

# 자연어 처리

## 04. 텍스트 분류

### 영어 텍스트 분류

---

김상화(201978314)

01 Overview

02 문제 소개

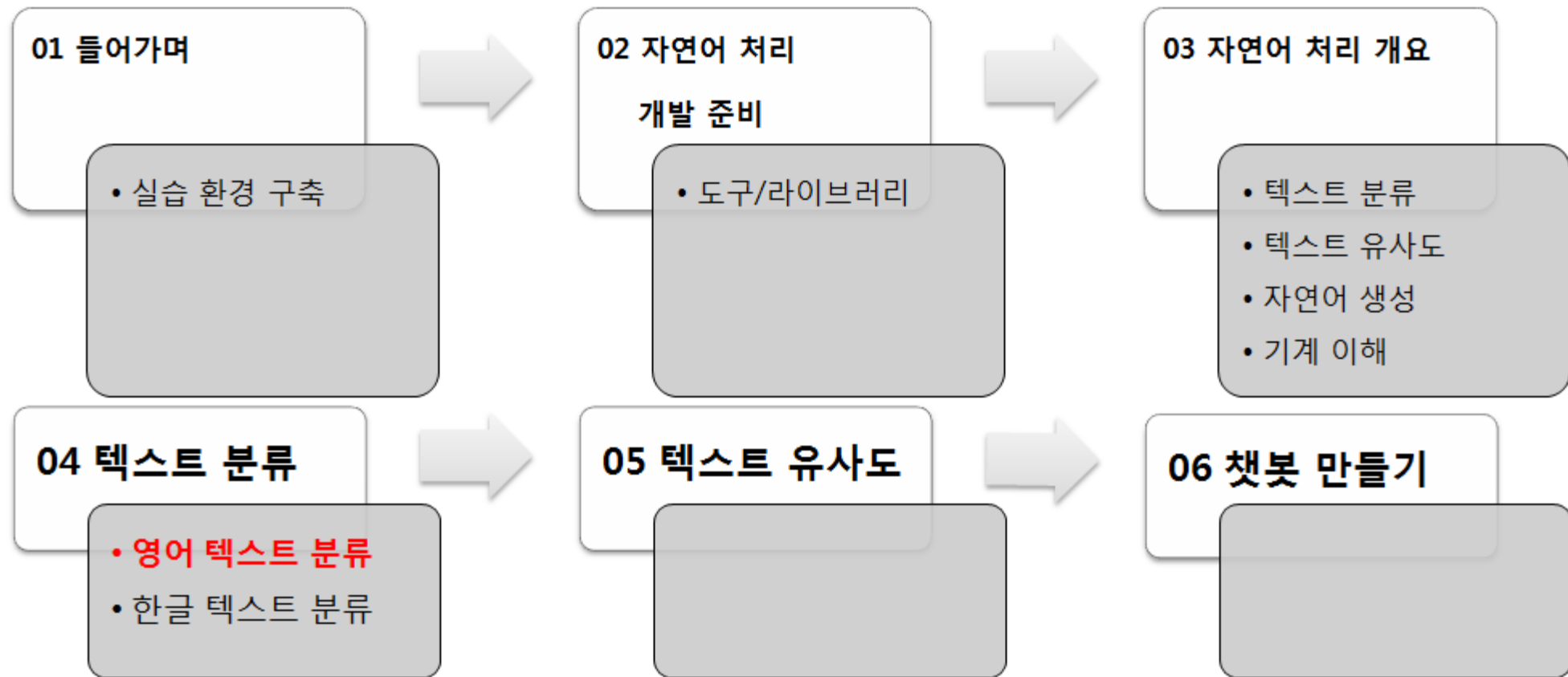
03 데이터 분석 및 전처리

04 모델링 소개

04-1 랜덤 포레스트 분류 모델

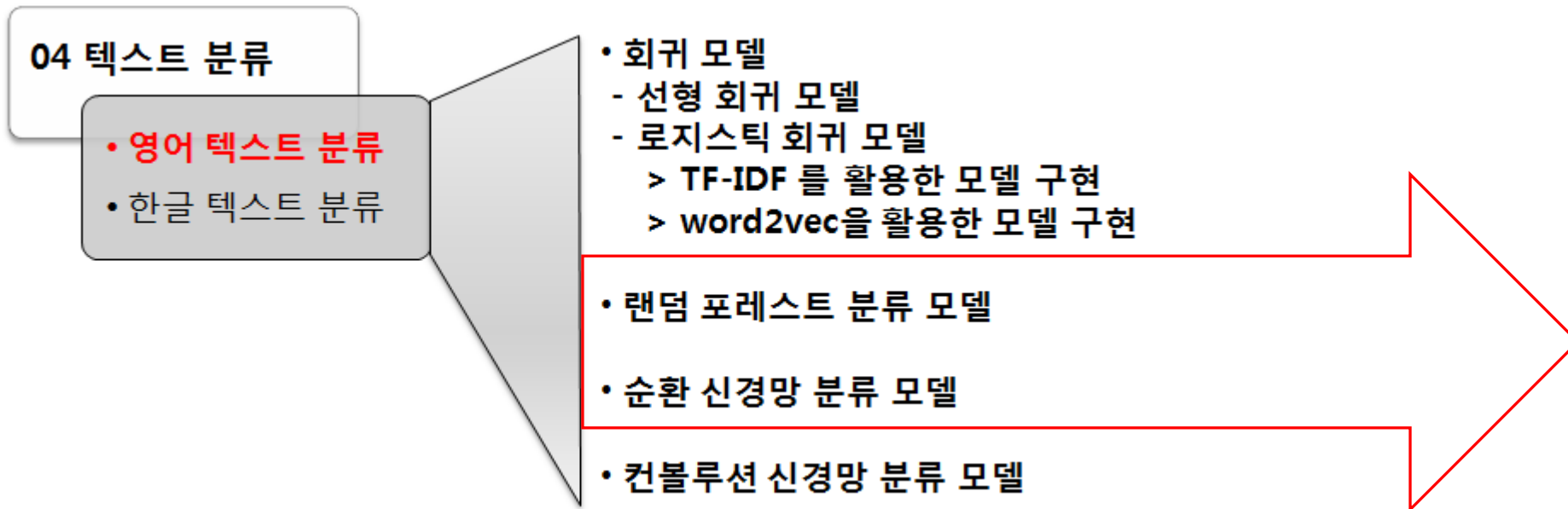
04-2 순환 신경망 분류 모델

# 01. Overview



# 01. Overview

**텍스트 분류:** 자연어 처리 기술을 활용, 글의 정보를 추출,  
문제에 맞게 사람이 정한 범주(Class)로 분류하는 문제



## 02. 문제 소개

- 워드 팝콘: 인터넷 영화 데이터베이스(IMDB)에서 나온 영화 평점을 활용한 캐글 문제

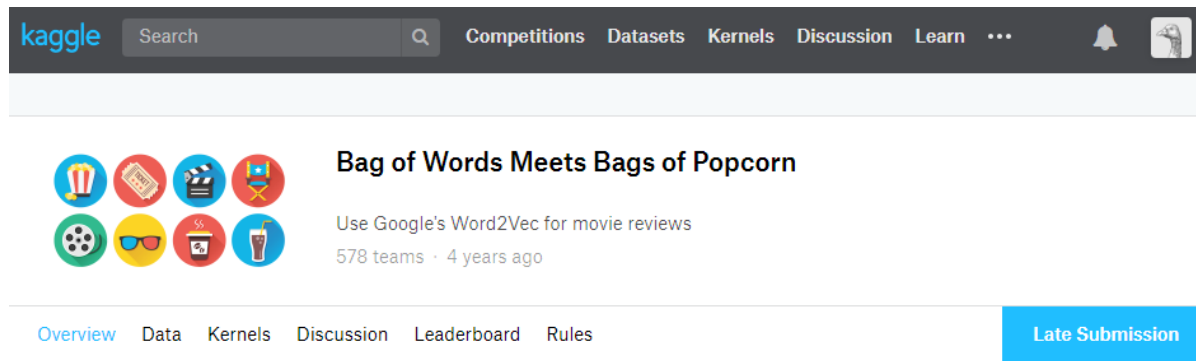
- 데이터: 영화 리뷰 텍스트와 평점에 따른 감정값(긍정/부정) => [감정 분류]

- 목표: 긍정 / 부정을 예측하는 모델

영화 리뷰 데이터를 학습해 새로운 리뷰가 긍정인지 부정인지 분류

참고) 단순 긍정 부정이 아닌 중립, 더 세부적인 정도로 나누는 것도 가능

<https://www.kaggle.com/c/word2vec-nlp-tutorial>



# 03. 데이터 분석 및 전처리 – 데이터 불러오기 및 분석

## 03-1. 데이터 불러오기 및 분석

### 03-1-1. 데이터 불러오기

- 1) API를 통해 내려 받는 방법
- 2) 사이트에서 zip 파일로 내려받는 방법

### 03-1-2. 데이터 분석(EDA)

- 1) 데이터 크기
- 2) 데이터 개수
- 3) 각 리뷰의 문자 길이 분포
- 4) 많이 사용된 단어
- 5) 긍, 부정 데이터의 분포
- 6) 각 리뷰의 단어 개수 분포
- 7) 특수문자 및 대, 소문자 비율

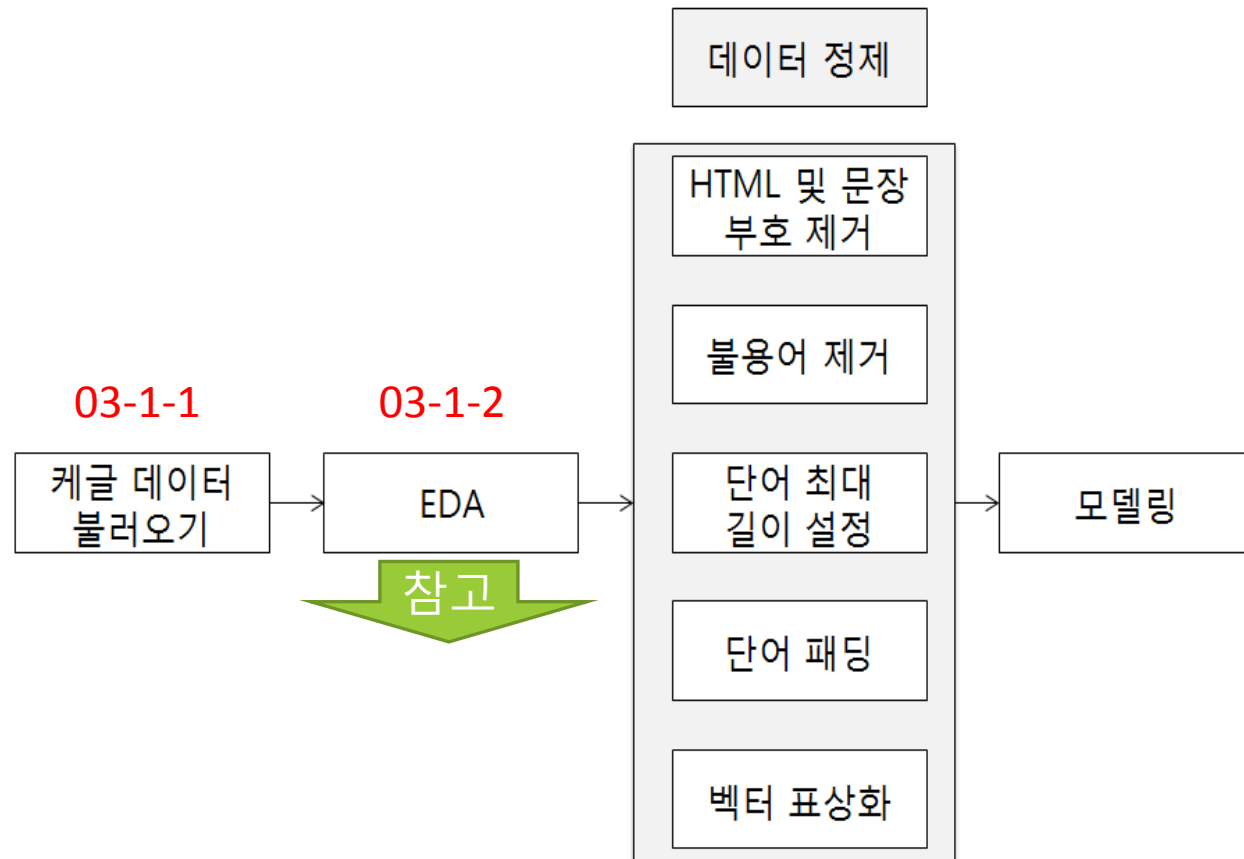


그림 4.2 워드팝콘 데이터의 처리 과정

# 03. 데이터 분석 및 전처리 - 데이터 불러오기 및 분석

## 03-1. 데이터 불러오기 및 분석

방법1) API를 통해 내려 받는 방법

방법2) 사이트에서 zip 파일로 내려받는 방법

Overview **Data** Kernels Discussion Leaderboard Rules Late Submission

Data (52 MB) 1 API kaggle competitions download -c word2vec-nlp-tut... ? 2 Download All ✕

**Data Sources**

- labeledTrainData.tsv
- sampleSubmission.c... 25.0k x 2
- testData.tsv
- unlabeledTrainData.tsv

**About this file**

No description yet

이름	수정한 날짜	유형	크기
labeledTrainData.tsv	2019-05-04 오후 3...	TSV 파일	32,770KB
sampleSubmission	2019-05-04 오후 3...	Microsoft Excel 씬...	277KB
testData.tsv	2019-05-04 오후 3...	TSV 파일	31,958KB
unlabeledTrainData.tsv	2019-05-04 오후 3...	TSV 파일	65,705KB
<input checked="" type="checkbox"/> word2vec-nlp-tutorial	2019-05-04 오후 3...	압축(ZIP) 폴더	53,688KB

# 03. 데이터 분석 및 전처리 – 데이터 불러오기 및 분석

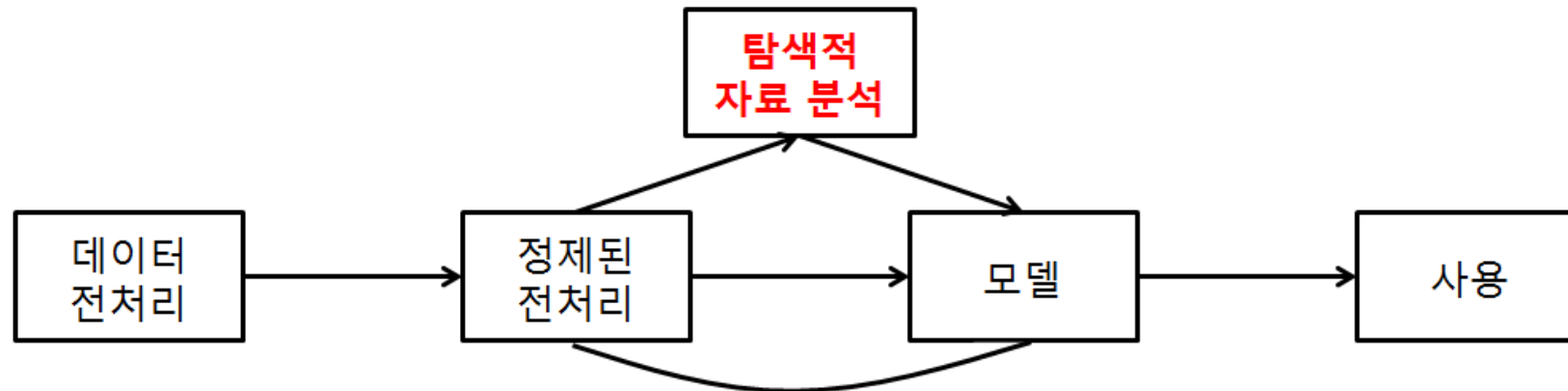
## 참고) 탐색적 데이터 분석(EDA: Exploratory Data Analysis)

- 모델 성능이 낮을 경우, 모델을 바꿀 수도 있지만 데이터에 대한 이해가 선행되어야 함!
- 데이터 이해 과정에서 생각지 못한 데이터의 여러 패턴이나 잠재적 문제점 발견 가능

