

Day 12

Text Analytics

HTML Tags

- HTML documents are consists of a lot of tags that tell a web browser how to display the web page. **Html은 웹페이지를 출력하기 위한 태그들로 구성되어 있음**
- Opening tag tells the browser where the instruction begins e.g., <p>, and closing tag tells it where the instruction ends. e.g., </p> **오픈태그와 클로징태그가 짝지어져 있음**
- Some tags don't have closing tags, like , <input>, and
 tags **혼자서 존재하는 태그도 있음**

Structure of HTML Document

```
<html>  
  <head>  
    { head content  
  </head>  
  <body>  
    { body content  
  </body>  
</html>
```

- Two main sections of an HTML document are the head and the body

Types of HTML

Version	Date	Description
HTML1.0	1989	The first public version of HTML.
HTML 2.0	1995	Added interactive elements including Web forms.
HTML 3.0	1996	A proposed replacement for HTML 2.0 that was never widely adopted.
HTML 3.2	1997	Included additional support for Web tables and expanded the options for interactive form elements and a scripting language.
HTML 4.01	1999	Added support for style sheets to give Web designers greater control over page layout and appearance, and provided support for multimedia elements such as audio and video. Current browsers support almost all of HTML 4.01.
XHTML 1.0	2001	A reformulation of HTML 4.01 in the XML language in order to provide enforceable standards for HTML content and to allow HTML to interact with other XML languages.
XHTML 1.1	2002	A minor update to XHTML 1.0 that allows for modularity and simplifies writing extensions to the language.
XHTML 2.0	discontinued	The follow-up version to XHTML 1.1 designed to fix some of the problems inherent in HTML 4.01 syntax. Work on this version was discontinued in 2009 due to lack of browser support.
HTML 5.0	In development	An update to HTML 4.01 that provides support for a variety of new features including semantic page elements, column layout, form validation, offline storage, and enhanced multimedia.
XHTML 5.0	In development	A version of HTML 5.0 written under the XML language; unlike XHTML 2.0, XHTML 5.0 will be backward compatible with XHTML 1.1.

- For example, HTML 4.01 Transitional, HTML 4.01 Strict, XHTML 1.0 Transitional, XHTML 1.0 Strict, etc.
- HTML5's doctype is short and simple!

```
<!DOCTYPE html>
```

Character Sets

ASCII

- American Standard Code for Information Interchange
- Used for the alphabet of English characters

Latin-1 or ISO 8859-1

- More extended character sets supporting 255 characters
- Used by most languages that employ the Latin alphabet (e.g., English, French, Spanish, and Italian)

Unicode

- The most extended character set supporting up to 65,536 symbols
- Used for any of the world's languages.

UTF-8

- Most commonly used languages on the Web
- A compressed version of Unicode
- Default character set assumed by the browser.

- Before adding special characters to web pages, you need to specify the character set you are using. `<meta charset="UTF-8">`

Page Headings

- Tags like `<h1>` and `<h2>` denote headlines and assign them relative importance

```
<h1>Heading 1</h1>
```

```
<h2>Heading 2</h2>
```

```
<h3>Heading 3</h3>
```

```
<h4>Heading 4</h4>
```

```
<h5>Heading 5</h5>
```

```
<h6>Heading 6</h6>
```

Paragraphs

- The `<p>` tag indicates a basic paragraph of information.
- You tell a web browser where a paragraph of text begins with a `<p>` (opening paragraph tag), and where it ends with a `</p>` (closing paragraph tag).

```
<p>This is a  
paragraph.</p>
```

```
<p>This is a  
paragraph.</p>
```

```
<p>This is a  
paragraph.</p>
```

Inserting Line Breaks

- The `
` tag marks line break!

`<p>`

To break lines
in a text,
use the `br` element.

`</p>`

Forms

- The HTML `<form>` element defines a form that is used to collect user input
- The action attribute defines the action to be performed when the form is submitted.
- The method attribute specifies the HTTP method (GET or POST) to be used when submitting the form data

```
<form  
action="/action_page.php  
" method="post"> </form>
```

GET vs. POST

```
/action_page.php?firstname=Mickey&lastname=Mouse
```

GET

- visible in the page address field

POST

- does not display the submitted form data in the page address field

Inputs

- The `<input>` element is the most important form element.

<code><input type="text"></code>	Defines a one-line text input field
--	-------------------------------------

<code><input type="radio"></code>	Defines a radio button (for selecting one of many choices)
---	--

<code><input type="submit"></code>	Defines a submit button (for submitting the form)
--	---

```
First name:<br>
<input type="text"
name="firstname"
value="Mickey"><br>
Last name:<br>
<input type="text"
name="lastname"
value="Mouse"><br>
<input type="submit"
value="Submit">
```

Unordered Lists

- A `` tag identifies a bulleted list (e.g., a list of recipe ingredients).
- A `` tag marks an item in the list.

```
<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>
```

