(text_list):
```

```
        txt = re.sub('[^가-힣a-z]', ' ', text.lower())
```

```
        token = tokenizer.morphs(txt)
```

```
        token = [t for t in token if t not in stopwords or type(t) != float]
```

```
        token_list.append(token)
```

```
    return token_list, tokenizer
```

## 3.2.3. 1D CNN 적용 모델링

```
from tensorflow.keras.layers import Dense, Conv1D, GlobalMaxPooling1D, Embedding, Dropout, MaxPooling1D
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
model = Sequential()
model.add(Embedding(vocab_size, 300, weights = [embedding_matrix], input_length = 1000))
model.add(Dropout(0.2))
model.add(Conv1D(32, 5, strides=1, padding='valid', activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.summary()
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

1D 합성곱 연산을 수행하되, 커널수는 32, 커널의 크기는 5을 사용

GlobalMaxPooling1D를 사용하고, 두 개의 밀집층으로 은닉층과 출력층을 설계

정확도: 0.9724

```
es = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 3)
mc = ModelCheckpoint('best_model.h5', monitor = 'val_acc', mode = 'max', verbose = 1, save_best_only = True)
history = model.fit(train_X, train_y, epochs = 10, batch_size=64, validation_split=0.2, callbacks=[es, mc])
```



## 3. Modeling

3.1. 자연어 처리 과정 소개

3.2. 사용한 모델 소개

3.3. 최종 선정 모델

## 3.3.1. Tokenizing 및 불용어 제거

```
from konlpy.tag import Komoran
import re
import tqdm
komoran = Komoran()
```

```
def text_preprocessing(text_list):
```

```
    stopwords = ['을', '를', '이', '가', '은', '는', 'null']
```

```
    tokenizer = Komoran()
```

```
    token_list = []
```

```
    for text in tqdm.tqdm(text_list):
```

```
        try:
```

```
            txt = re.sub('[^가-힣a-z]', ' ', text.lower()) # 한글과 영어 소문자만 남기고 다른 글자 모두 제거
```

```
            token = tokenizer.morphs(txt) # 형태소 분석
```

```
            token = [t for t in token if t not in stopwords or type(t) != float] # 형태소 분석 결과 중 stopwords에 해당하지 않는 것만 추출
```

```
            token_list.append(token)
```

```
        except:
```

```
            token_list.append(None)
```

```
    return token_list, tokenizer
```

```
train['new_article'], komoran = text_preprocessing(train['content'])
```

→ 불용어 제거

→ 꼬모란 형태소 분석기 사용 (최적의 성능)

→ 토큰화한 형태소 저장

test data에 적용할 형태소 분석기 저장

## 3.3.1. Tokenizing 및 불용어 제거

```
1 train['new_article'].isnull().sum()
```

80

→ 앞서 사용한 Mecab tokenizer와 달리 komoran 사용시 결측치가 생김

```
1 train[train['new_article'].isnull() & train['info']==1]
```

→ 결측치 중 가짜 정보

	n_id	date	title	content	ord	info	new_article
584	NEWS07141	20200312	두산중공업아직 끝나지 않았습니다! 맥점과 대응전략 반드시 확인하세요!	}}	8	1	None

## 3.3.1. Tokenizing 및 불용어 제거

```
train[(train['new_article'].isnull()) & (train['info']==0)]
```

→ 결측치 중 진짜 정보

숫자 정보는 진짜로 판별함

	n_id	date	title	content	ord	info	new_article
4573	NEWS09664	20200308	[민후의 기·꼭·법]코로나19로 납기지연 됐다면?	10	9	0	None
4576	NEWS09664	20200308	[민후의 기·꼭·법]코로나19로 납기지연 됐다면?	12	12	0	None
4577	NEWS09664	20200308	[민후의 기·꼭·법]코로나19로 납기지연 됐다면?	27	13	0	None
11789	NEWS01847	20200429	휴일의 이슈&테마 스케줄(2020.04.30~05.04)	10	11	0	None
11791	NEWS01847	20200429	휴일의 이슈&테마 스케줄(2020.04.30~05.04)	11	13	0	None
...	...	...	...	...	...	...	...
107307	NEWS05354	20200303	"대구에 산다고 무조건 진료거부" 속타는 일반환자들	#2	31	0	None
107377	NEWS05354	20200303	"대구에 산다고 무조건 진료거부" 속타는 일반환자들	#1	101	0	None
107380	NEWS05354	20200303	"대구에 산다고 무조건 진료거부" 속타는 일반환자들	#1	104	0	None
107391	NEWS05354	20200303	"대구에 산다고 무조건 진료거부" 속타는 일반환자들	#2	115	0	None
107399	NEWS05354	20200303	"대구에 산다고 무조건 진료거부" 속타는 일반환자들	#2	123	0	None