## EDA 과정

- 정해진 틀 없이 데이터에 대하여 최대한 많은 정보를 뽑아냄
- 평균값, 중앙값, 최솟값, 최댓값, 범위, 분포, 이상치 등

데이터 분석 시, 선입견을 철저히 배제하고 데이터가 보여주는 수치만으로 분석 진행





# 03. 데이터 분석 및 전처리 - 데이터 불러오기 및 분석

## 03-1. 데이터 불러오기 및 분석

labeledTrainData.tsv

	A	B	C	D
1	id	sentiment	review	
2	5814_8	1	With all this stuff goi	
3	2381_9	1	WThe Classic War of	
4	7759_3	0	The film starts with a	
5	3630_4	0	It must be assumed	
24995	10957_3	0	I am a student of	
24996	2372_1	0	Unimaginably stup	
24997	3453_3	0	It seems like more	
24998	5064_1	0	I don't believe the	
24999	10905_3	0	Guy is a loser. Car	
25000	10194_3	0	This 30 minute do	
25001	8478_8	1	I saw this movie a	
25002				

// 헤더 제외 25,000 건

### Data fields

- **id** - Unique ID of each review
- **sentiment** - Sentiment of the review; 1 for positive reviews and 0 for negative reviews
- **review** - Text of the review

```
In [4]: import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

train\_data    labeledTrainData.tsv

```
In [6]: train_data = pd.read_csv(DATA_IN_PATH + 'labeledTrainData.tsv', header = 0, delimiter = '\t', quoting = 3)
train_data.head()    // 앞에서 5개의 데이터만 확인
```

Out[6]:

	id	sentiment	review
0	"5814_8"	1	"With all this stuff going down at the moment ...
1	"2381_9"	1	"\"The Classic War of the Worlds\" by Timothy ...
2	"7759_3"	0	"The film starts with a manager (Nicholas Bell...
3	"3630_4"	0	"It must be assumed that those who praised thi...
4	"9495_8"	1	"Superbly trashy and wondrously unpretentious ...

긍정

1

부정

0

참고  
판다스

## 03. 데이터 분석 및 전처리 – 데이터 불러오기 및 분석

### 참고) 판다스(Pandas) 라이브러리

- 편리한 데이터 구조와 데이터 분석 기능 제공

#### □ 판다스 데이터 구조

- 시리즈(Series): 1차원
- 데이터프레임(DataFrame): 2차원
- 패널(Panels): 3차원

#### □ 판다스 함수

- read\_csv: csv 파일 읽어 옴
- describe: 데이터의 평균, 표준편차 등 다양한 수치 값 얻음

## 03. 데이터 분석 및 전처리 - 데이터 불러오기 및 분석

### 03-1. 데이터 불러오기 및 분석 - 1. 데이터 크기

#### 데이터 크기

```
In [5]: ▶ print("파일 크기 : ")
        for file in os.listdir(DATA_IN_PATH):
            if 'tsv' in file and 'zip' not in file:
                print(file.ljust(30) + str(round(os.path.getsize(DATA_IN_PATH + file) / 1000000, 2)) + 'MB')
```

파일 크기 :

labeledTrainData.tsv	33.56MB
testData.tsv	32.72MB
unlabeledTrainData.tsv	67.28MB

### 03-1. 데이터 불러오기 및 분석 - 2. 데이터 개수

#### 데이터 개수

```
In [7]: ▶ print('전체 학습데이터의 개수: {}'.format(len(train_data)))
```

전체 학습데이터의 개수: 25000

# 03. 데이터 분석 및 전처리 - 데이터 불러오기 및 분석

## 03-1. 데이터 불러오기 및 분석 - 3. 각 리뷰의 문자 길이 분포(1/2)

### 각 리뷰의 문자 길이 분포

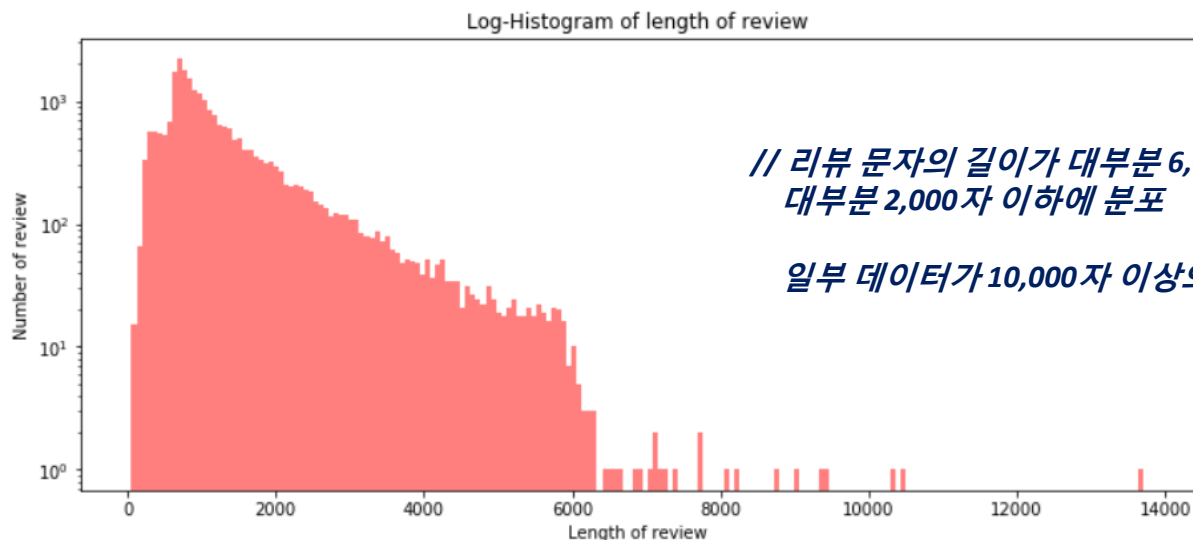
```
In [8]: ▶ train_length = train_data['review'].apply(len)
```

```
In [9]: ▶ train_length.head()
```

```
Out[9]: 0    2304
        1     948
        2    2451
        3    2247
        4    2233
        Name: review, dtype: int64
```

```
In [10]: ▶ # 그래프에 대한 이미지 사이즈 선언 // 시각화
# figsize: (가로, 세로) 형태의 튜플로 입력
plt.figure(figsize=(12, 5))
# 히스토그램 선언
# bins: 히스토그램 값들에 대한 버킷 범위
# range: x축 값의 범위
# alpha: 그래프 색상 투명도
# color: 그래프 색상
# label: 그래프에 대한 라벨
plt.hist(train_length, bins=200, alpha=0.5, color='r', label='word')
plt.yscale('log', nonposy='clip')
# 그래프 제목
plt.title('Log-Histogram of length of review')
# 그래프 x 축 라벨
plt.xlabel('Length of review')
# 그래프 y 축 라벨
plt.ylabel('Number of review')
```

```
Out[10]: Text(0, 0.5, 'Number of review')
```



// 리뷰 문자의 길이가 대부분 6,000자 이하이고  
대부분 2,000자 이하에 분포

일부 데이터가 10,000자 이상의 이상치(Outlier)

계속

# 03. 데이터 분석 및 전처리 - 데이터 불러오기 및 분석

## 03-1. 데이터 불러오기 및 분석 - 3. 각 리뷰의 문자 길이 분포(2/2)

```
In [11]: ▶ print('리뷰 길이 최대 값: {}'.format(np.max(train_length)))
print('리뷰 길이 최소 값: {}'.format(np.min(train_length)))
print('리뷰 길이 평균 값: {:.2f}'.format(np.mean(train_length)))
print('리뷰 길이 표준편차: {:.2f}'.format(np.std(train_length)))
print('리뷰 길이 중간 값: {}'.format(np.median(train_length)))
# 사분위의 대한 경우는 0~100 스케일로 되어있음
print('리뷰 길이 제 1 사분위: {}'.format(np.percentile(train_length, 25)))
print('리뷰 길이 제 3 사분위: {}'.format(np.percentile(train_length, 75)))
```

리뷰 길이 최대 값: 13710  
리뷰 길이 최소 값: 54  
리뷰 길이 평균 값: 1329.71  
리뷰 길이 표준편차: 1005.22  
리뷰 길이 중간 값: 983.0  
리뷰 길이 제 1 사분위: 705.0  
리뷰 길이 제 3 사분위: 1619.0

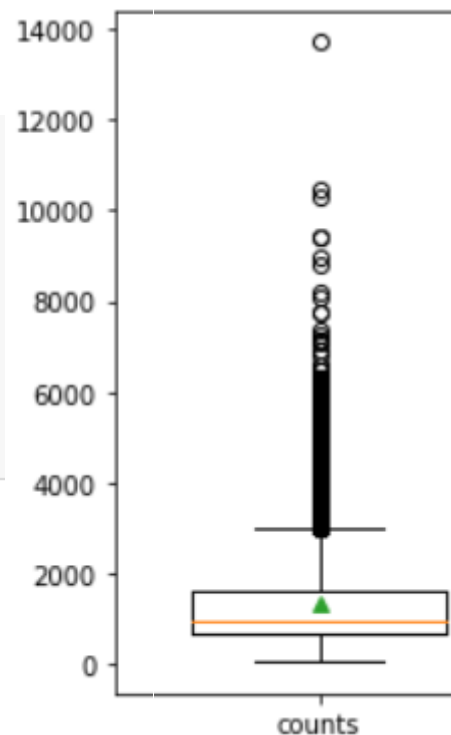
// 평균 1,329자  
최댓값 13,710자

대부분 2,000자 이하에 분포  
4,000자 이상의 이상치 데이터도 많이 분포

```
In [12]: ▶ plt.figure(figsize=(12, 5))
# 박스플롯 생성
# 첫번째 파라미터: 여러 분포에 대한 데이터 리스트를 입력
# labels: 입력한 데이터에 대한 라벨
# showmeans: 평균값을 마크함

plt.boxplot(train_length,
            labels=['counts'],
            showmeans=True)
```

```
Out[12]: {'whiskers': [<matplotlib.lines.Line2D at 0x9a1c22af98>,
<matplotlib.lines.Line2D at 0x9a1c22af60>],
'caps': [<matplotlib.lines.Line2D at 0x9a1c241668>,
<matplotlib.lines.Line2D at 0x9a1c2419b0>],
'boxes': [<matplotlib.lines.Line2D at 0x9a1c22ab00>],
'medians': [<matplotlib.lines.Line2D at 0x9a1c241cf8>],
'fliers': [<matplotlib.lines.Line2D at 0x9a1c2483c8>],
'means': [<matplotlib.lines.Line2D at 0x9a1c241f98>]}
```



## 03. 데이터 분석 및 전처리 - 데이터 불러오기 및 분석

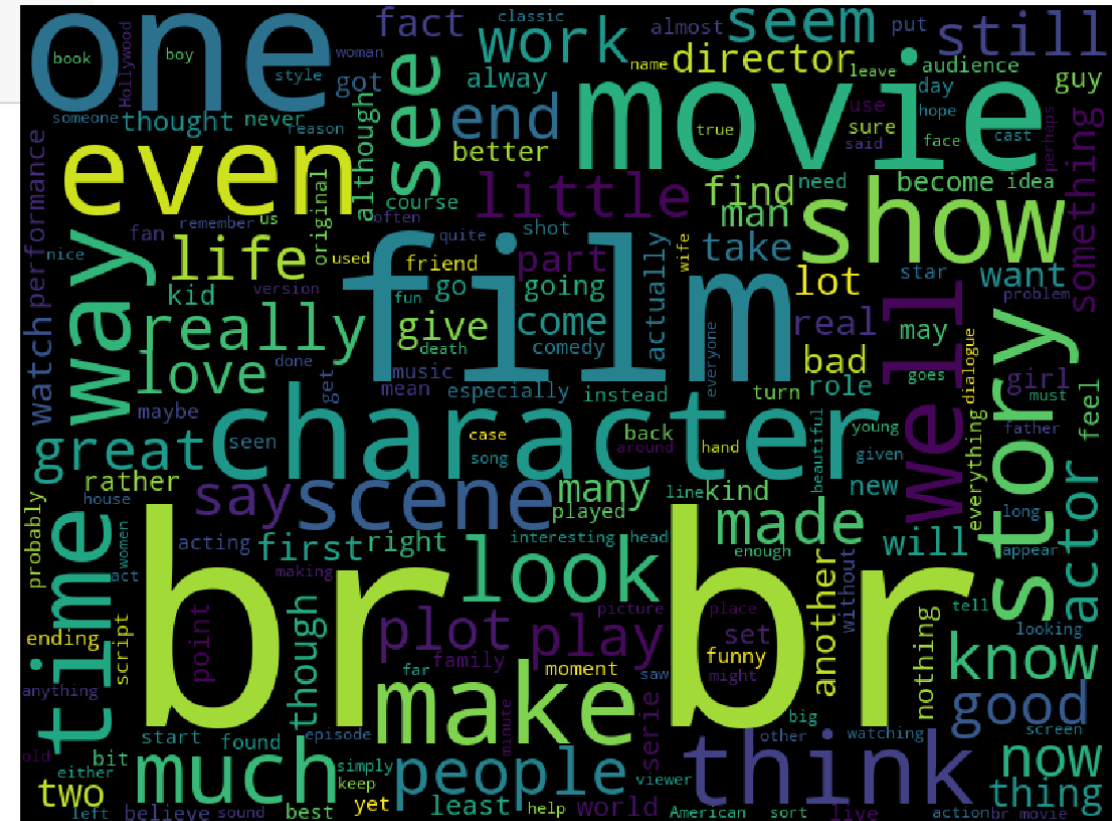
### 03-1. 데이터 불러오기 및 분석 - 4. 많이 사용된 단어

## 많이 사용된 단어

```
In [13]: from wordcloud import WordCloud // 워드 클라우드 라이브러리 사용
cloud = WordCloud(width=800, height=600).generate(" ".join(train_data['review']))
plt.figure(figsize=(20, 15))
plt.imshow(cloud)
plt.axis('off')
```

```
Out[13]: (-0.5, 799.5, 599.5, -0.5)
```

// <br> html 태그가 가장 많이 사용되는 단어  
→ 전처리 시 제거 필요!



# 03. 데이터 분석 및 전처리 - 데이터 불러오기 및 분석

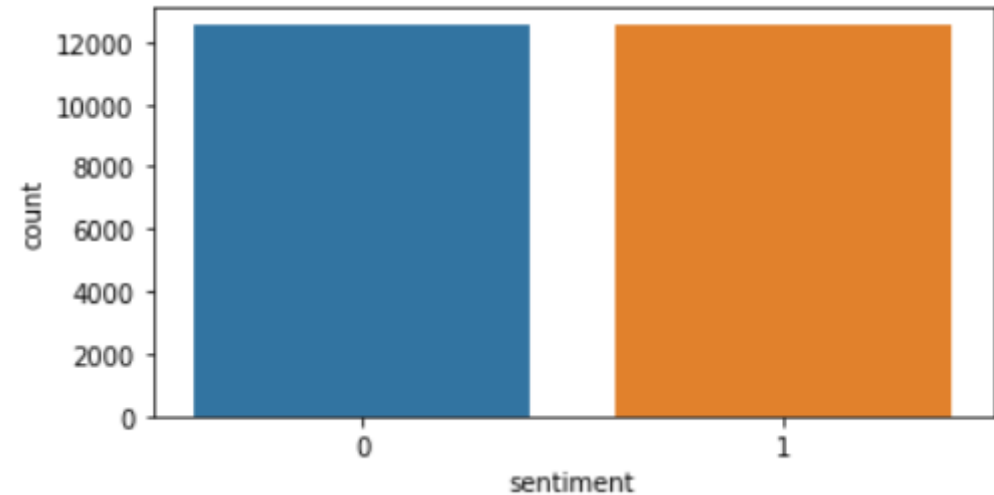
## 03-1. 데이터 불러오기 및 분석 - 5. 긍정, 부정 데이터의 분포

긍정	부정
1	0

### 긍, 부정 데이터의 분포

```
In [14]: fig, axe = plt.subplots(ncols=1)
fig.set_size_inches(6, 3)
sns.countplot(train_data['sentiment'])
```

Out[14]: <matplotlib.axes.\_subplots.AxesSubplot at 0x9a1c2eb198>