Ordered Lists

- The list items will be marked with numbers by default

```
<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

Tables

- Each table row is defined with the `<tr>` tag.
- A table data/cell is defined with the `<td>` tag.
- A table header is defined with the `<th>` tag. By default, table headings are bold and centered.

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Images

- The `` tag is used to insert images into the web page.

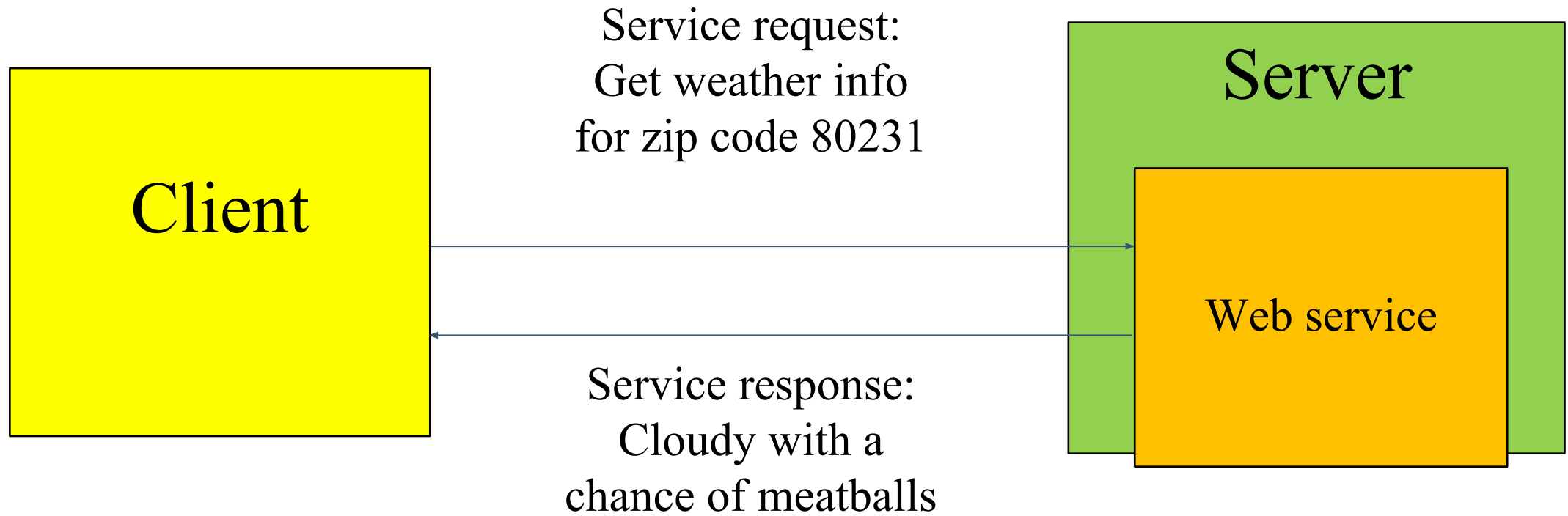
```

```

Web Service

A way for two applications or electronic devices to communicate over a network.

Example



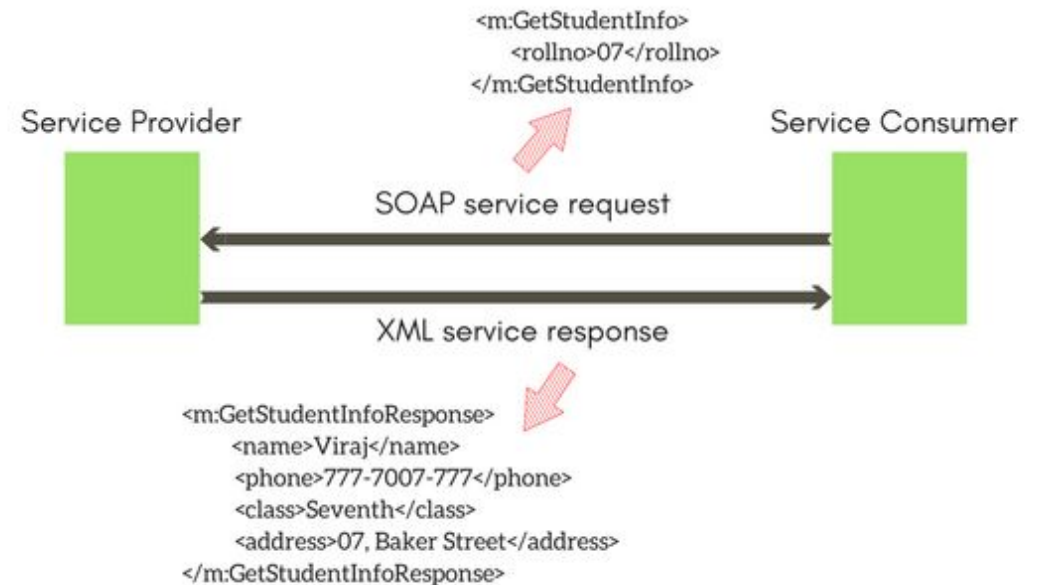
Types of Web Services

SOAP Web
Services

REST Web
Services

SOAP Web Services

- Stands for Simple Object Access Protocol.
- A technique to send an XML request over the Internet using HTTP protocol (hitting a URL), and in return getting an XML response.



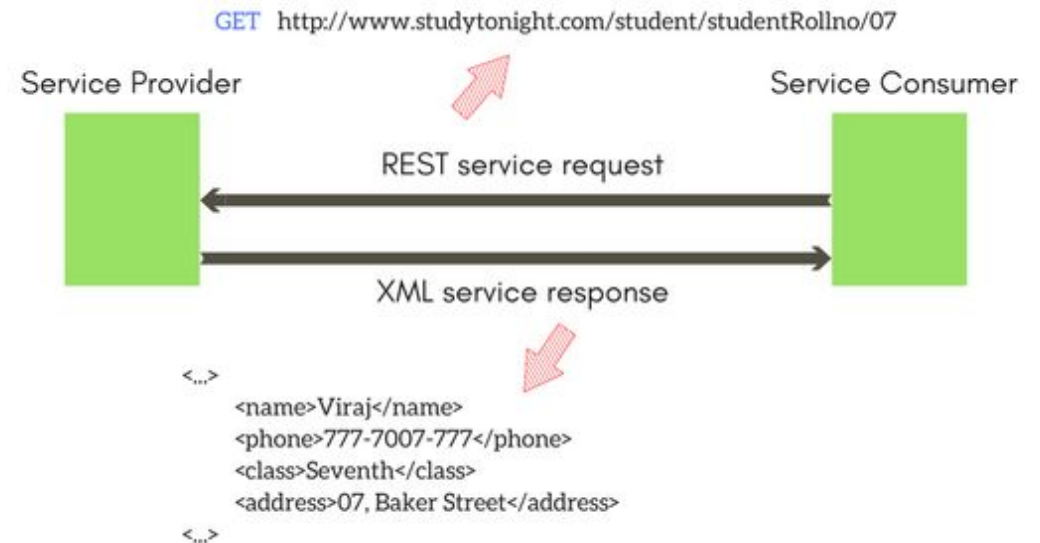
WSDL

- Stands for Web Service Description Language.
- An XML that describes all the methods available in the web service, along with the request and response types.
- Every application serving SOAP requests has a WSDL file.
- It describes the contract between service and client.

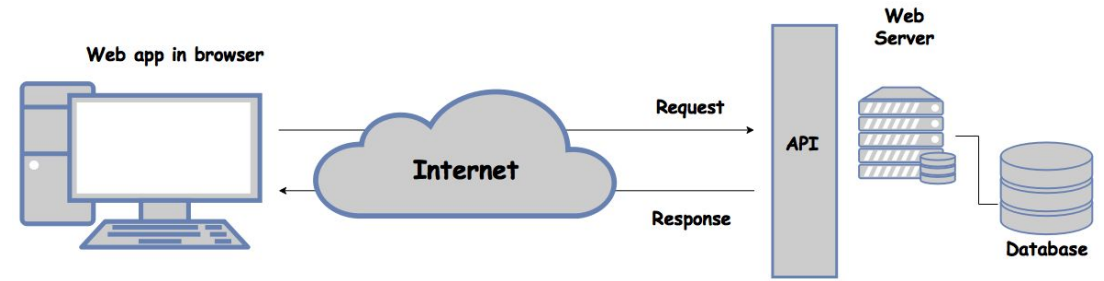
REST Web Services

- Stands for Representational State Transfer.
- Not a set of standards or rules, rather it is a style of software architecture

The REST service locates the resource based on the URL and performs the action based on the transport action verb



API



- Application Program Interface consumed by a client application
- The way for an application to interact with certain system, application, or library
- For example, API's for OS (WinAPI), API's for databases or other applications, and API's for specific libraries like image processing


Web API

- An application programming interface for either a web server or a web browser

공공데이터

DATA 공공데이터포털
GO . KR

[바로가기](#)

 서울 열린데이터 광장
SEOUL OPEN DATA PLAZA


[바로가기](#)

KOSIS 국가통계포털
Korean Statistical Information Service

[바로가기](#)

 **aT FIS** 식품산업통계정보
FOOD INFORMATION STATISTICS SYSTEM

[바로가기](#)

 **한국도로공사**
고속도로 공공데이터 포털

[바로가기](#)

 **SGIS^{plus}**
통계지리정보서비스

[바로가기](#)

국토교통부
 **국가교통정보센터**
National Transport Information Center

[바로가기](#)

 **국가공간정보포털**
국토교통부 Korea National Spatial Data Infrastructure Portal

[바로가기](#)

 **농림축산식품**
공공데이터 포털
농림축산식품부

[바로가기](#)

 **보건복지데이터포털**
Health and Welfare Data Portal

[바로가기](#)

정보공개 & 공공데이터개방
재난안전데이터포털

[바로가기](#)

특허정보넷 **키프리스**

[바로가기](#)

 **경기도**
경기데이터드림

[바로가기](#)

 **부산** 공공데이터포털

[바로가기](#)

 **전라북도** 공공데이터포털

[바로가기](#)

 **경상북도** 공공데이터포털시스템

[바로가기](#)

공공데이터 포털 Open API



- 공공데이터 포털 □
데이터셋 □ 오픈API □
온라인가격정보

(<https://www.data.go.kr/search/index.do?index=OPENAPI&query=¤tPage=1&countPerPage=10>)

xml Page

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <response>
  - <body>
    <ic>A019170</ic>
    <in>냉 동식품</in>
  - <items>
    - <item>
      <pi>999334384</pi>
      <pn>월드프랑스 까망베르(125g)</pn>
      <sp>11930.0</sp>
      <dp/>
      <bp/>
      <sd>2015-01-01</sd>
    </item>
    - <item>
      <pi>99781659</pi>
      <pn>플루원 옛두부 300g*2</pn>
      <sp>3861.0</sp>
      <dp/>
      <bp/>
      <sd>2015-01-01</sd>
    </item>
    - <item>
      <pi>996779248</pi>
      <pn>피자 소스(피자푸드 태원 2K)</pn>
      <sp>6170.0</sp>
      <dp/>
      <bp/>
      <sd>2015-01-01</sd>
    </item>
    - <item>
      <pi>996514461</pi>
      <pn>코리원[패스츄리도우 25X25(5개입)쌀도우/오곡도우/녹차 곡물도우/9인치/12인치/임실/피자재료</pn>
      <sp>9510.0</sp>
      <dp/>
      <bp/>
      <sd>2015-01-01</sd>
    </item>
    - <item>
      <pi>996182902</pi>
      <pn>월남쌈튀김겸용 원형22Cm 500g X 1 가공식품 썸 월남</pn>
      <sp>6540.0</sp>
      <dp/>
```