## 3.3.1. Tokenizing 및 불용어 제거

```
# train으로 학습시킨 형태소 분석기 test에 적용
```

```
stopwords = ['을', '를', '이', '가', '은', '는', 'null']
```

```
token_list2 = []
```

```
for text in tqdm.tqdm(test['content']):
```

```
    try:
```

```
        txt2 = re.sub('[^가-힣a-z]', ' ', text.lower())
```

```
        token2 = komoran.morphs(txt2)
```

→ 앞서 저장해둔 형태소 분석기를 이용해 test data 토크나이징

```
        token2 = [t for t in token2 if t not in stopwords or type(t) != float]
```

```
        token_list2.append(token2)
```

```
    except:
```

```
        token_list2.append(None)
```

```
test['new_article'] = token_list2
```

→ 토큰화한 형태소 저장

## 3.3.2. Vectorization

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

train_notnull = train[train['new_article'].notnull()]

def text2sequence(train_text, max_len=1000):
    tokenizer = Tokenizer() #keras의 vectorizing 함수 호출
    tokenizer.fit_on_texts(train_text) #train 문장에 fit
    train_X_seq = tokenizer.texts_to_sequences(train_text) #각 토큰들에 정수 부여
    vocab_size = len(tokenizer.word_index) + 1 #모델에 알려줄 vocabulary의 크기 계산
    print('vocab_size : ', vocab_size)
    X_train = pad_sequences(train_X_seq, maxlen = max_len) #설정한 문장의 최대 길이만큼 padding

    return X_train, vocab_size, tokenizer

train_y = train_notnull['info']
train_X, vocab_size, vectorizer = text2sequence(train_notnull['new_article'], max_len = 100)
print(train_X.shape, train_y.shape)

test_notnull = test[test['new_article'].notnull()]
test_X_seq = vectorizer.texts_to_sequences(test_notnull['new_article'])
test_X = pad_sequences(test_X_seq, maxlen = 100)
```

→ 벡터화 결과, 단어 사전 사이즈,  
test data에 적용할 벡터라이저 저장

→ 저장해둔 벡터라이저를 이용해  
test data 벡터화

### 3.3.3. Embedding

```
import gensim
import numpy as np
```

```
word2vec = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin.gz', binary = True)
embedding_matrix = np.zeros((vocab_size, 300))
```

```
for index, word in enumerate(vectorizer.word_index):
    if word in word2vec:
        embedding_vector = word2vec[word]
        embedding_matrix[index] = embedding_vector
    else:
        print("word2vec에 없는 단어입니다.")
        continue
```

사전 훈련된 워드투벡 임베딩 모델

벡터라이저에 저장된 단어 사전을  
워드투벡 임베딩 모델로 임베딩시켜  
embedding\_matrix에 저장

## 3.3.4. 모델링

```
from tensorflow.keras.layers import Dense, Conv1D, GlobalMaxPooling1D, Embedding, Dropout, MaxPooling1D
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
model = Sequential()
model.add(Embedding(vocab_size, 300, weights = [embedding_matrix], input_length = 1000))
model.add(Dropout(0.2))
model.add(Conv1D(32, 5, strides=1, padding='valid', activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.summary()
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

embedding\_matrix가 가중치로 모델에 적용됨

1D CNN 모델 사용

```
es = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 3)
mc = ModelCheckpoint('best_model.h5', monitor = 'val_acc', mode = 'max', verbose = 1, save_best_only = True)
history = model.fit(train_X, train_y, epochs = 10, batch_size=64, validation_split=0.2, callbacks=[es, mc])
```



## 3.3.5. 예측

최종 결과물 정확도: 0.9730

```
pred_test = model.predict(test_X)
test_notnull.loc[:, 'info'] = np.where(pred_test > 0.5, 1, 0).reshape(-1)
test_notnull['id'] = test_notnull['n_id'] + "_" + test_notnull['ord'].map(str)
test_notnull.drop(['n_id', 'ord'], axis='columns', inplace=True)

test['id'] = test['n_id'] + "_" + test['ord'].map(str)
test.drop(['n_id', 'ord'], axis='columns', inplace=True)
merge = pd.merge(test, test_notnull, how='outer', on='id')
merge = merge.loc[:, ['id', 'info']]
merge

# train에서 발견한 결과 적용하여 결측치 채우기

merge.fillna(0, inplace=True) # 숫자들은 모두 0
merge.at[29937, 'info'] = 1
merge.at[31326, 'info'] = 1 # test 셋에서 확인 결과 이 두 인덱스의 content만 ']]' 있음 (train에서 이를 1로 판별)

merge.loc[:, ['id', 'info']].to_csv("submission_1DCNN.csv", index = False)
```

예측 확률이 0.5보다 크면 1, 그렇지 않으면 0

가짜 정보

## 4. 결론 및 한계점

# 4. 결론 및 한계점

## 결론

### 최종선정모델

Tokenizer : Komoran  
+  
Vectorizer : text2sequence  
+  
Embedding : word2vec  
+  
Modeling : 1D CNN

## 한계

- 1.한국어의 불용어 제거 기준 불명확
2. 다양한 파라미터 조정 시도의 부족
3. 유의미한 EDA 결과 적용의 한계

감사합니다