```
In [15]: print("긍정 리뷰 개수: {}".format(train_data['sentiment'].value_counts()[1]))
print("부정 리뷰 개수: {}".format(train_data['sentiment'].value_counts()[0]))
```

긍정 리뷰 개수: 12500  
부정 리뷰 개수: 12500

// 동일한 개수 분포 확인

# 03. 데이터 분석 및 전처리 - 데이터 불러오기 및 분석

## 03-1. 데이터 불러오기 및 분석 - 6. 각 리뷰의 단어 개수 분포 // 리뷰를 단어 기준으로 나눠 리뷰당 단어 개수 확인

### 각 리뷰의 단어 개수 분포

In [16]: `train_word_counts = train_data['review'].apply(lambda x: len(x.split(' ')))`

// 띄어쓰기를 기준으로 단어를 구분

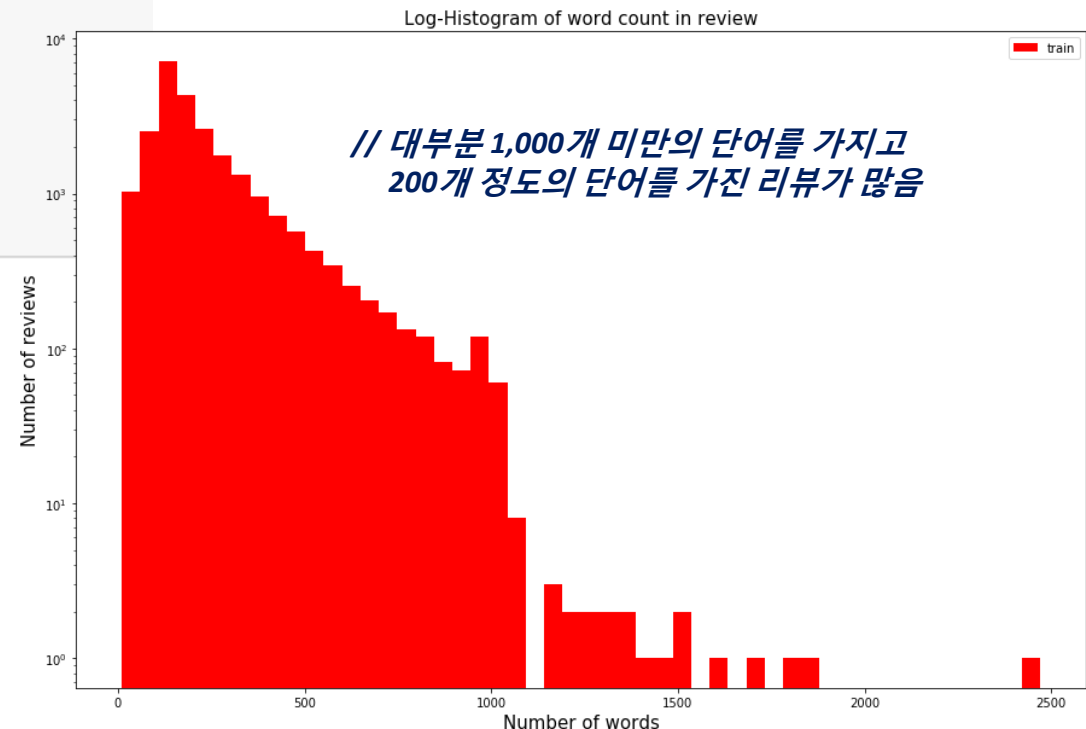
한글과 차이점: 국어는 띄어쓰기를 기준으로 모든 단어 처리 불가

In [17]: `plt.figure(figsize=(15, 10))  
plt.hist(train_word_counts, bins=50, facecolor='r', label='train')  
plt.title('Log-Histogram of word count in review', fontsize=15)  
plt.yscale('log', nonposy='clip')  
plt.legend()  
plt.xlabel('Number of words', fontsize=15)  
plt.ylabel('Number of reviews', fontsize=15)`

Out[17]: `Text(0, 0.5, 'Number of reviews')`

// 통계

리뷰 단어 개수 최대 값:	2470
리뷰 단어 개수 최소 값:	10
리뷰 단어 개수 평균 값:	233.79
리뷰 단어 개수 표준편차:	173.74
리뷰 단어 개수 중간 값:	174.0
리뷰 단어 개수 제 1 사분위:	127.0
리뷰 단어 개수 제 3 사분위:	284.0





## 03. 데이터 분석 및 전처리 - 데이터 불러오기 및 분석

### 03-1. 데이터 불러오기 및 분석 - 7. 특수문자 및 대, 소문자 비율

#### 특수문자 및 대, 소문자 비율

```
In [19]: ▶ qmarks = np.mean(train_data['review'].apply(lambda x: '?' in x)) # 물음표가 구두점으로 쓰임
fullstop = np.mean(train_data['review'].apply(lambda x: '.' in x)) # 마침표
capital_first = np.mean(train_data['review'].apply(lambda x: x[0].isupper())) # 첫번째 대문자
capitals = np.mean(train_data['review'].apply(lambda x: max([y.isupper() for y in x]))) # 대문자가 몇개
numbers = np.mean(train_data['review'].apply(lambda x: max([y.isdigit() for y in x]))) # 숫자가 몇개

print('물음표가있는 질문: {:.2f}%'.format(qmarks * 100))
print('마침표가 있는 질문: {:.2f}%'.format(fullstop * 100))
print('첫 글자가 대문자 인 질문: {:.2f}%'.format(capital_first * 100))
print('대문자가있는 질문: {:.2f}%'.format(capitals * 100))
print('숫자가있는 질문: {:.2f}%'.format(numbers * 100))
```

물음표가있는 질문: 29.55%  
 마침표가 있는 질문: 99.69%  
 첫 글자가 대문자 인 질문: 0.00%  
 대문자가있는 질문: 99.59%  
 숫자가있는 질문: 56.66%

// 대부분 마침표를 가지며(99.69%),  
 대문자도 사용(99.59%)

→ 전처리 시  
 모두 소문자로 통일하고  
 특수문자는 제거!

# 03. 데이터 분석 및 전처리 - 데이터 전처리

## 03-2. 데이터 전처리

### 1) 데이터 정제

- HTML 태그 제거
- 특수문자 제거

### 2) 불용어(stopword) 제거

### 3) 벡터화

### 4) 길이 통일(패딩)

케글 데이터

전처리 데이터

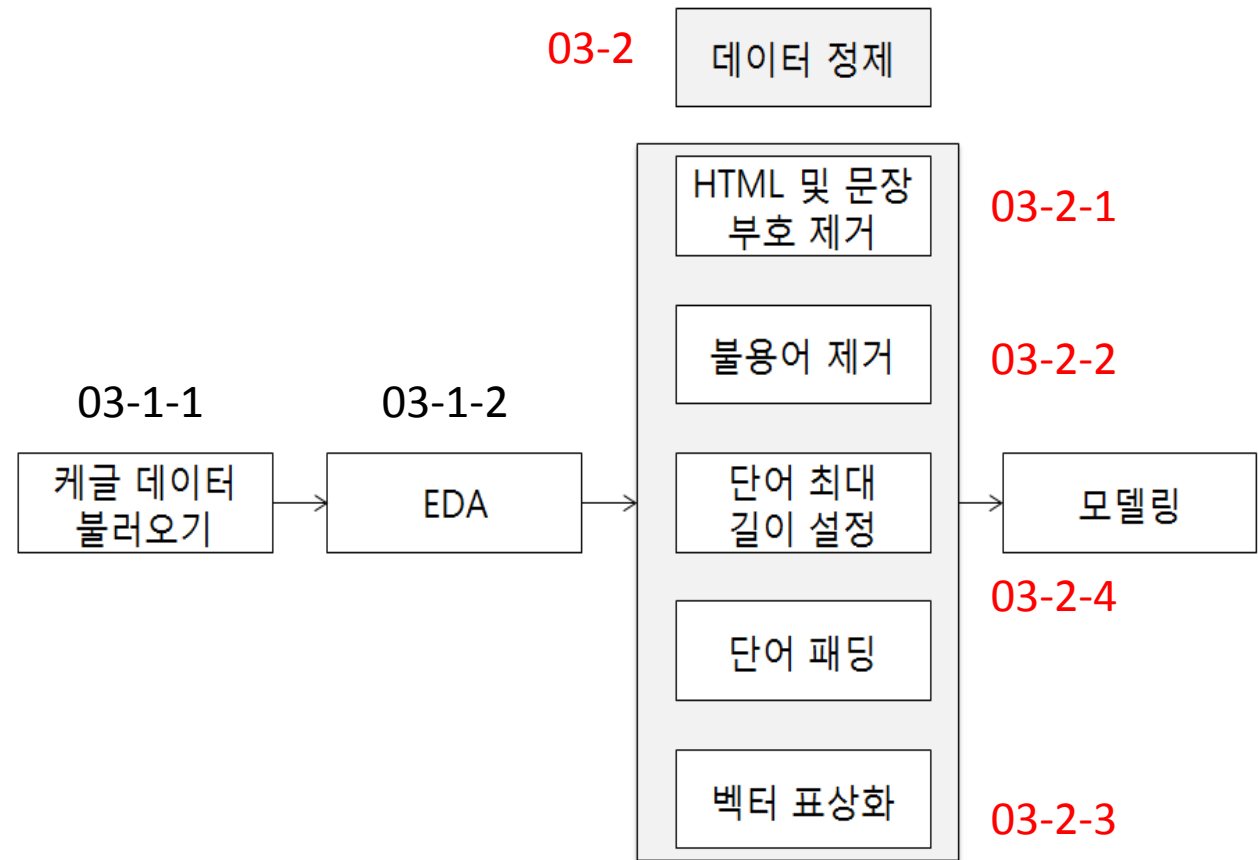


그림 4.2 워드팝콘 데이터의 처리 과정

# 03. 데이터 분석 및 전처리 – 데이터 전처리

## 03-2. 데이터 전처리

### 데이터 전처리

데이터 분석과정을 바탕으로 데이터를 모델에 적용시키기 위해 전처리 과정을 진행한다.

```
In [20]: import re
import json
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
from tensorflow.python.keras.preprocessing.text import Tokenizer
```

목적	라이브러리
데이터 정제	re / BeautifulSoup
불용어 제거	stopwords
텐서플로 전처리 모듈	pad_sequences / tokenizer

# 03. 데이터 분석 및 전처리 - 데이터 전처리

## 03-2-1. 데이터 정제

### 첫 번째 review 데이터 확인

```
In [21]: DATA_IN_PATH = './data_in/'

train_data = pd.read_csv(DATA_IN_PATH + 'labeledTrainData.tsv', header = 0, delimiter = '\t', quoting = 3)
print(train_data['review'][0])
```

- <br> 태그와 ... 등 특수문자 포함

의미에 큰 영향을 미치지 않으므로 최적화된 학습을 위해 제거

"With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there, watched The Wiz and watched Moonwalker again. Maybe i just want to get a certain insight into this guy who i thought was really cool in the eighties just to maybe make up my mind whether he is guilty or innocent. Moonwalker is part biography, part feature film which i remember going to see at the cinema when it was originally released. Some of it has subtle messages about MJ's feeling towards the press and also the obvious message of drugs are bad m'kay.<br /><br />Usually impressive but of course this is all about Michael Jackson so unless you remotely like MJ in anyway then you are going to hate this and find it boring. Some may call MJ an egotist for consenting to the making of this movie BUT MJ and most of his fans would say that he made it for the fans which if true is really nice of him.<br /><br />The actual feature film bit when it finally starts is only on for 20 minutes or so excluding the Smooth Criminal sequence and Joe Pesci is convincing as a psychopathic all powerful drug lord. Why he wants MJ dead so bad is beyond me. Because MJ overheard his plans? Nah, Joe Pesci's character ranted that he wanted people to know it is he who is supplying drugs etc so i dunno, maybe he just hates MJ's music.<br /><br />Lots of cool things in this like MJ turning into a car and a robot and the whole Speed Demon sequence. Also, the director must have had the patience of a saint when it came to filming the kiddy Bad sequence as usually directors hate working with one kid let alone a whole bunch of them performing a complex dance scene.<br /><br />Bottom line, this movie is for people who like MJ on one level or another (which i think is most people). If not, then stay away. It does try and give off a wholesome message and ironically MJ's bestest buddy in this movie is a girl! Michael Jackson is truly one of the most talented people ever to grace this planet but is he guilty? Well, with all the attention i've gave this subject...hmmm well i don't know because people can be different behind closed doors, i know this for a fact. He is either an extremely nice but stupid guy or one of the most sickest liars. I hope he is not the latter."

# 03. 데이터 분석 및 전처리 - 데이터 전처리

## 03-2-1. 데이터 정제

라이브러리	역할
re	특수 문제 제거
Beautiful Soup	HTML 태그 제거

```
In [22]: ▶ review = train_data['review'][0] # 리뷰 중 하나를 가져온다.
review_text = BeautifulSoup(review, "html5lib").get_text() # HTML 태그 제거
review_text = re.sub("[^a-zA-Z]", " ", review_text) # 영어 문자를 제외한 나머지는 모두 공백으로 바꾼다.
```

```
In [23]: ▶ print(review_text)
```

With all this stuff going down at the moment with MJ i ve started listening to his music. Maybe i just want to get a certain insight into whether he is guilty or innocent. Moonwalker was originally released. Some of it has substance and also the obvious message of drugs are bad m'kay. Visually impressive but of course you're not going to hate this and find it boring. Some of his fans would say that he made it for the fans while the feature film bit when it finally starts is only on for 20 minutes or so excluding the Smiling as a psychopathic all powerful drug lord. Why he wants MJ dead so bad is beyond me. His character ranted that he wanted people to know it is he who is supplying drugs etc so of cool things in this like MJ turning into a car and a robot and the whole Speed Demon patience of a saint when it came to filming the kiddy Bad sequence as usually directors much of them performing a complex dance scene Bottom line this movie is for people who is most people. If not then stay away. It does try and give off a wholesome message and is a girl. Michael Jackson is truly one of the most talented people ever to grace this attention i've gave this subject. Hmmmm well i don't know because people can be different. He is either an extremely nice but stupid

# 03. 데이터 분석 및 전처리 – 데이터 전처리

## 참고) 토큰나이징

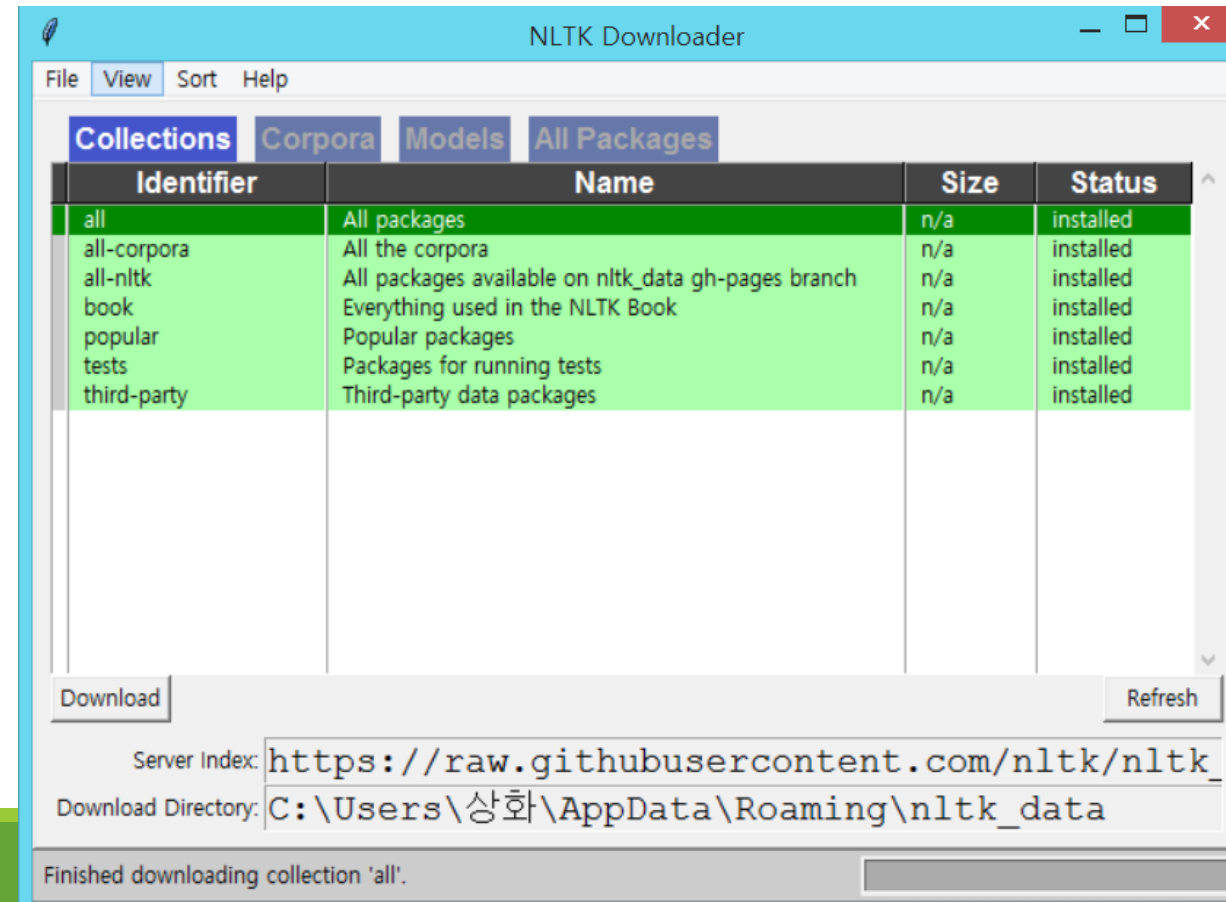
- 예측해야 할 입력 정보(문장 또는 발화)를 하나의 특정 기본 단위로 자르는 것

## □ 영어 토큰나이징 라이브러리: NLTK(Natural Language ToolKit), Spacy 가 대표적

- All-corpora: 텍스트 언어 분석을 위한 말뭉치 데이터셋
- book: 예시 데이터셋
- ※ All-corpora만 받아도 무방

## □ 불용어 제거

- NLTK 라이브러리에서 불용어 사전이 내장되어  
따로 정의할 필요 없이 바로 사용 가능



# 03. 데이터 분석 및 전처리 – 데이터 전처리

## 03-2-2. 불용어(stopword) 제거

※ 불용어: 문장에서 자주 출현하나 전체 의미에 큰 영향을 주지 않는 단어

ex) [영어] 관사, 조사 등

### 불용어 제거의 장단점

장점	단점
노이즈 요인 제거	문장 구문에 대한 전체적인 패턴 모델링 시 역효과

→ 진행중인 감정분석은 불용어가 감정 판단에 영향을 주지 않는다고 가정하고 제거!

### 방법: 불용어 사전(NLTK) 활용

- 사용자가 지정해도 무방하나 너무 많아 일반적으로 라이브러리 사전 활용
- NLTK 불용어 사전은 **모두 소문자**이므로 변환 후 활용

# 03. 데이터 분석 및 전처리 - 데이터 전처리

## 03-2-2. 불용어(stopword) 제거