- 계정 만든후 마이페이지
 - 개발계정 □ 온라인가격정보
 - 개발계정 상세보기
- 상세기능정보 □ 미리보기
- 다운로드 □ 실행 □ 미리보기
- 클릭

XML (Extensible Markup Language)

```
<?xml version="1.0" standalone="yes" ?>
- <shop location="Birmingham" size="Large">
  - <food>
    <Name>Apple</Name>
    <type>fruit</type>
    <cost>15</cost>
  </food>
  - <food>
    <Name>Carrot</Name>
    <type>vegetable</type>
    <cost>10</cost>
  </food>
</shop>
```

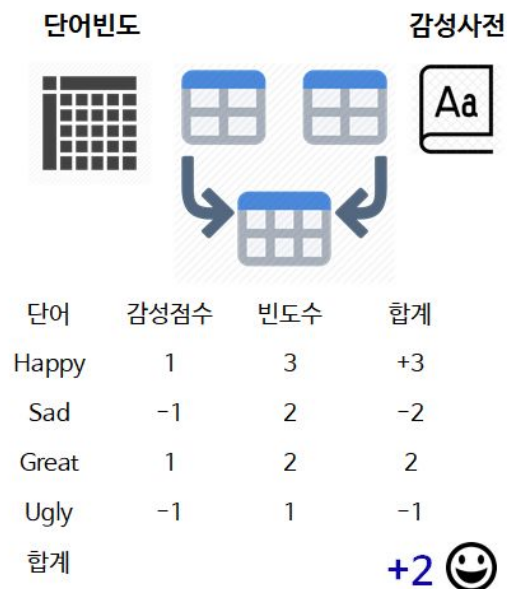
- A text-based format that allows for the structuring of electronic documents and is not limited to a set of labels.

Exercise #12

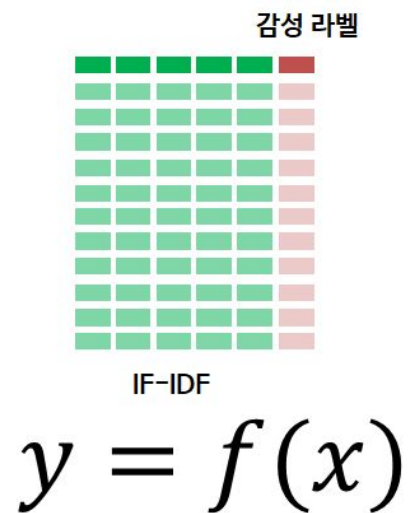
- Request some 공공 데이터, parse them, append them to lists, and create a data frame with the lists, and save it as an excel file
- Refer to <https://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-python/> and <https://www.dataquest.io/blog/web-scraping-tutorial-python/> for parsing xml

Type of Sentiment Analysis

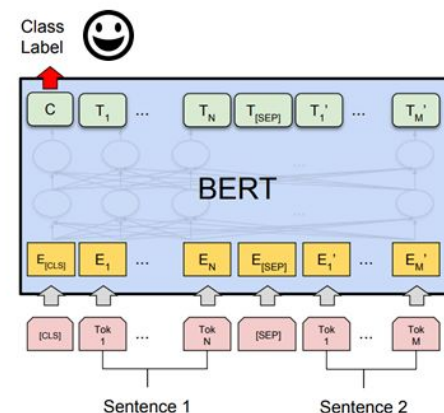
규칙/어휘 기반
감성분석



M/L 기반
감성분석



딥러닝 기반
감성분석



Word Embedding

- The collective name for a set of language modeling and feature learning techniques where words or phrases from the vocabulary are mapped to vectors of real numbers
단어나 구절을 숫자의 벡터로 바꾸는 방법을 총칭함
- The technique is primarily used with Neural Network Models
신경망과 함께 사용됨

Bag-of-Words (BoW)

- A way to represent the data in a tabular format with columns representing the total vocabulary of the corpus and each row representing a single observation. The cell (intersection of the row and column) represents the count of the word represented by the column in that particular observation. 단어사전과 비교해서 단어가 몇번 나타났는지 테이블로 정리

	it	is	puppy	cat	pen	a	this
it is a puppy	1	1	1	0	0	1	0
it is a kitten	1	1	0	0	0	1	0
it is a cat	1	1	0	1	0	1	0
that is a dog and this is a pen	0	2	0	0	1	2	1
it is a matrix	1	1	0	0	0	1	0

BoW Code

```
from sklearn.feature_extraction.text import CountVectorizer

corpus = ['This is the first document.',
          'This document is the second document.',
          'And this is the third one.',
          'Is this the first document?']

vectorizer = CountVectorizer()

X = vectorizer.fit_transform(corpus)

print(vectorizer.get_feature_names())

print(X.toarray())

X_df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names())

X_df.head(10)
```


N-Grams

- The process of combining the nearby words together for representation purposes where N represents the number of words to be combined together 옆에 있는 단어들을 함께 묶어주는 과정. N은 합칠 단어의 수
- Example
 - “Natural Language Processing is essential to Computer Science.”

- 1-gram or unigram
 - “Natural, Language, Processing, is, essential, to, Computer, Science”
- Bigram
 - “Natural Language, Language Processing, Processing is, is essential, essential to, to Computer, Computer Science”
- Trigram
 - “Natural Language Processing, Language Processing is, Processing is essential, is essential to, essential to Computer, to Computer Science”

N-Gram Code

```
vectorizer2 = CountVectorizer(analyzer='word',  
ngram_range=(2, 2))  
X2 = vectorizer2.fit_transform(corpus)  
print(vectorizer2.get_feature_names())  
print(X2.toarray())  
X2_df = pd.DataFrame(X2.toarray(),  
columns=vectorizer2.get_feature_names())  
X2_df.head(10)
```

Stopwords and Lowercase

```
from sklearn.feature_extraction.text import  
CountVectorizer  
  
from nltk.tokenize import RegexpTokenizer  
  
token = RegexpTokenizer(r'[a-zA-Z0-9]+')  
  
cv =  
CountVectorizer(lowercase=True, stop_words='english', ngram  
_range = (1,1), tokenizer = token.tokenize)  
  
text_counts= cv.fit_transform(data['Phrase'])
```

RegexTokenizer

```
from nltk.tokenize import RegexTokenizer  
  
sentence = "Think and wonder, wonder and think."  
  
token = RegexTokenizer(r"\w+") #similar to tokens =  
re.split('\W+', text)  
  
new_words = token.tokenize(sentence)  
  
print(new_words)
```

TF-IDF

- A way of scoring the vocabulary so as to provide adequate weight to a word in proportion of the impact it has on the meaning of a sentence.

문장에서의 의미에 따라 단어에 가중치를 주는 방법

TF-IDF Formula

- The score is a product of 2 independent scores, term frequency(tf) and inverse document frequency (idf) TF와 IDF를 곱한 값

$$w_{i,j} = tf_{i,j} \times \log \left(\frac{N}{df_i} \right)$$

$tf_{i,j}$ = number of occurrences of i in j

df_i = number of documents containing i

N = total number of documents

Log (N/d) where, N is total number of documents and d is the number of documents in which the word appears.