```
In [24]: ▶ stop_words = set(stopwords.words('english')) # 영어 불용어들의 set을 만든다.

review_text = review_text.lower() // 소문자로 변환
words = review_text.split() # 소문자 변환 후 단어마다 나눠서 단어 리스트로 만든다.
words = [w for w in words if not w in stop_words] # 불용어 제거한 리스트를 만든다 // not in 속하지 않는 단어 리스트 셋을 words 에 담음
```

```
In [25]: ▶ print(words)

['stuff', 'going', 'moment', 'mj', 'started', 'listening', 'music', 'watching', 'odd', 'documentary', 'watched', 'wiz', 'watched', 'mo  
onwalker', 'maybe', 'want', 'get', 'certain', 'insight', 'guy', 'thought', 'really', 'cool', 'eighties', 'maybe', 'make', 'mind', 'whe  
ther', 'guilty', 'innocent', 'moonwalker', 'part', 'biography', 'part', 'feature', 'film', 'remember', 'going', 'see', 'cinema', 'orig  
inally', 'released', 'subtle', 'messages', 'mj', 'feeling', 'towards', 'press', 'also', 'obvious', 'message', 'drugs', 'bad', 'kay',  
'visually', 'impressive', 'course', 'michael', 'jackson', 'unless', 'remotely', 'like', 'mj', 'anyway', 'going', 'hate', 'find', 'bori  
ng', 'may', 'call', 'mj', 'egotist', 'consenting', 'making', 'movie', 'mj', 'fans', 'would', 'say', 'made', 'fans', 'true', 'really',  
'nice', 'actual', 'feature', 'film', 'bit', 'finally', 'starts', 'minutes', 'excluding', 'smooth', 'criminal', 'sequence', 'joe', 'pes  
ci', 'convincing', 'psychopathic', 'powerful', 'drug', 'lord', 'wants', 'mj', 'dead', 'bad', 'beyond', 'mj', 'overheard', 'plans', 'na  
h', 'joe', 'pesci', 'character', 'ranted', 'wanted', 'people', 'know', 'supplying', 'drugs', 'etc', 'dunno', 'maybe', 'hates', 'mj',  
'music', 'lots', 'cool', 'things', 'like', 'mj', 'turning', 'car', 'robot', 'whole', 'speed', 'demon', 'sequence', 'also', 'director',  
'must', 'patience', 'saint', 'came', 'filming', 'kiddy', 'bad', 'sequence', 'usually', 'directors', 'hate', 'working', 'one', 'kid',  
'let', 'alone', 'whole', 'bunch', 'performing', 'complex', 'dance', 'scene', 'bottom', 'line', 'movie', 'people', 'like', 'mj', 'one',  
'level', 'another', 'think', 'people', 'stay', 'away', 'try', 'give', 'wholesome', 'message', 'ironically', 'mj', 'bestest', 'buddy',  
'movie', 'girl', 'michael', 'jackson', 'truly', 'one', 'talented', 'people', 'ever', 'grace', 'planet', 'guilty', 'well', 'attention',  
'gave', 'subject', 'hmmm', 'well', 'know', 'people', 'different', 'behind', 'closed', 'doors', 'know', 'fact', 'either', 'extremely',  
'nice', 'stupid', 'guy', 'one', 'sickest', 'liars', 'hope', 'latter']
```



## 03. 데이터 분석 및 전처리 – 데이터 전처리

### 03-2-2. 불용어(stopword) 제거

- 불용어가 제거된 단어를 join 으로 하나의 글로 만듦 (모델에 적용하기 위해 하나의 문자열로 만듦)

In [26]: `clean_review = ' '.join(words) # 단어 리스트들을 다시 하나의 글로 합친다.  
print(clean_review)`

stuff going moment mj started listening music watching odd documentary watched wiz watched moonwalker maybe want get certain insight g  
uy thought really cool eighties maybe make mind whether guilty innocent moonwalker part biography part feature film remember going see  
cinema originally released subtle messages mj feeling towards press also obvious message drugs bad kay visually impressive course mich  
ael jackson unless remotely like mj anyway going hate find boring may call mj egotist consenting making movie mj fans would say made f  
ans true really nice actual feature film bit finally starts minutes excluding smooth criminal sequence joe pesci convincing psychopath  
ic powerful drug lord wants mj dead bad beyond mj overheard plans nah joe pesci character ranted wanted people know supplying drugs et  
c dunno maybe hates mj music lots cool things like mj turning car robot whole speed demon sequence also director must patience saint c  
ame filming kiddy bad sequence usually directors hate working one kid let alone whole bunch performing complex dance scene bottom line  
movie people like mj one level another think people stay away try give wholesome message ironically mj bestest buddy movie girl michae  
l jackson truly one talented people ever grace planet guilty well attention gave subject hmmm well know people different behind closed  
doors know fact either extremely nice stupid guy one sickest liars hope latter

현재까지 전처리 과정을  
25,000건에 적용하기 쉽도록 함수로 만듦

# 03. 데이터 분석 및 전처리 – 데이터 전처리

## 03-2. 데이터 전처리

### 1) 데이터 정제

+

### 2) 불용어(stopword) 제거

proprocessing 이름으로  
현재까지 전처리 과정을  
정의한 함수를 구현

```
In [27]: ▶ def preprocessing( review, remove_stopwords = False ):
# 불용어 제거는 옵션으로 선택 가능하다.

# 1. HTML 태그 제거
review_text = BeautifulSoup(review, "html5lib").get_text()

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

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

if remove_stopwords:
    # 4. 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]
    # 5. 단어 리스트를 공백을 넣어서 하나의 글로 합친다.
    clean_review = ' '.join(words)

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

return clean_review
```

# 03. 데이터 분석 및 전처리 - 데이터 전처리

proprocessing 함수

반복문을 사용해 25,000건의 리뷰를 전처리하고 저장

```
In [28]: ▶ clean_train_reviews = []
for review in train_data['review']:
    clean_train_reviews.append(preprocessing(review, remove_stopwords = True))

# 전처리한 데이터 출력
clean_train_reviews[0]
```

```
Out[28]: 'stuff going moment mj started listening music watching odd documentary watched wiz watched moonwalker maybe want get certain insight
guy thought really cool eighties maybe make mind whether guilty innocent moonwalker part biography part feature film remember going se
e cinema originally released subtle messages mj feeling towards press also obvious message drugs bad kay visually impressive course mi
chael jackson unless remotely like mj anyway going hate find boring may call mj egotist consenting making movie mj fans would say made
fans true really nice actual feature film bit finally starts minutes excluding smooth criminal sequence joe pesci convincing psychopat
hic powerful drug lord wants mj dead bad beyond mj overheard plans nah joe pesci character ranted wanted people know supplying drugs e
tc dunno maybe hates mj music lots cool things like mj turning car robot whole speed demon sequence also director must patience saint
came filming kiddy bad sequence usually directors hate working one kid let alone whole bunch performing complex dance scene bottom lin
e movie people like mj one level another think people stay away try give wholesome message ironically mj bestest buddy movie girl mich
ael jackson truly one talented people ever grace planet guilty well attention gave subject hmmm well know people different behind clos
ed doors know fact either extremely nice stupid guy one sickest liars hope latter'
```

```
In [29]: ▶ clean_train_df = pd.DataFrame({'review': clean_train_reviews, 'sentiment': train_data['sentiment']})
```

# 03. 데이터 분석 및 전처리 - 데이터 전처리

Tokenizer 모듈을 정제된 데이터에 적용하여 인덱스로 구성된 벡터로 변환

```
In [30]: > tokenizer = Tokenizer()
> tokenizer.fit_on_texts(clean_train_reviews)
> text_sequences = tokenizer.texts_to_sequences(clean_train_reviews)
```

리뷰가 텍스트가 아닌 인덱스의 벡터로 구성

```
In [31]: > print(text_sequences[0])

[404, 70, 419, 8815, 506, 2456, 115, 54, 873, 516, 178, 18686, 178,
0, 581, 2333, 1194, 11242, 71, 4826, 71, 635, 2, 253, 70, 11, 302,
5, 4424, 1851, 998, 146, 342, 1442, 743, 2424, 4, 8815, 418, 70, 63,
20, 323, 167, 10, 207, 633, 635, 2, 116, 291, 382, 121, 15535, 3315,
1, 15, 576, 8815, 22224, 2274, 13426, 734, 10013, 27, 28606, 340, 1,
4, 8815, 1430, 380, 2163, 114, 1919, 2503, 574, 17, 60, 100, 4875,
46, 114, 615, 3266, 1160, 684, 48, 1175, 224, 1, 16, 4, 8815, 3, 50,
4, 1, 128, 342, 1442, 247, 3, 865, 16, 42, 1487, 997, 2333, 12, 549,
8, 207, 254, 117, 3, 18688, 18689, 316, 1356]
```

```
In [32]: > word_vocab = tokenizer.word_index
> print(word_vocab)
```

```
{'movie': 1, 'film': 2, 'one': 3, 'like': 4, 'good': 5, 't
l': 12, 'much': 13, 'get': 14, 'bad': 15, 'people': 16, 'a
uld': 23, 'movies': 24, 'think': 25, 'characters': 26, 'ch
'life': 33, 'plot': 34, 'acting': 35, 'never': 36, 'love':
43, 'better': 44, 'end': 45, 'still': 46, 'say': 47, 'scen
'watching': 54, 'though': 55, 'thing': 56, 'old': 57, 'yea
3, 'nothing': 64, 'funny': 65, 'actually': 66, 'makes': 67
'world': 74, 'cast': 75, 'us': 76, 'quite': 77, 'want': 78
ror': 84, 'got': 85, 'however': 86, 'fact': 87, 'take': 88
94, 'give': 95, 'original': 96, 'action': 97, 'right': 98}
```

총 74,066개 단어로 구성

```
In [33]: > print("전체 단어 개수: ", len(word_vocab) + 1)

전체 단어 개수: 74066
```

획득 정보 추가 저장

```
In [35]: > data_configs = {}

> data_configs['vocab'] = word_vocab
> data_configs['vocab_size'] = len(word_vocab)
```

# 03. 데이터 분석 및 전처리 - 데이터 전처리

인덱스화 된 word\_index 사전

In [34]: `tokenizer.word_index`

```
Out[34]: {'movie': 1,
          'film': 2,
          'one': 3,
          'like': 4,
          'good': 5,
          'time': 6,
          'even': 7,
          'would': 8,
          'story': 9,
          'really': 10,
          'see': 11,
          'well': 12,
          'much': 13,
          'get': 14,
          'bad': 15,
          'people': 16,
          'also': 17,
          'first': 18,
          'great': 19,
          'mode': 20,
```

In [34]: `tokenizer.word_index`

```
adult': 982,
'amusing': 983,
'former': 984,
'focus': 985,
'common': 986,
'spend': 987,
'portrayal': 988,
'rated': 989,
'appreciate': 990,
'hair': 991,
'books': 992,
'pointless': 993,
'trash': 994,
'younger': 995,
'solid': 996,
'planet': 997,
'impressive': 998,
'super': 999,
'tone': 1000,
...}
```

# 03. 데이터 분석 및 전처리 - 데이터 전처리

## 03-2-4. 길이 통일(패딩)

- 모델에 적용하기 위한 동일 길이 통일 작업(특정 길이를 최대 길이로 지정)

1) 더 긴 데이터는 뒷부분을 자르고

2) 짧은 데이터는 0으로 패딩

```
In [36]: ▶ MAX_SEQUENCE_LENGTH = 174 // 최대길이지정 // 패딩적용
train_inputs = pad_sequences(text_sequences, maxlen=MAX_SEQUENCE_LENGTH, padding='post')
print('Shape of train data: ', train_inputs.shape)

Shape of train data: (25000, 174) // 25,000건의 데이터가 동일한 길이 174를 가지게 됨
```

```
In [37]: ▶ train_labels = np.array(train_data['sentiment']) // sentiment (정답)을 배열로 저장
print('Shape of label tensor:', train_labels.shape)

Shape of label tensor: (25000,)
```

리뷰 단어 개수 최대 값: 2470  
리뷰 단어 개수 최소 값: 10  
리뷰 단어 개수 평균 값: 233.79  
리뷰 단어 개수 표준편차: 173.74  
리뷰 단어 개수 중간 값: 174.0  
리뷰 단어 개수 제 1 사분위: 127.0  
리뷰 단어 개수 제 3 사분위: 284.0

// 평균은 이상치 영향력이 커서  
평균이 아닌 중간 값을 사용하는 경우가 많음

# 03. 데이터 분석 및 전처리 - 데이터 전처리

## 03-2 결과> 모델에 적용할 입력 데이터(4)[파일 형식/확장자]

- 1) 벡터화 한 데이터[넘파이 파일 npy]
- 2) 정답 라벨 [넘파이 파일 npy]
- 3) 정제된 텍스트 데이터[csv]
- 4) 데이터 정보(단어 사전, 전체 단어 개수)[json]

In [38]:

```

▶ TRAIN_INPUT_DATA = 'train_input.npy'
  TRAIN_LABEL_DATA = 'train_label.npy'
  TRAIN_CLEAN_DATA = 'train_clean.csv'
  DATA_CONFIGS = 'data_configs.json'

import os
# 저장하는 디렉토리가 존재하지 않으면 생성
if not os.path.exists(DATA_IN_PATH):
    os.makedirs(DATA_IN_PATH)

# 전처리 된 데이터를 넘파이 형태로 저장
np.save(open(DATA_IN_PATH + TRAIN_INPUT_DATA, 'wb'), train_inputs)
np.save(open(DATA_IN_PATH + TRAIN_LABEL_DATA, 'wb'), train_labels)

# 정제된 텍스트를 csv 형태로 저장
clean_train_df.to_csv(DATA_IN_PATH + TRAIN_CLEAN_DATA, index = False)

// 원하는 행태로 저장
# 데이터 사전을 json 형태로 저장
json.dump(data_configs, open(DATA_IN_PATH + DATA_CONFIGS, 'w'), ensure_ascii=False)
  
```

학습 데이터에 대한  
전처리

평가 데이터에 대한  
전처리

# 03. 데이터 분석 및 전처리 - 데이터 전처리

평가 데이터에 대한 전처리 및 저장

train\_data

labeledTrainData.tsv

test\_data

testData.tsv