전체문서에서 그 단어를 포함하는 문서의 수

TF and IDF

- Term Frequency (TF) - frequency of word in the current document
현재문장에서 단어의 빈도수
- Inverse Document Frequency (IDF) - A measure of how much information the word provides, i.e., if it's common or rare across all documents.
단어가 제공하는 정보의 양을 측정

sentence 1	earth is the third planet from the sun					
sentence 2	Jupiter is the largest planet					
Word	TF (Sentence 1)	TF (Sentence 2)	IDF	TF*IDF (sentence 1)	TF*IDF (Sentence 2)	
earth	1/8	0	$\log(2/1)=0$	0.0375	0	
is	1/8	1/5	$\log(2/2)=0$	0	0	
the	2/8	1/5	$\log(2/2)=0$	0	0	
third	1/8	0	$\log(2/1)=0.3$	0.0375	0	
planet	1/8	1/5	$\log(2/2)=0$	0	0	
from	0	0	$\log(2/1)=0.3$	0	0	
sun	1/8	0	$\log(2/1)=0.3$	0.0375	0	
largest	0	1/5	$\log(2/1)=0.3$	0	0.06	
Jupiter	0	1/5	$\log(2/1)=0.3$	0	0.06	

TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer

corpus = [

    'This is the first document.',

    'This document is the second document.',

    'And this is the third one.',

    'Is this the first document?',

]

vectorizer3 = TfidfVectorizer()

X3 = vectorizer3.fit_transform(corpus)

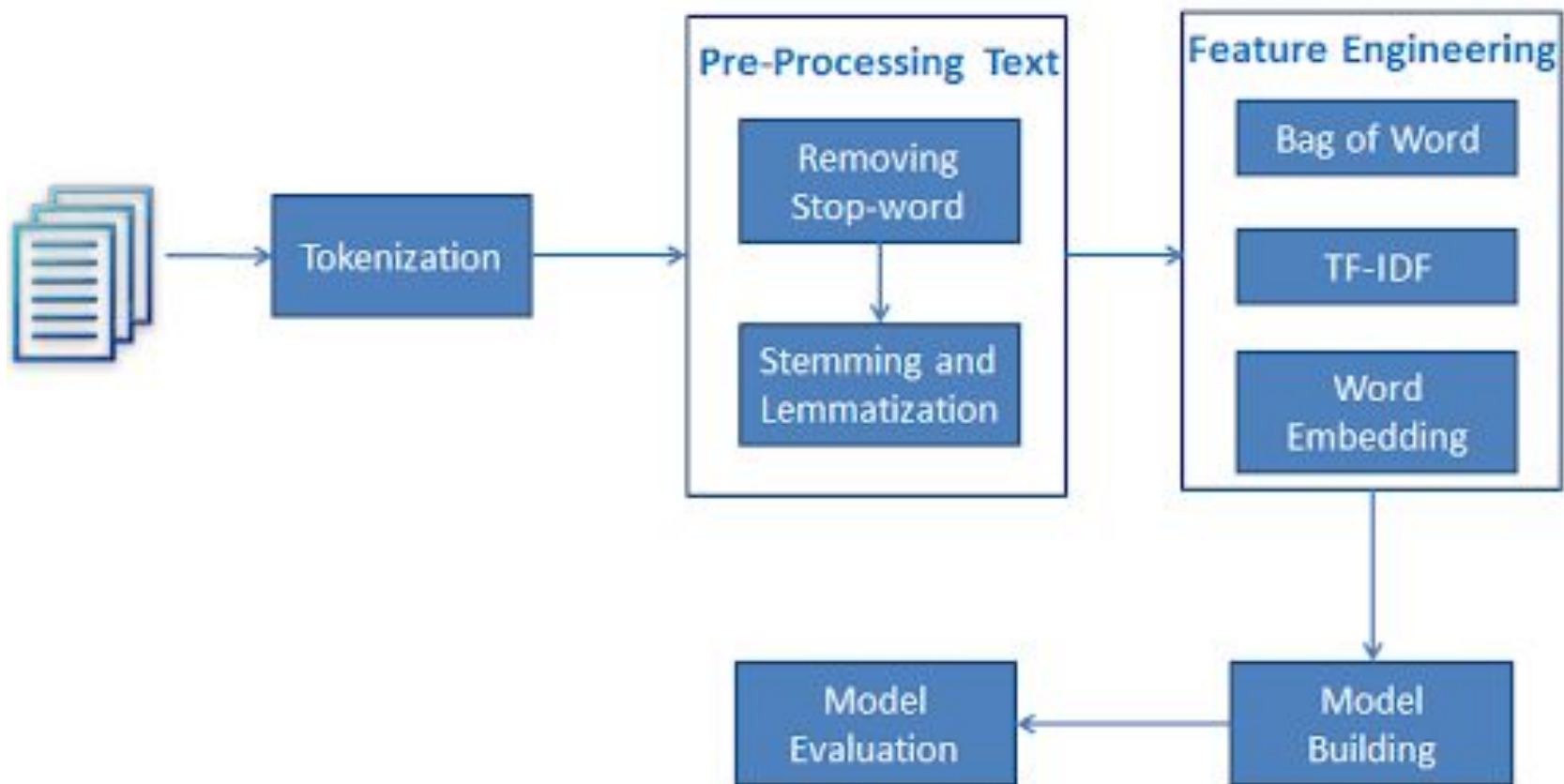
print(vectorizer3.get_feature_names())

print(X3.toarray())

X3_df = pd.DataFrame(X3.toarray(), columns=vectorizer3.get_feature_names())