```
In [40]: ▶ test_data = pd.read_csv(DATA_IN_PATH + "testData.tsv", header=0, delimiter="\t", quoting=3)

clean_test_reviews = []
for review in test_data['review']:
    clean_test_reviews.append(preprocessing(review, remove_stopwords = True))

clean_test_df = pd.DataFrame({'review': clean_test_reviews, 'id': test_data['id']})
test_id = np.array(test_data['id'])

text_sequences = tokenizer.texts_to_sequences(clean_test_reviews)
test_inputs = pad_sequences(text_sequences, maxlen=MAX_SEQUENCE_LENGTH, padding='post')
```

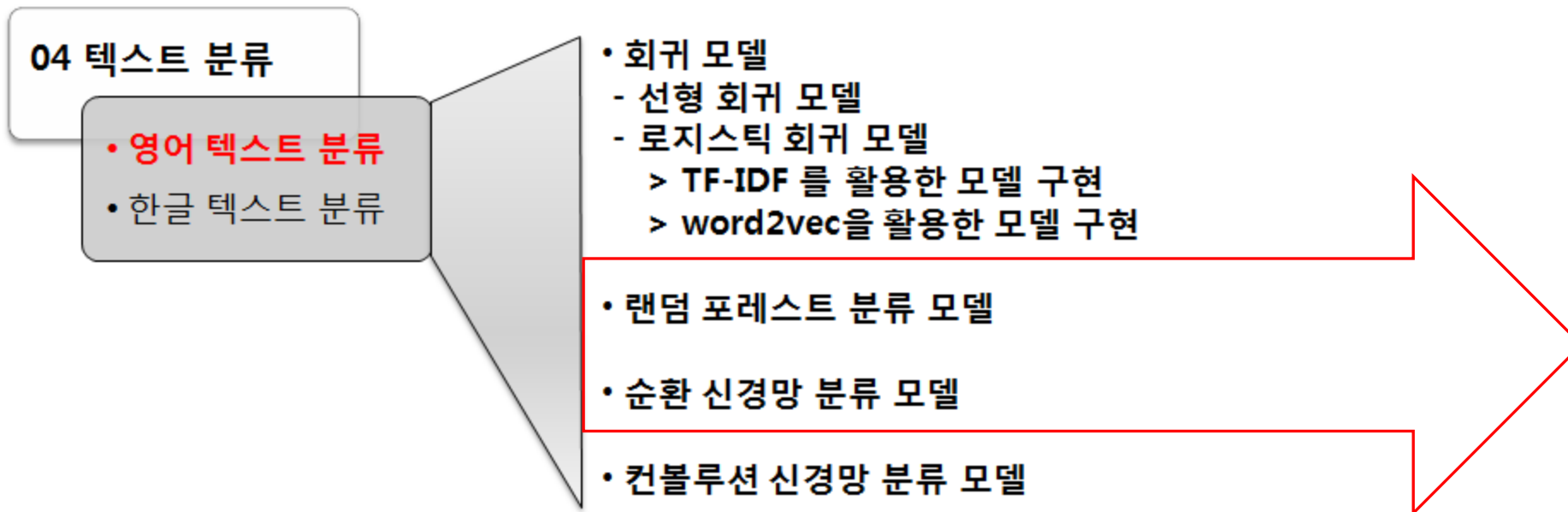
labeledTrainData.tsv  
sampleSubmission  
testData.tsv  
unlabeledTrainData.tsv

```
In [41]: ▶ TEST_INPUT_DATA = 'test_input.npy'
TEST_CLEAN_DATA = 'test_clean.csv'
TEST_ID_DATA = 'test_id.npy'

np.save(open(DATA_IN_PATH + TEST_INPUT_DATA, 'wb'), test_inputs)
np.save(open(DATA_IN_PATH + TEST_ID_DATA, 'wb'), test_id)
clean_test_df.to_csv(DATA_IN_PATH + TEST_CLEAN_DATA, index = False)
```



# 04. 모델링 소개

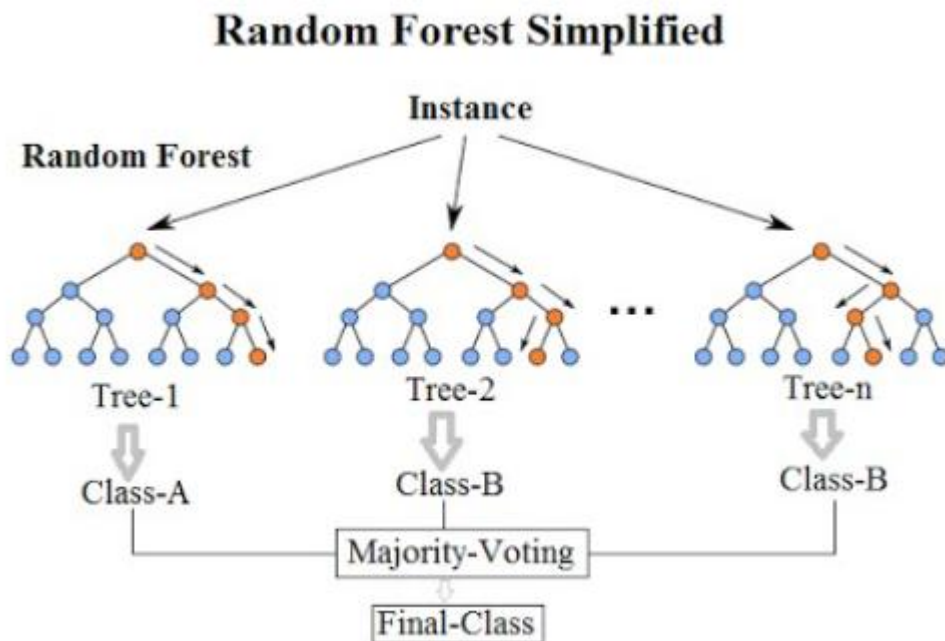


구분	모델	핵심 라이브러리
머신러닝	회귀모델 랜덤 포레스트	사이킷런
딥러닝	CNN(합성곱 신경망) RNN(순환 신경망)	텐서플로

# 04. 모델링 소개 - 랜덤 포레스트 분류 모델

랜덤 포레스트: 여러 개의 의사결정 트리 결과값을 평균 낸 것 -> 분류/회귀

학습 전용 데이터를 랜덤 샘플링하여 다수의 결정 트리 만들고, 만들어진 결정 트리들의 결과들을 모아 다수결로 최종 결과를 도출하는 알고리즘이다. 집단 학습을 기반으로 분류, 회귀, 클러스터링 등을 구현한다. 앙상블(ensemble) 학습 방법의 일종이다.(집단 학습)



출처: <https://blog.naver.com/PostView.nhn?blogId=sbd38&logNo=221373436623>

# 참고1) 앙상블(ensemble) 학습/방법 (1/2)

## 1. 투표 기반 분류기(Voting Classifier)

1.1 직접 투표(Hard voting): 새로운 데이터에 대한 모델 별 예측 값을 다수결 투표를 통해  
최종 클래스를 예측하는 방법

1.2 간접 투표(Soft voting): 각 모델 예측 값의 확률을 가지고 평균을 구한 뒤,  
평균이 가장 높은 클래스로 최종 앙상블 예측

## 2. 배깅과 페이스팅(Bagging and Pasting)

- 하나의 모델에 학습 데이터를 다르게 학습시키는 방법

2.1 배깅(bagging = bootstrap aggregating): 중복 허용 리샘플링(resampling) = bootstrapping

2.2 페이스팅(pasting): 중복 허용하지 않는 샘플링 방식

### 2.1.1 OOB(Out-of-Bag) 평가

- 배깅은 중복을 허용하는 리샘플링 방식이므로 전체 데이터 셋에서 어떤 샘플은 여러 번 샘플링 되고, 어떤 샘플은 전혀 샘플링 되지 않을 수 있음
- 샘플링 되지 않는 샘플을 OOB 샘플이라고 함
- 배깅에서는 이런 OOB 샘플을 validation set 이나 cross validation 에 사용 가능

출처: <https://excelsior-cjh.tistory.com/166>

# 참고1) 앙상블(ensemble) 학습/방법 (2/2)

## 3. 랜덤 포레스트

- 배깅을 적용한 의사결정나무의 앙상블

## 4. 부스팅(Boosting)

- 성능이 약한 학습기 여러 개를 연결해 강한 학습기를 만드는 앙상블 학습
  - 4.1 아다부스트(AdaBoost, Adaptive Boosting):  
과소적합(underfitted) 됐던 학습 데이터 샘플의 가중치를 높이며  
새로 학습된 모델이 학습하기 어려운 데이터를 더 잘 적합 되도록 하는 방식
  - 4.2 그래디언트 부스팅(Gradient Boosting):  
아다부스트처럼 학습 단계마다 가중치를 업데이트 하지 않고  
학습 전 모델에서의 잔여 오차(residual error)에 대해 새로운 모델에 학습

## 5 스택킹(Stacking)

- 각 모델의 예측값을 가지고 새로운 메타모델을 학습시켜 최종 예측 모델을 만드는 방법

출처: <https://excelsior-cjh.tistory.com/166>

# 04. 모델링 소개 - 랜덤 포레스트 분류 모델

참고) 사이킷런을 이용한 특징추출

모듈: **CountVectorizer**

텍스트 데이터를 수치화/벡터화

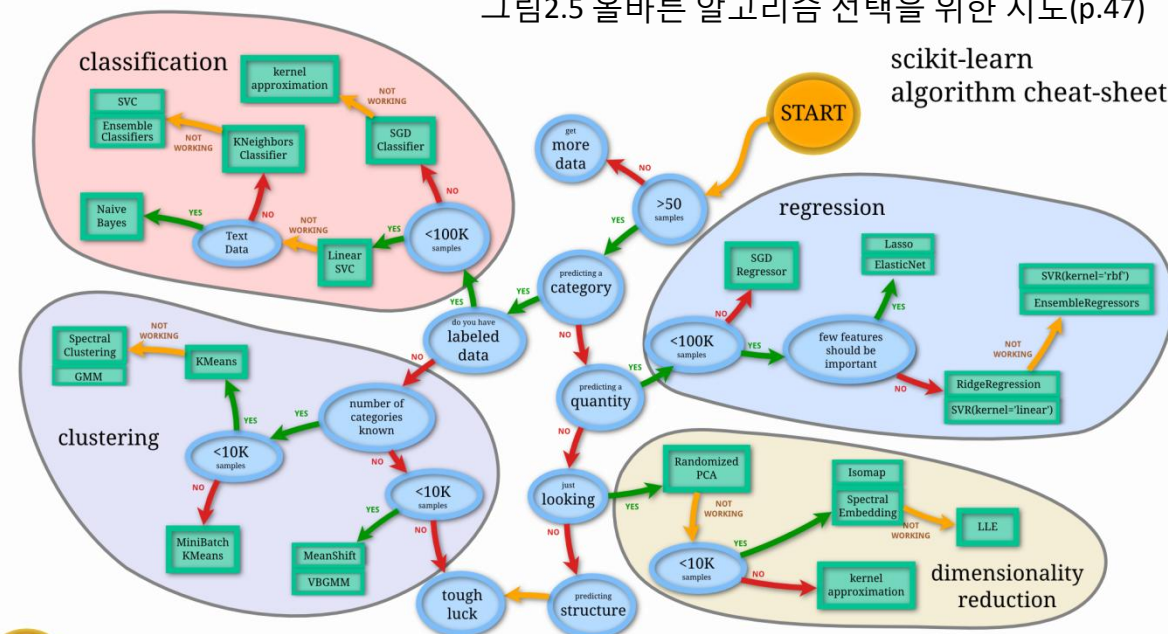
각 텍스트에서 **횟수**를 기준으로 특징을 추출하는 방법

// 사이킷런(scikit-learn): 파이썬용 머신러닝 라이브러리

- 지도학습을 위한 모듈  
: 나이브 베이즈, 의사결정트리, SVM
- 비지도학습을 위한 모듈  
: 군집화, 가우시안 혼합 모델
- 모델 선택 및 평가를 위한 모듈  
: 교차검증, 모델평가  
, 모델의 지속성을 위한 모델 저장과 불러오기
- 데이터 변환 및 데이터를 불러오기 위한 모듈  
: 파이프라인, 특징 추출, 전처리, 차원 축소
- 계산 성능 향상을 위한 모듈

장점	단점
<ul style="list-style-type: none"> <li>■간단하게 특징 추출 가능</li> <li>■직관적이며 여러 상황에 적용 가능</li> </ul>	<ul style="list-style-type: none"> <li>■큰 의미 없이 자주 사용되는 단어(조사, 지시대명사 등)가 높은 특징을 가져 유의미하게 사용하기 어려울 수 있음</li> </ul>

그림2.5 올바른 알고리즘 선택을 위한 지도(p.47)



## 04. 모델링 소개 - 랜덤 포레스트 분류 모델

전처리 된 (train\_clean.csv) 를 불러옴

### 패키지 및 데이터 불러오기 불러오기

```
In [1]: ▶ import pandas as pd  
import numpy as np  
import os  
from sklearn.feature_extraction.text import CountVectorizer
```

```
In [2]: ▶ DATA_IN_PATH = './data_in/'  
DATA_OUT_PATH = './data_out/'  
TRAIN_CLEAN_DATA = 'train_clean.csv'  
TEST_SIZE = 0.2  
RANDOM_SEED = 42
```

```
In [3]: ▶ train_data = pd.read_csv(DATA_IN_PATH + TRAIN_CLEAN_DATA)
```

```
In [4]: ▶ reviews = list(train_data['review'])  
y = np.array(train_data['sentiment'])
```

## 04. 모델링 소개 - 랜덤 포레스트 분류 모델

텍스트 데이터를 특징 벡터로 만들어야 모델에 입력으로 사용 가능

CountVectorizer 로 리뷰 정보를 벡터화

### CountVectorizer를 활용한 벡터화

```
In [5]: ▶ vectorizer = CountVectorizer(analyzer = "word", max_features = 5000) // 분석 단위: 단어 하나
        train_data_features = vectorizer.fit_transform(reviews)           벡터 최대 길이 5,000
```

train\_data\_features 에 담아서 input 으로 모델에 적용!

Before) 학습 데이터와 검증 데이터를 만들어서 모델 성능 측정 필요!

- 검증 데이터는 학습 데이터의 일부로 만들

## 04. 모델링 소개 - 랜덤 포레스트 분류 모델

사이킷런으로 검증 데이터 생성

// 사이킷런(scikit-learn)  
train\_test\_split 함수에  
학습데이터와 라벨을 넣으면 정의한 비율로 데이터를 나눔

### 학습과 검증 데이터 분리

```
In [6]: ▶ from sklearn.model_selection import train_test_split
```

```
In [7]: ▶ train_input, eval_input, train_label, eval_label = train_test_split(train_data_features, y, test_size=TEST_SIZE, random_state=RANDOM_SEED)
```

사이킷런 라이브러리의 RandomForestClassifier 객체로 모델 구현



## 04. 모델링 소개 - 랜덤 포레스트 분류 모델

### 모델 구현 및 학습

```
In [8]: ▶ from sklearn.ensemble import RandomForestClassifier

# 랜덤 포레스트 분류기에 100개 의사 결정 트리를 사용한다.
forest = RandomForestClassifier(n_estimators = 100) // 100개의 트리로 각 결과를 앙상블하고 최종 결과를 만들어냄

# 단어 묶음을 벡터화한 데이터와 정답 데이터를 가지고 학습을 시작한다.
forest.fit( train_input, train_label ) // 생성한 모델에 학습 데이터와 라벨(정답)을 적용해 학습
```

```
Out[8]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
                                oob_score=False, random_state=None, verbose=0,
                                warm_start=False)

// score 함수로 성능 측정
```

```
In [9]: ▶ print("Accuracy: %f" % forest.score(eval_input, eval_label)) # 검증함수로 정확도 측정

Accuracy: 0.845000 // 84%로 그리 높지 않은 정확도를 보임.
```

모델을 바꾸지 않더라도 입력값 만드는 방식을 바꾸면 성능이 좋아질 수도 있음!











# 04. 모델링 소개 - 랜덤 포레스트 분류 모델

kaggle 점수 0.84156

Name	Submitted	Wait time	Execution time	Score
Bag_of_Words_model.csv	just now	0 seconds	0 seconds	0.84156

Complete

[Jump to your position on the leaderboard](#) ▼

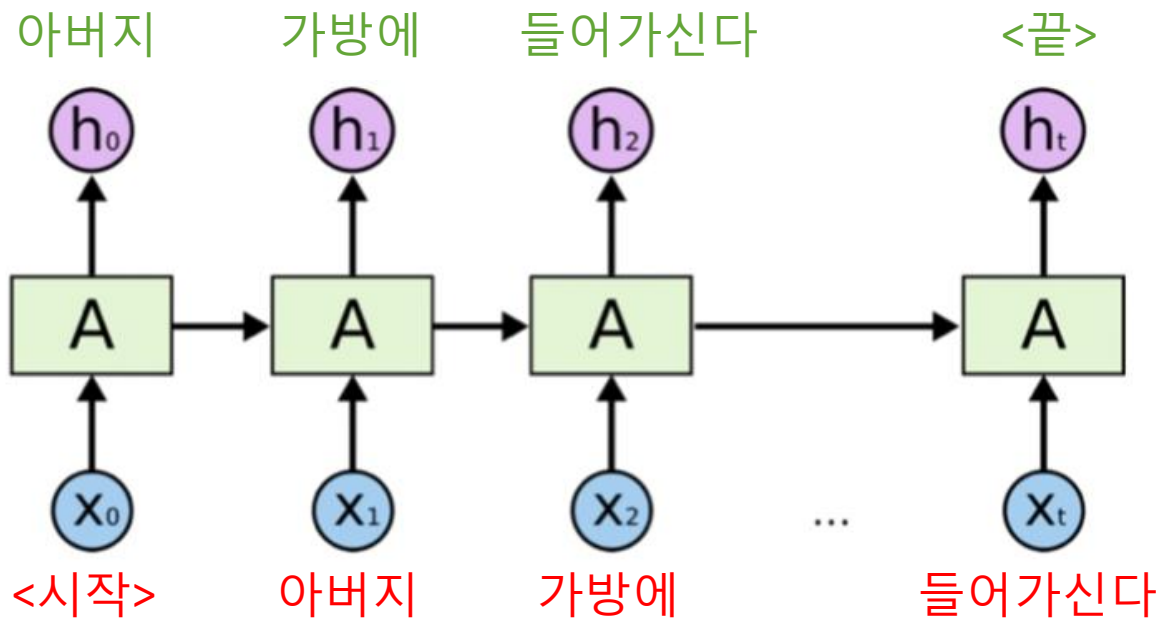
#	Team Name	Kernel	Team Members	Score ?	Entries	Last
1	Alejandro Peláez			0.99259	25	4y
2	Sebastian Raschka_			0.99156	5	4y
3	vgng			0.97663	22	4y
4	gump			0.97617	9	4y
5	Flexic			0.97567	24	4y
6	honzas			0.97383	1	4y
7	broucek			0.97325	1	4y
8	V			0.97309	8	4y
9	Cristian			0.97209	6	4y
10	andy1618			0.97144	5	4y

# 04. 모델링 소개 - RNN(순환 신경망)

RNN: 현재 정보는 이전 정보가 점층적으로 쌓이면서 정보를 표현할 수 있는 모델

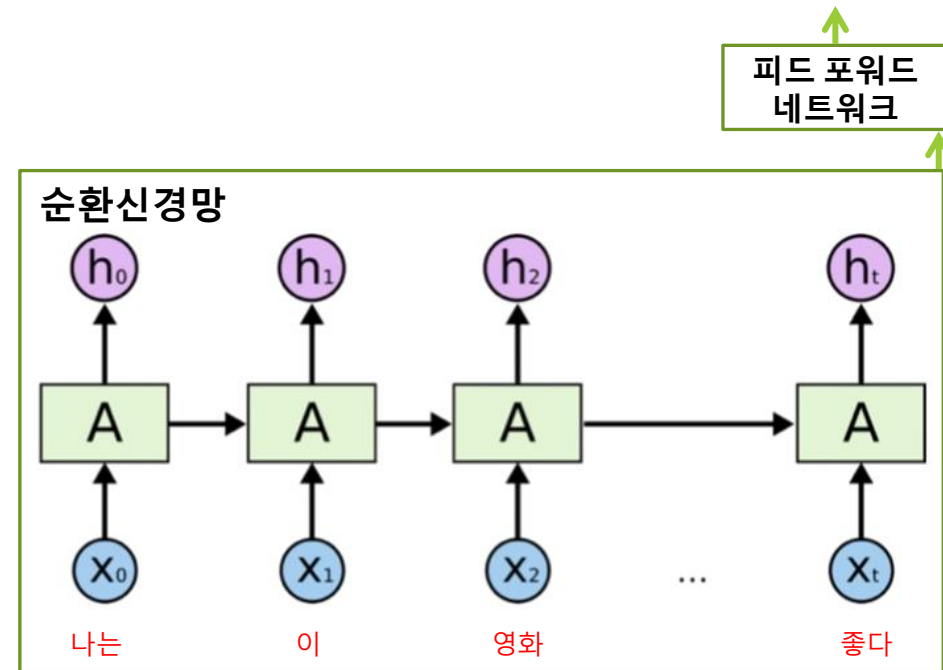
시간에 의존적인/순차적인 데이터에 대한 문제에 활용

// 연관검색어 예측 같은 개념으로 이해



// '가방에'를 넣으면 '들어가신다'를 예측  
예측 시, '아버지', '가방에'를 활용

긍정 or 부정?



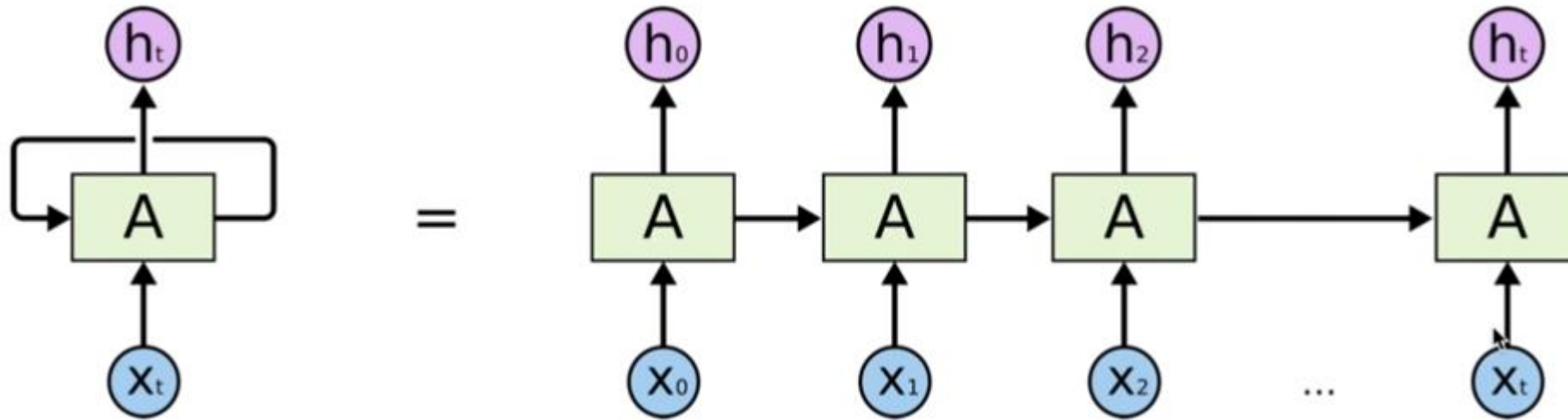
입력: 나는 이 주인공이 연기를 잘해서 이 영화 좋다

**피드 포워드:** 앞으로의 징후를 계산에 의해 예측하고 그 정보에 기준하여 제어를 행하는 방식

**워드 임베딩(word embedding)** 모델은 의미론 측면에서 서로 비슷한 단어끼리 서로 가까운 점에 사상(mapping) 하는 방식으로 각 단어의 벡터를 학습해 이러한 문제를 해결한다. 또한, 우리는 단어 주머니 모델을 사용할 때보다 훨씬 더 작은 벡터 공간에서 전체 어휘를 표현할 것이다. 이렇게 하면 차원을 줄일 수 있고, 우리에게 단어의 의미적 가치를 포착하는 더 작고 밀도가 높은 벡터가 남게 된다.

## 참고2) RNN

Sequence data: 하나의 단어로 이해되지 않고, 이전 단어와 결합되어 이해되는 time series  
serial data 학습에 적합



출처: [https://www.youtube.com/watch?v=-SHPG\\_KMukQ&list=PLIMkM4tgfjnLSOjrEJN31gZATbcj\\_MpUm&index=41](https://www.youtube.com/watch?v=-SHPG_KMukQ&list=PLIMkM4tgfjnLSOjrEJN31gZATbcj_MpUm&index=41)

## 참고2) RNN

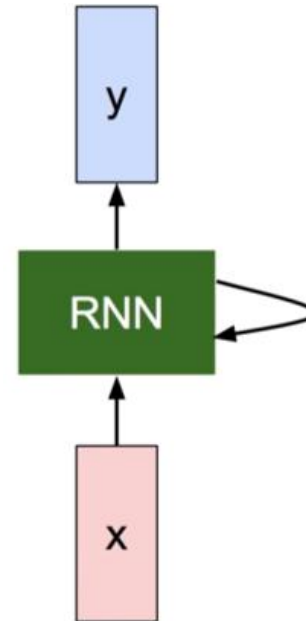
old state 와 입력값(x)을 input 으로 새로운 state 를 도출함  
연산하는 function(fw) 은 동일

We can process a sequence of vectors  $\mathbf{x}$  by applying a recurrence formula at every time step:

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state / old state input vector at some time step

some function with parameters  $W$



### (Vanilla) Recurrent Neural Network

The state consists of a single "hidden" vector  $\mathbf{h}$ :

$$h_t = f_W(h_{t-1}, x_t)$$



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