X3_df.head(10)
```


Text Analytics Process



Naive Bayes 나이트 베이지언

- A classification technique based on Bayes' Theorem with an assumption of independence among predictors.
독립변수들끼리 서로 독립적이라고 가정하는
베이지언 이론에 기초한 분류기술
- Easy to build and particularly useful for very large data sets
데이터가 많을때 유용

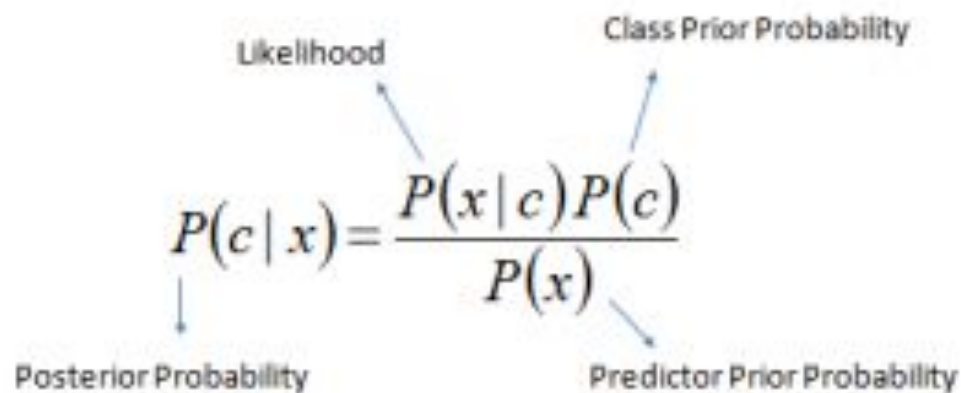
Naive Bayes 나이브 베이지언

- Outperforms even highly sophisticated classification methods
복잡한 분류문제에서 잘 작동함.
- Mostly used in text classification and with problems having multiple classes
클래스가 많은 텍스트분류 문제에 주로 사용됨
- Assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature
속성들끼리 서로 연관성이 없다고 가정함

Example

- A fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple 사과가 사과라고 인식되기 위해서는 빨간색, 둥근 모양, 3인치의 직경을 가져야 됩니다. 각 속성들은 서로 관련이 있고, 다른 속성에 연관되어 있지만, 각각의 속성은 이 과일이 사과라고 인식되는 확률에 독립적으로 영향을 미친다고 가정합니다.
- $P(\text{사과}|\text{빨강})$, $P(\text{사과}|\text{둥근모양})$, $P(\text{사과}|\text{3인치직경})$

Posterior Probability



The diagram shows the formula for posterior probability: $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$. Arrows point from labels to the corresponding parts of the formula: 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

- $P(c|x)$ is the posterior probability of *class* (c , *target*) given *predictor* (x , *attributes*).
주어진 조건에서 그 클래스가 될 확률
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*. 그 클래스에서 그 조건이 나올 확률
- $P(c)$ is the prior probability of *class*. 그 클래스가 나왔던 확률
- $P(x)$ is the prior probability of *predictor*. 전에 그 조건이 나온 확률

Text Classification

- Works particularly well with natural language processing (NLP) problems 언어처리에 유용함

Text	Tag
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports



Word	P(word Sports)	P(word Not Sports)
a	$\frac{2+1}{11+14}$	$\frac{1+1}{9+14}$
very	$\frac{1+1}{11+14}$	$\frac{0+1}{9+14}$
close	$\frac{0+1}{11+14}$	$\frac{1+1}{9+14}$
game	$\frac{2+1}{11+14}$	$\frac{0+1}{9+14}$

$$\begin{aligned}
 &P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times P(Sports) \\
 &= 2.76 \times 10^{-5} \\
 &= 0.0000276
 \end{aligned}$$

$$\begin{aligned}
 &P(a|Not Sports) \times P(very|Not Sports) \times P(close|Not Sports) \times P(game|Not Sports) \times P(Not Sports) \\
 &= 0.572 \times 10^{-5} \\
 &= 0.00000572
 \end{aligned}$$

$$P(sports|a \text{ very close game}) = \frac{P(a \text{ very close game}|sports) \times P(sports)}{P(a \text{ very close game})}$$

Algorithm Example

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Step 1: Convert the data set into a frequency table 현재있는 데이터로 빈도테이블을 만듦

Step 2: Create Likelihood table

확률로 전환하여 가능성테이블을 만듦

Likelihood table				
Weather	No	Yes		
Overcast		4	$=4/14$	0.29
Rainy	3	2	$=5/14$	0.36
Sunny	2	3	$=5/14$	0.36
All	5	9		
	$=5/14$	$=9/14$		
	0.36	0.64		

Step 3: Use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction 각 클래스에 후속확률을 구함. 높은 후속확률을 갖는 클래스가 예측값이 됨

Example

- Will players play if weather is sunny 해가 난 날 나가서 놀 확률은??
 - $P(\text{Yes} \mid \text{Sunny}) = P(\text{Sunny} \mid \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$
 - $P(\text{Sunny} \mid \text{Yes}) = 3/9 = 0.33$ 나가서 논 중에 해가 난 경우
 - $P(\text{Yes}) = 9/14 = 0.64$ 나가서 논 경우
 - $P(\text{Sunny}) = 5/14 = 0.36$ 해가 난 경우
 - $P(\text{Yes} \mid \text{Sunny}) = 0.33 * 0.64 / 0.36 = 0.60$, which has higher probability **60%의 확률이 있음**

Applications

- Disease diagnosis 질병진단
- Sentiment analysis 감성분석
- Email spam filtering 이메일 스팸 필터링

Three Types of Naive Bayes Model

- Gaussian - used in classification and it assumes that features follow a normal distribution 변수들이 정규분포를 따를때
- Multinomial - used for discrete counts 별개의 것을 셀때
- Bernoulli - useful if your feature vectors are binary (i.e. zeros and ones) 속성이 두가지 값만 가질때

Importing Library

```
#Import Library of Gaussian Naive Bayes model  
from sklearn.naive_bayes import GaussianNB, BernoulliNB,  
MultinomialNB
```

Building GaussianNB Model

```
#Create a Gaussian Classifier  
gnb = GaussianNB()  
  
# Train the model using the training sets  
gnb.fit(x, y)
```

Building BernoulliNB Model

```
#Create a BernoulliNB Classifier  
bnb = BernoulliNB()  
  