출처: [https://www.youtube.com/watch?v=-SHPG\\_KMUKQ&list=PLIMkM4tgfjnLSOjrEJN31gZATbcj\\_MpUm&index=41](https://www.youtube.com/watch?v=-SHPG_KMUKQ&list=PLIMkM4tgfjnLSOjrEJN31gZATbcj_MpUm&index=41)

# 참고2) RNN

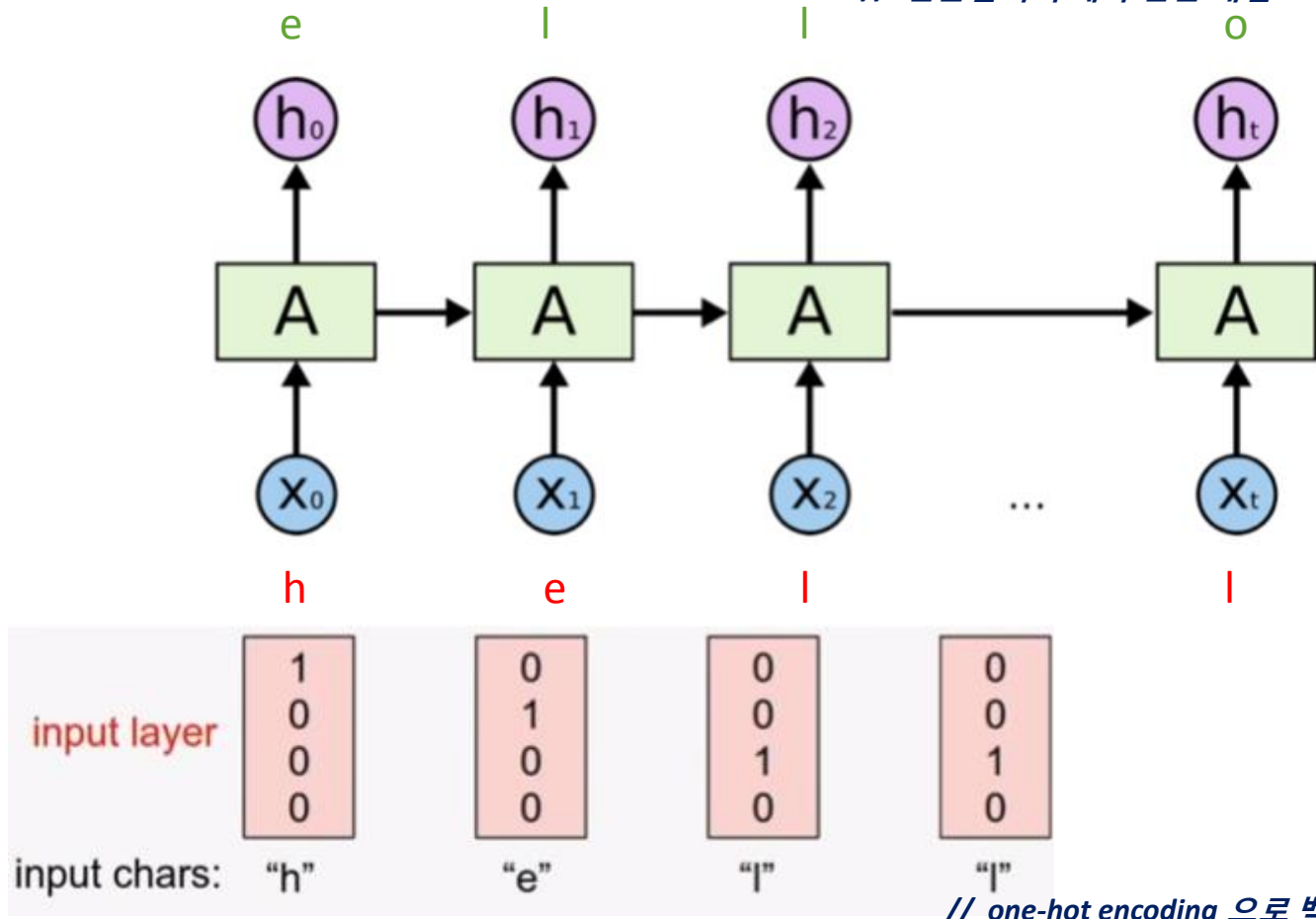
Character-level  
language model  
example

Vocabulary:  
[h,e,l,o]

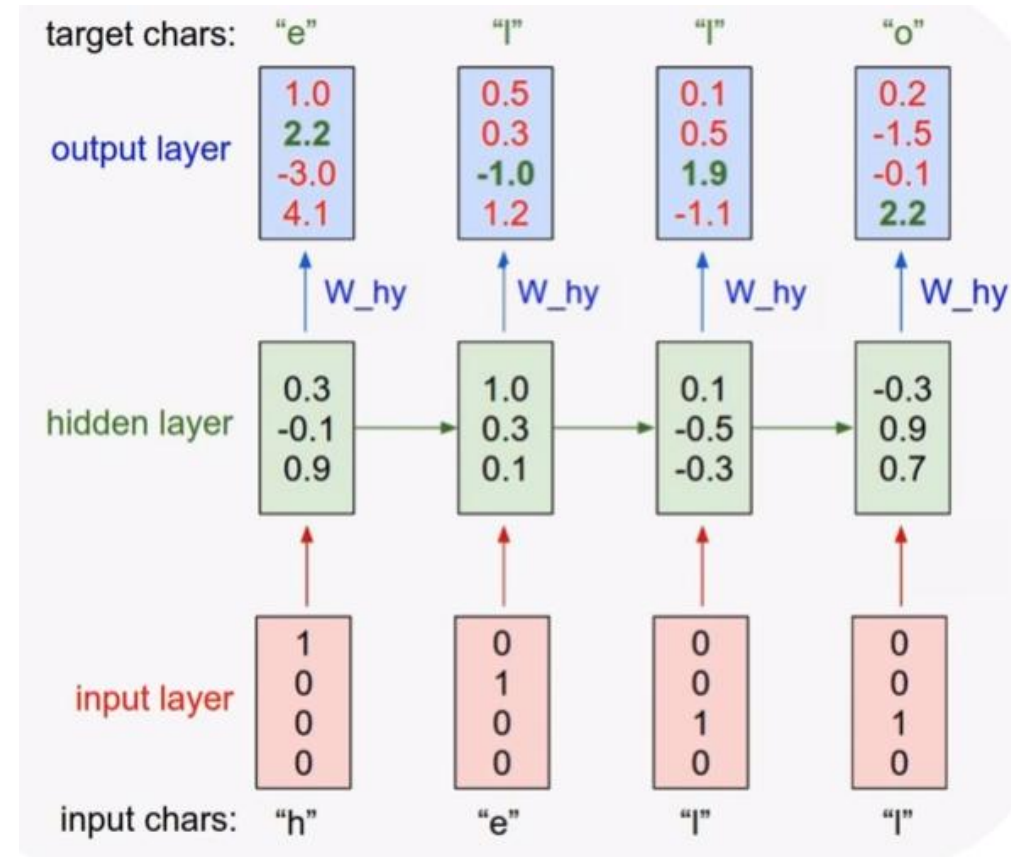
Example training  
sequence:  
"hello"

character-level model 의 경우, 입력 h, e, l, l, o 를 넣었을 때, 그 다음 단어가 예측되는 시스템

// 연관검색어 예측 같은 개념으로 이해



// one-hot encoding 으로 벡터로 표현

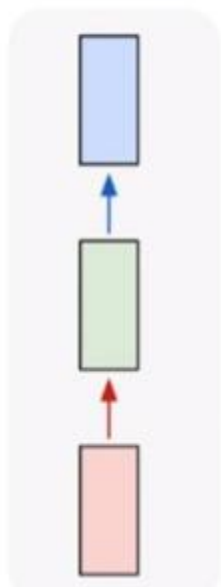


출처: [https://www.youtube.com/watch?v=-SHPG\\_KMUKQ&list=PLIMkM4tgfjnLSOjrEJN31gZATbcj\\_MpUm&index=41](https://www.youtube.com/watch?v=-SHPG_KMUKQ&list=PLIMkM4tgfjnLSOjrEJN31gZATbcj_MpUm&index=41)

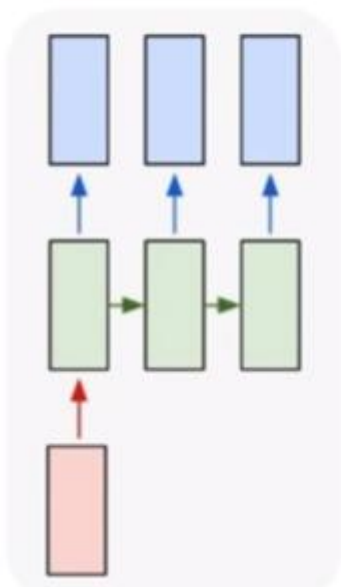


### Vanilla Neural Networks

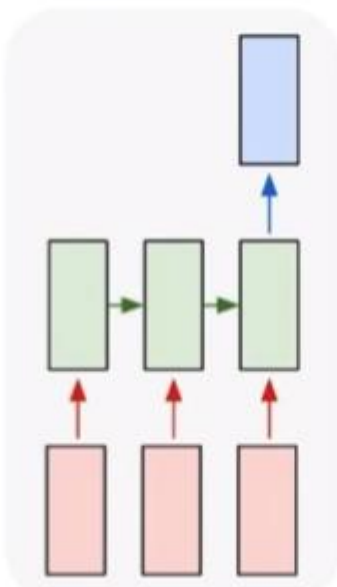
one to one



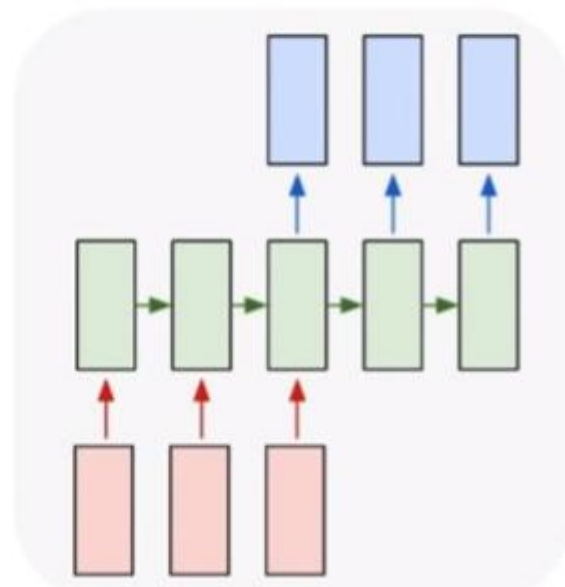
one to many



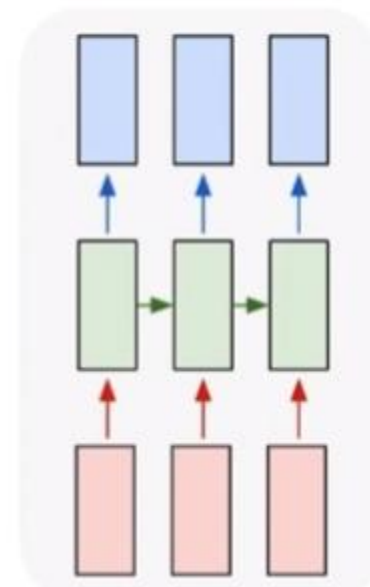
many to one



many to many



many to many



e.g. **Sentiment Classification**  
sequence of words -> sentiment

e.g. **Video classification on frame level**

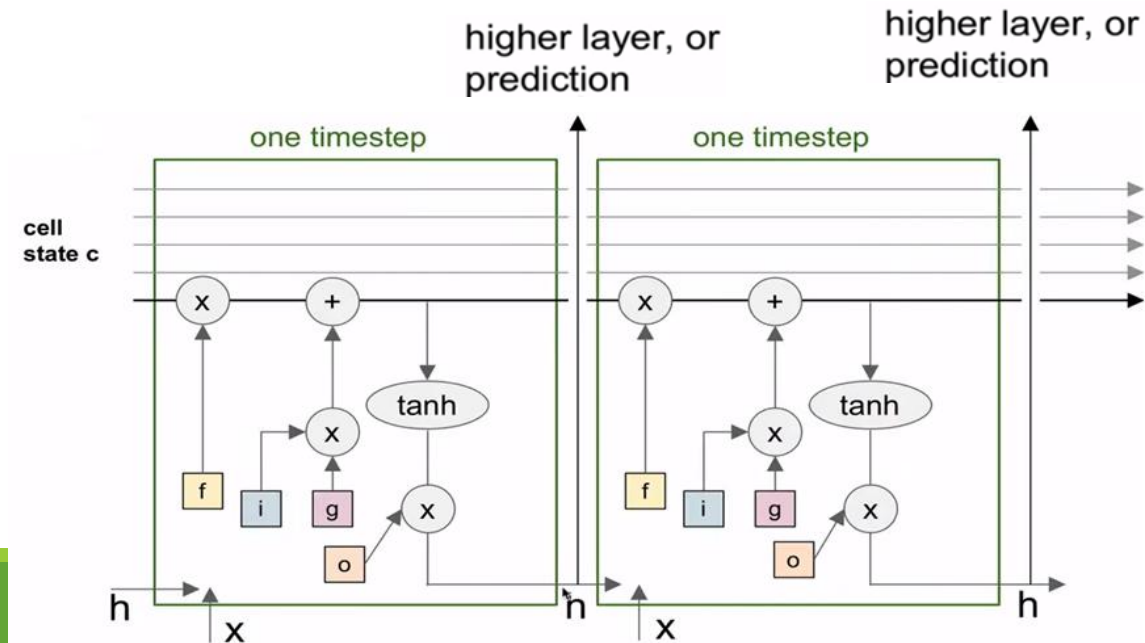
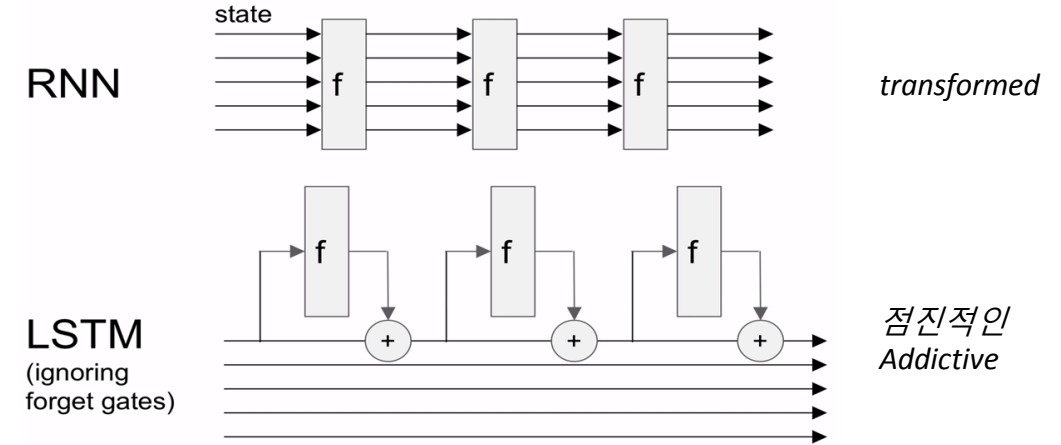
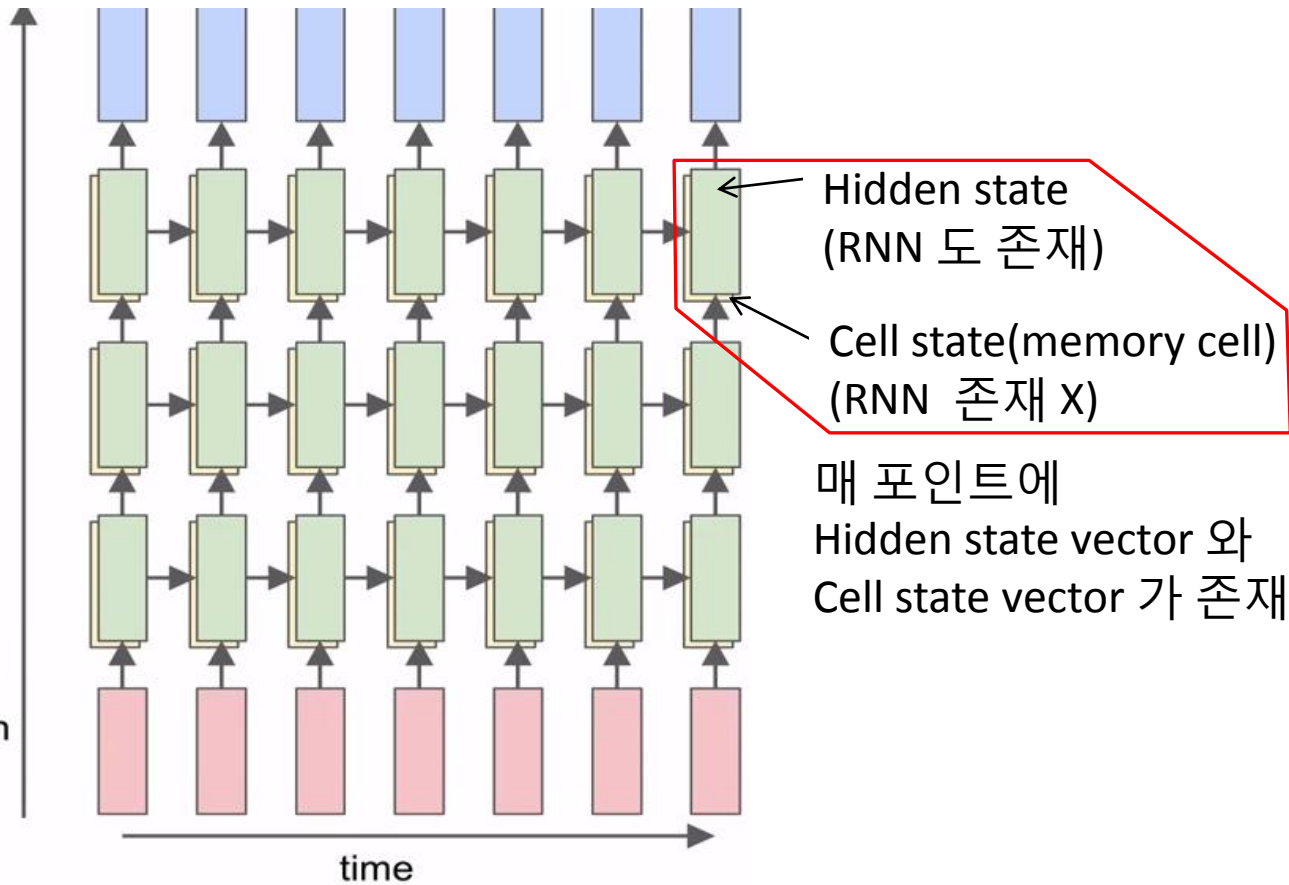
e.g. **Image Captioning**  
image -> sequence of words

e.g. **Machine Translation**  
seq of words -> seq of words

# 04. 모델링 소개 - RNN(순환 신경망)

RNN 이 대용량 학습 시 Gradient Vanishing / Exploding Problem 등 발생 **비효율적**

대안: LSTM(Long Short Term Memory), GRU





## 참고2) LSTM

RNN:

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$h \in \mathbb{R}^n$        $W^l [n \times 2n]$

LSTM:

$$W^l [4n \times 2n]$$

[Cell state vector]

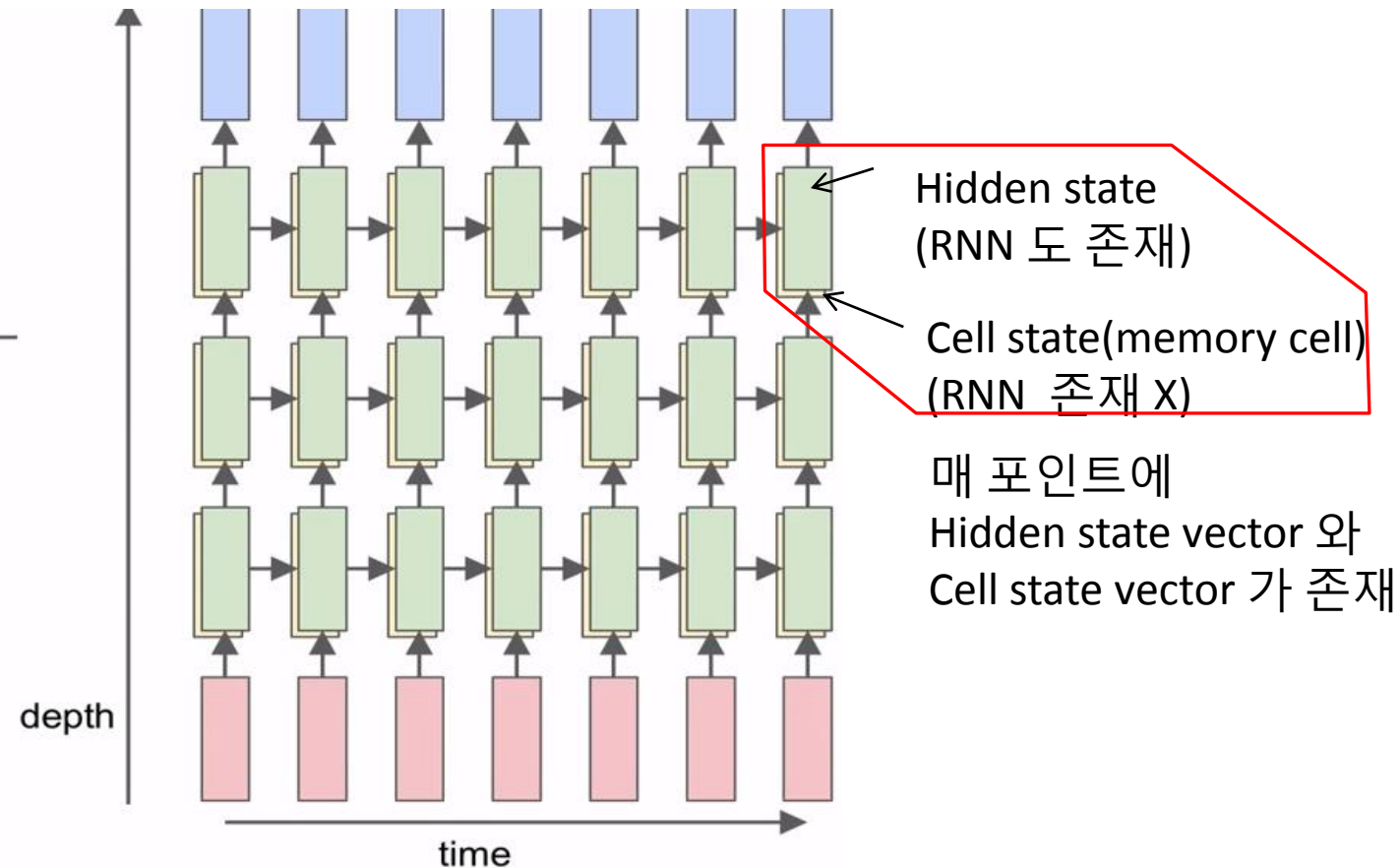
gate

Input  
Forget  
Output  
g

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$



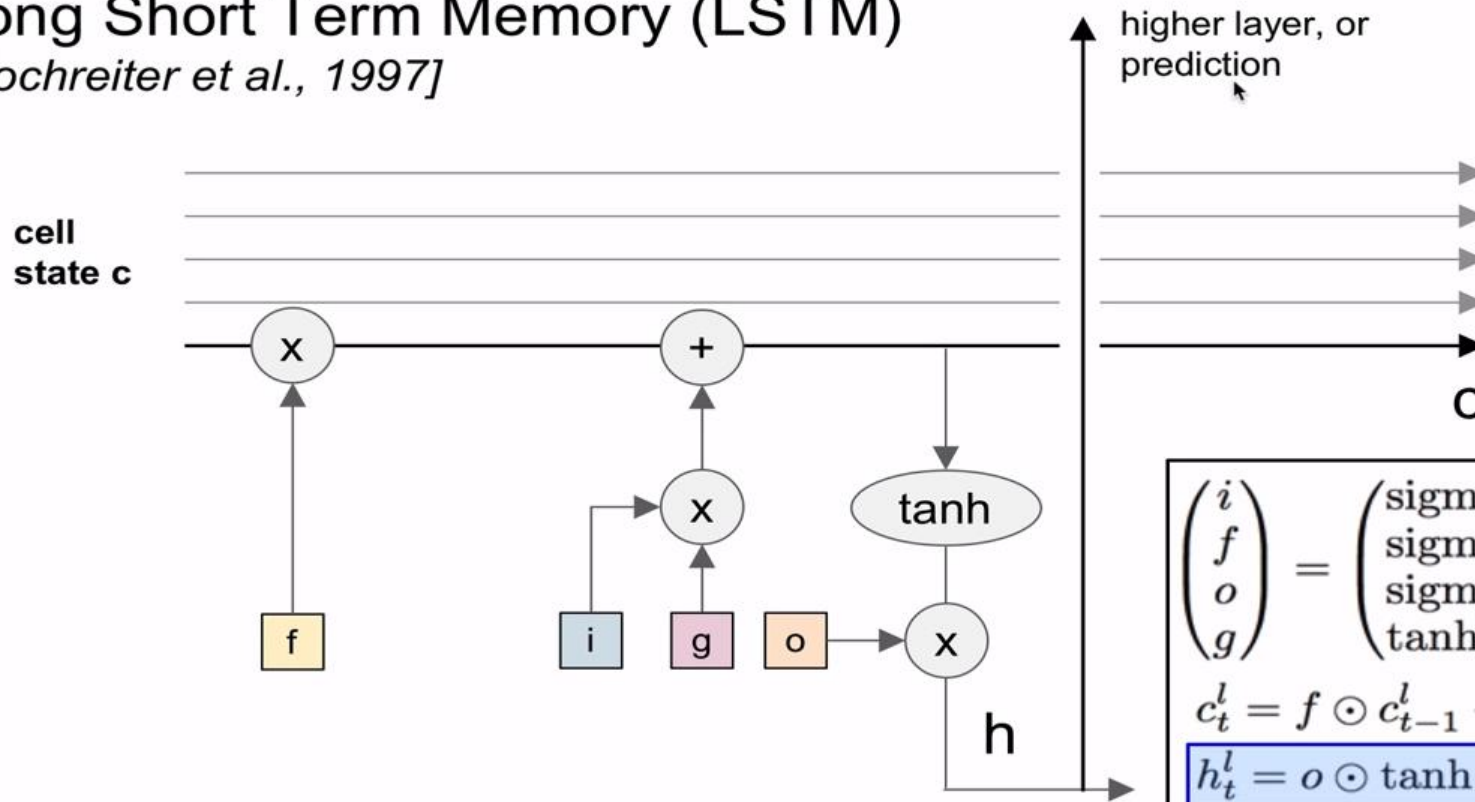
현재의 Cell state = 바로전의 cell state(t-1)를 얼마나 잊을것인가(f)  
+ input (sigmoid: 0~1)과 g(tanh: -1~1) : input을 얼마나 반영시킬것인가  
Hidden state vector = output 에 현재의 cell state의 tanh - 노란색이 녹색에 영향을 줌

## 참고2) LSTM

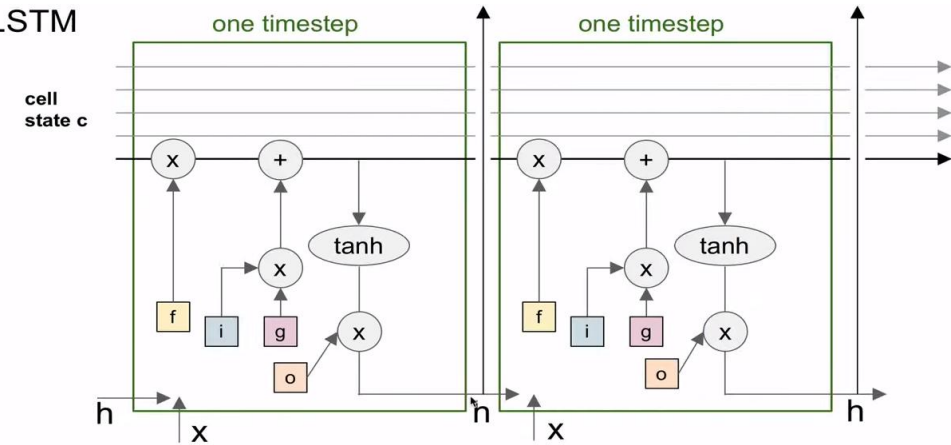
Cell state 가 반영된 h 는 다음 레이어에 전달될 수도, 상위 레벨로 전달될 수도, 예측으로 갈 수도 있음

### Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



LSTM



## 04. 모델링 소개 – RNN(순환 신경망)

학습 데이터 불러오기 – 앞에 전처리 해놓은 데이터 활용