# Train the model using the training sets  
bnb.fit(x, y)
```

Building MultinomialNB Model

```
#Create a MultinomialNB Classifier  
mnb = MultinomialNB()  
  
# Train the model using the training sets  
mnb.fit(x, y)
```

Predicting Output

```
#Predict Output  
y_pred = gnb.predict(x_test)  
print(y_pred)
```

Evaluating Output

```
accuracy_score(y, gnb.predict(x)) #train performance  
accuracy_score(y_test, y_pred) #test performance  
confusion_matrix(y_test, y_pred) #test performance  
print(classification_report(y_test, y_pred))
```


Exercise #12

- Import the PlayTennis data set, do some basic exploration tasks, and use the graphs to visualize the data. 기본탐색, 시각화
 - `head()`, `describe()`, `value_counts()`
 - Histogram, boxplot, barplot, scatter plot

Exercise #12

- Encode the data to apply classification algorithms
분류알고리즘을 사용하기 위해 인코딩
 - "Outlook", "Temperature", "Humidity", "Wind", "Play Tennis"

Exercise #12

- Build a naïve Bayesian model 예측모델 구축 및 예측
- Check the r^2 , accuracy score, confusion matrix, and classification report of the model 성능평가

Exercise #12

- Read train.tsv **훈련데이터 읽기**
- Count phrase by sentiments and draw bar chart **감정에 따른 표현을 카운트, 막대그래프**
- Remove stopwords and punctuations **스탑워드와 문장기호 삭제**
- Use CountVectorizer and TF-IDF to generate bag of words **단어를 벡터화**
- Train and test split (.3 test set) **30프로의 테스트셋**
- Build a naïve Bayesian model (MultinomialNB) **나이브베이지안 모델 생성**
- Compare the results **결과 비교**

Read train.tsv 훈련 데이터 읽기

data - DataFrame

Index	Phraseld	Sentenceld	Phrase	Sentiment
0	1	1	A serie...	1
1	2	1	A serie...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2
5	6	1	of esca...	2
6	7	1	of	2
7	8	1	escapad...	2
8	9	1	escapad...	2
9	10	1	demonst...	2
10	11	1	demonst...	2

```
data=pd.read_csv('train.tsv',  
sep='\t')
```

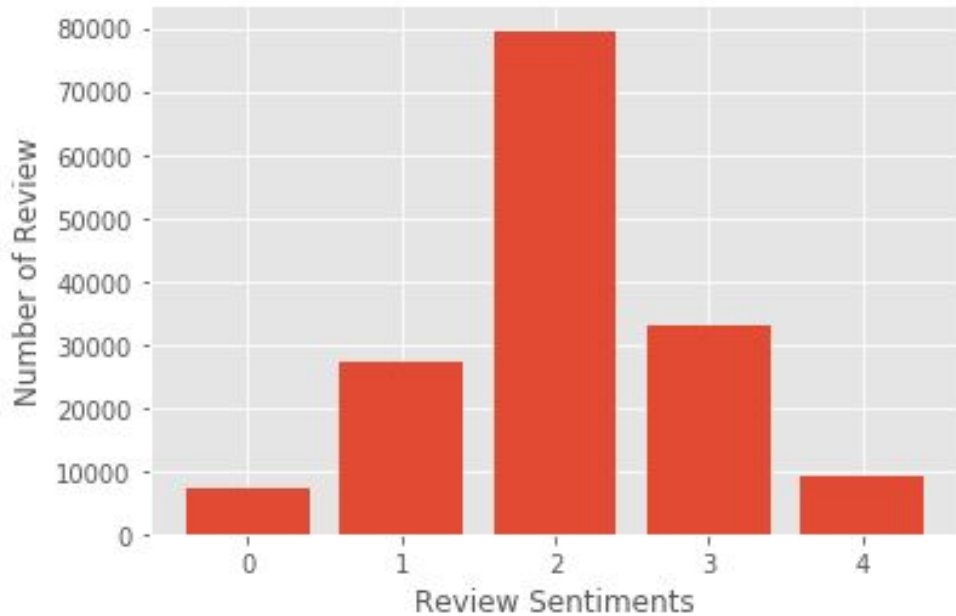
```
data.head()
```

```
data.info()
```

```
data.Sentiment.value_ counts(sort=False  
)
```

Count phrase by sentiments and draw bar chart

감정에 따른 표현을 카운트, 막대그래프



```
Sentiment_count=data.groupby  
('Sentiment').count()  
  
plt.style.use('ggplot')  
  
plt.bar(Sentiment_count.index.  
values,  
Sentiment_count['Phrase'])  
  
plt.xlabel('Review  
Sentiments')  
  
plt.ylabel('Number of  
Review')  
  
plt.show()
```

Remove stopwords and punctuations

스탑워드와 문장기호 삭제

```
import string

import re

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

ps = PorterStemmer()
```

```
def clean_text(text):

    text = "".join([word.lower() for word in text if word not in string.punctuation])

    tokens = re.split('\W+', text) #or word_tokenize(text)

    text = [ps.stem(word) for word in tokens if word not in set(stopwords.words('english'))]

    return text

clean_text(text)
```

하나의 텍스트를
처리할때 편함

Use CountVectorizer to generate bag of words 단어를 벡터화

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer

token = RegexpTokenizer(r'[a-zA-Z0-9]+')

cv =
CountVectorizer(lowercase=True, stop_words='english', ngram_range
e = (1,1), tokenizer = token.tokenize)

text_counts= cv.fit_transform(data['Phrase'])
text_counts = text_counts.toarray()
```

시리즈를
처리할때
편함

Train and test split (.3 test set)

30프로의 테스트셋

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(text_counts, data['Sentiment'],
test_size=0.3, random_state=1)
```

Build a naïve Bayesian model (MultinomialNB) 나이브베이지안 모델 생성