```
In [1]: ▶ import tensorflow as tf
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split

import os
import json
```

```
In [2]: ▶ DATA_IN_PATH = './data_in/'
DATA_OUT_PATH = './data_out/'

TRAIN_INPUT_DATA = 'train_input.npy'
TRAIN_LABEL_DATA = 'train_label.npy'
DATA_CONFIGS = 'data_configs.json'
```

```
In [3]: ▶ input_data = np.load(open(DATA_IN_PATH + TRAIN_INPUT_DATA, 'rb'))
label_data = np.load(open(DATA_IN_PATH + TRAIN_LABEL_DATA, 'rb'))
prepro_configs = None

with open(DATA_IN_PATH + DATA_CONFIGS, 'r') as f:
    prepro_configs = json.load(f)
```

```
In [4]: ▶ TEST_SPLIT = 0.1           // 학습 데이터와 검증 데이터셋 분리
RANDOM_SEED = 13371447

train_input, test_input, train_label, test_label = train_test_split(input_data, label_data,
                                                                    test_size=TEST_SPLIT, random_state=RANDOM_SEED)
```

# 04. 모델링 소개 – RNN(순환 신경망)

## 데이터 입력 함수 구현

In [5]:

```
BATCH_SIZE = 16
NUM_EPOCHS = 3

def mapping_fn(X, Y):
    inputs, labels = {'x': X}, Y
    return inputs, labels

def train_input_fn():
    dataset = tf.data.Dataset.from_tensor_slices((train_input, train_label))
    dataset = dataset.shuffle(buffer_size=50000)
    dataset = dataset.batch(BATCH_SIZE)
    dataset = dataset.repeat(count=NUM_EPOCHS)
    dataset = dataset.map(mapping_fn)
    iterator = dataset.make_one_shot_iterator()

    return iterator.get_next()

def eval_input_fn():
    dataset = tf.data.Dataset.from_tensor_slices((test_input, test_label))
    dataset = dataset.map(mapping_fn)
    dataset = dataset.batch(BATCH_SIZE * 2)
    iterator = dataset.make_one_shot_iterator()

    return iterator.get_next()
```

// tf.data

데이터의 셔플, 배치, 반복 등 여러 기능을  
간단하게 사용할 수 있게 해줌

// tf.data.Dataset.from\_tensor\_slices

주어진 train\_input과 train\_label 을  
묶어 조각으로 만들고 함께 사용할 수 있게 함

// make\_one\_shot\_iterator

데이터를 하나씩 사용할 수 있게 해줌  
iterator.get\_next를 호출하면 하나씩 나옴

// 학습 input

// 평가 input

## 04. 모델링 소개 – RNN(순환 신경망)

모델 함수 정의 > 하이퍼파라미터 정의

```
In [6]: ▶ VOCAB_SIZE = prepro_configs['vocab_size']+1 // 단어 사전에 대한 크기 지정
          WORD_EMBEDDING_DIM = 100                  각 변수의 차원과
          HIDDEN_STATE_DIM = 150
          DENSE_FEATURE_DIM = 150

          learning_rate = 0.001                     학습률 지정
```

```
In [7]: ▶ print(len(prepro_configs['vocab']), VOCAB_SIZE)
```

74065 74066

모델 구현

## 04. 모델링 소개 – RNN(순환 신경망)

// 모델 구현  
(1/2)

In [8]:

```
def model_fn(features, labels, mode):
    TRAIN = mode == tf.estimator.ModeKeys.TRAIN
    EVAL = mode == tf.estimator.ModeKeys.EVAL
    PREDICT = mode == tf.estimator.ModeKeys.PREDICT

    embedding_layer = tf.keras.layers.Embedding(
        VOCAB_SIZE,
        WORD_EMBEDDING_DIM)(features['x'])

    embedding_layer = tf.keras.layers.Dropout(0.2)(embedding_layer)

    rnn_layers = [tf.nn.rnn_cell.LSTMCell(size) for size in [HIDDEN_STATE_DIM, HIDDEN_STATE_DIM]]
    multi_rnn_cell = tf.nn.rnn_cell.MultiRNNCell(rnn_layers)

    outputs, state = tf.nn.dynamic_rnn(
        cell=multi_rnn_cell,
        inputs=embedding_layer,
        dtype=tf.float32)

    outputs = tf.keras.layers.Dropout(0.2)(outputs)
    hidden_layer = tf.keras.layers.Dense(DENSE_FEATURE_DIM, activation=tf.nn.tanh)(outputs[:, -1, :])
    hidden_layer = tf.keras.layers.Dropout(0.2)(hidden_layer)
    logits = tf.keras.layers.Dense(1)(hidden_layer)
    logits = tf.squeeze(logits, axis=-1)
```

// 임베딩 층

모델에서 배치 데이터 받게 되면 인덱스로 구성된 시퀀스 형태  
텐서플로 embedding 함수 호출

// 임베딩 층을 거친 데이터는 RNN 층을 거쳐 문장의 의미 벡터를 출력  
LSTM 모델 활용

LSTMCell 을 여러 개 생성해 하나의 리스트를 만들(MultiRNN)

// dynamic\_rnn 으로 for 문 없이 자동으로 순환 신경망 만들

// 하이퍼볼릭 탄젠트

// 감정의 긍/부정을 판단하기 위한 하나의 출력값 필요  
Dense로 차원변환

// tf.keras.layers.Dense  
신경망 구조의 가장 기본적인

$$y = f(Wx + b)$$

수식을 만족하는 기본적인  
신경망 형태 층을 만드는 함수

// tf.keras.layers.Dropout  
과적합 방지  
(0.2) 전체 입력 값 중  
20%를 0으로 만들

cf) tf.nn.dropout (0.2)  
80%를 0으로 만들

## 04. 모델링 소개 – RNN(순환 신경망)

// 모델 구현  
(2/2)  
> 학습

```
sigmoid_logits = tf.nn.sigmoid(logits) // sigmoid - 출력을 0~1 사이값으로 정의
```

```
if PREDICT:
    predictions = {'sentiment': sigmoid_logits}
```

// 모델 예측

```
return tf.estimator.EstimatorSpec(
    mode=mode,
    predictions=predictions)
```

```
loss = tf.losses.sigmoid_cross_entropy(labels, logits) // loss 도출
```

```
if EVAL:
```

```
    accuracy = tf.metrics.accuracy(labels, tf.round(sigmoid_logits)) // 예측값과 정답 라벨 일치도
    eval_metric_ops = {'acc': accuracy}
```

// 모델 평가

```
return tf.estimator.EstimatorSpec(mode, loss=loss, eval_metric_ops=eval_metric_ops)
```

```
if TRAIN:
```

```
    global_step = tf.train.get_global_step() // 현재 학습의 global_step 도출
    train_op = tf.train.AdamOptimizer(learning_rate).minimize(loss, global_step)
```

// 모델 예측

// Adam Optimizer minimize(경사하강법)

```
return tf.estimator.EstimatorSpec(
    mode=mode,
    train_op=train_op,
    loss=loss)
```

# 04. 모델링 소개 - RNN(순환 신경망)

// tf.estimator  
고수준 API로 모델 구현에만  
집중할 수 있는 환경 제공

// estimator 객체를 만들어  
학습, 평가, 예측 등 실행 가능

```
In [9]: ▶ if not os.path.exists(DATA_OUT_PATH):
        os.makedirs(DATA_OUT_PATH)

est = tf.estimator.Estimator(model_fn=model_fn,
                             model_dir=DATA_OUT_PATH + 'checkpoint/rnn')

INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_model_dir': './data_out/checkpoint/rnn', '_tf_random_seed': None, '_save_summary_steps': 100, '_save_
checkpoints_steps': None, '_save_checkpoints_secs': 600, '_session_config': allow_soft_placement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100, '_train_distribute': None, '_device
e fn': None, '_protocol': None, '_eval_distribute': None, '_experimental_distribute': None, '_service': None, '_cluster_spec': <tensor
In [10]: ▶ os.environ["CUDA_VISIBLE_DEVICES"]="4"
        est.train(train_input_fn)
```

// estimator 객체를 만들어  
학습, 평가, 예측 등 실행 가능

```
Use tf.cast instead.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 0 into ./data_out/checkpoint/rnn##model.ckpt.
INFO:tensorflow:loss = 0.69277143, step = 1
INFO:tensorflow:global_step/sec: 1.09267
INFO:tensorflow:loss = 0.68858707, step = 101 (91.543 sec)
INFO:tensorflow:loss = 0.6977031, step = 4201 (61.400 sec)
INFO:tensorflow:Saving checkpoints for 4221 into ./data_out/checkpoint/rnn##model.ckpt.
WARNING:tensorflow:From c:\programdata\anaconda3\envs\tutorial\lib\site-packages\tensorflow\python\training\saver.py:966: remove_che
ckpoint (from tensorflow.python.training.checkpoint_management) is deprecated and will be removed in a future version.
Instructions for updating:
Use standard file APIs to delete files with this prefix.
INFO:tensorflow:Loss for final step: 0.69812787.
```

Out[10]: <tensorflow\_estimator.python.estimator.estimator.Estimator at 0xd120c5d9b0>



## 04. 모델링 소개 – RNN(순환 신경망)

In [12]:

```
est.evaluate(eval_input_fn)
```

// estimator 객체를 만들어  
학습, 평가, 예측 등 실행 가능

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2019-05-05T10:46:10Z
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from ./data_out/checkpoint/rnn#model.ckpt-19627
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2019-05-05-10:46:24
INFO:tensorflow:Saving dict for global step 19627: acc = 0.8672, global_step = 19627, loss = 0.5735552
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 19627: ./data_out/checkpoint/rnn#model.ckpt-19627
```

Out[12]: {'acc': 0.8672, 'loss': 0.5735552, 'global\_step': 19627}

정확도를 성능 기준으로 보기 때문에 출력하게 된다면 다음과 같이 나올 것이다(경우에 따라 결과가 달라질 수 있다).



```
{'accuracy': 0.85565215, 'loss': 0.695675, 'global_step': 1296}
```

이처럼 모델 학습과 성능 측정을 진행하면서 가장 나은 성능의 모델을 만들 수 있다. 이렇게 모델 학습이 완료되면 캐글에 제출할 데이터를 만들어야 한다.

## 04. 모델링 소개 - RNN(순환 신경망)

Out[12]: {'acc': 0.8672, 'loss': 0.573552, 'global\_step': 19627}

{'accuracy': 0.85565215, 'loss': 0.695675, 'global\_step': 1296}

// 책 Score = 0.93818

Out[14]: {'acc': 0.848, 'loss': 0.6912221, 'global\_step': 23848}

Name	Submitted	Wait time	Execution time	Score
movie_review_result_rnn.csv	just now	0 seconds	0 seconds	0.90069

Complete

```
VOCAB_SIZE = prepro_configs['vocab_size']+1
WORD_EMBEDDING_DIM = 100
HIDDEN_STATE_DIM = 150
DENSE_FEATURE_DIM = 150

learning_rate = 0.001
```

학습률  
조정

```
VOCAB_SIZE = prepro_configs['vocab_size']+1
WORD_EMBEDDING_DIM = 100
HIDDEN_STATE_DIM = 150
DENSE_FEATURE_DIM = 150

learning_rate = 0.0005
```

Out[11]: {'acc': 0.5108, 'loss': 0.682688, 'global\_step': 4221}

Name	Submitted	Wait time	Execution time	Score
movie_review_result_rnn.csv	just now	1 seconds	0 seconds	0.58268

Complete

[Jump to your position on the leaderboard](#) ▾

# Q & A

---