```
from sklearn.naive_bayes import MultinomialNB  
  
mnb = MultinomialNB().fit(X_train, y_train)  
  
y_pred = mnb.predict(X_test)  
  
print(y_pr)
```

Compare the results 결과 비교

```
from sklearn.metrics import accuracy_score,  
confusion_matrix, classification_report  
  
accuracy_score(y_test, y_pred) #test performance  
  
accuracy_score(y_train, mnbpredict(X_train)) #train  
performance  
  
confusion_matrix(y_test, y_pred) #test performance  
  
print(classification_report(y_test, y_pred)) #test  
performance
```

NN Model

```
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
y_test.value_counts()
model = Sequential()
model.add(Dense(10, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(5, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=2, validation_data=(X_test, y_test))
y_pred = model.predict_classes(X_test)
```

Evaluation of NN Results

```
model.evaluate(X_test, y_test)
accuracy_score(y_test, y_pred) #test performance
accuracy_score(y_train, mnbpredict(X_train)) #train
performance
confusion_matrix(y_test, y_pred) #test performance
print(classification_report(y_test, y_pred))
```

Exercise #12

Amazon Customer Review Example

- Import data (columns = ['sentence', 'label'])
- X and y split
- Vectorization (CountVectorizer □ data frame)
- Train and test split (sklearn.model_selection, test=.25, random_state))
- NN Model (10 Dense with relu, 1 Dense with sigmoid)
- Compile (adam, binary_crossentropy, accuracy)
- Fit (epochs=100, batch_size=10)
- Evaluate (accuracy and loss)

Keras Embedding Layers

- Keras offers an Embedding layer that can be used with text data.
캐라스는 텍스트데이터에 사용할수 있는 임베딩레이어를 제공함
- In an embedding, words are represented by dense vectors where a vector represents the projection of the word into a continuous vector space.
임베딩에서는 연속스페이스에서 단어를 표현하는 댄스벡터로 사용함.

Keras Embedding Layers

```
e = Embedding(200, 32,  
input_length=50)
```

- input_dim: the size of the vocabulary in the text data.
E.g., 11 words. 단어의 개수.
즉, 단어 집합(vocabulary)의 크기

- output_dim: the size of the vector space in which words will be embedded. the size of the output vectors from this layer for each word.
E.g., 32, 100, or larger. 임베딩한 후의 벡터의 크기
- input_length: the length of input sequences, as you would define for any input layer of a Keras model. 각 입력 시퀀스의 길이

Word Embedding Code

```
from keras.layers import
Embedding

#text=[['Hope', 'to', 'see',
'you', 'soon'], ['Nice',
'to', 'see', 'you',
'again']]

text=[[0, 1, 2, 3, 4], [5, 1,
2, 3, 6]]

Embedding(7, 2,
input_length=5)
```

```
#+-----+-----+
#|   index   | embedding |
#+-----+-----+
#|     0     | [1.2, 3.1] |
#|     1     | [0.1, 4.2] |
#|     2     | [1.0, 3.1] |
#|     3     | [0.3, 2.1] |
#|     4     | [2.2, 1.4] |
#|     5     | [0.7, 1.7] |
#|     6     | [4.1, 2.0] |
#+-----+-----+
```

Keras Tokenizer

텍스트문장으로
단어인 텍스트를 만들고
테스트단어를 추가함

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

```
t = Tokenizer()
```

```
text = 'The earth is an awesome place to live'
```

```
t.fit_on_texts([text])
```

```
print('word index: ', t.word_index)
```

word index: {'the': 1, 'earth': 2,
'is': 3, 'an': 4, 'awesome': 5, 'place':
6, 'to': 7, 'live': 8}

```
t.texts_to_sequences([text])
```

[[1, 2, 3, 4, 5, 6, 7, 8]]

```
test = 'The earth is an great place to live'
```

```
sequences = t.texts_to_sequences([test])
```

```
print('sequences: ', sequences)
```

[[1, 2, 3, 4, 6, 7, 8]]

Padding Sequences

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

```
pad_sequences([[1, 2, 3], [3, 4, 5, 6], [7, 8]],  
maxlen=3, padding='pre')
```

```
Embedding(9, 2, input_length=3)
```

```
array([[1, 2, 3],  
       [4, 5, 6],  
       [0, 7, 8]])
```

Exercise #12

- X and y split (x – Phase, y-Sentiment, use only 5000 data!)
- Data preprocessing
 - Encoding (from `keras.preprocessing.text` import `one_hot`, use 5000 words)
 - Padding (from `keras.preprocessing.sequence` import `pad_sequences`, 400 post padding)
- Model building (Embedding 50 output vectors, 1 Flatten, 5 Dense with softmax)
- Compile (optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
- Evaluation (epochs=50)

X and y split 데이터 나누기

```
import pandas as pd  
data=pd.read_csv('train.tsv/train.tsv', sep='\t')  
docs = data['Phrase'][:5000]  
labels = data['Sentiment'][:5000]
```

Encoding

```
vocab_size = 5000

from
keras.preprocessing.text
import one_hot

encoded_docs =
[one_hot(d, vocab_size)
for d in docs]

print(encoded_docs)
```

■ 5000개의 단어로 전체
컬럼을 숫자화 시킴

Padding

```
max_length = 400

from
keras.preprocessing.sequen
ce import pad_sequences

padded_docs =
pad_sequences(encoded_docs
, maxlen=max_length,
padding='post')

print(padded_docs)
```

- 400개의 넓이로 숫자르 패딩함
- 결과값은 5000x400

CNN Model Building

```
from keras.models import Sequential
from keras.layers import Dense, Flatten, Embedding
model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=max_length)) #5000개 단어공간,
50개의 결과벡터, 400개의 변수
model.add(Flatten())
model.add(Dense(5, activation='softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
print(model.summary())
history = model.fit(padded_docs, labels, epochs=50, verbose=1)
```


Evaluation

```
score = model.evaluate(padded_docs, labels, verbose=1)
print('Accuracy: %f' % (score[1]*100))
```

Word Embedding Example

```
docs = ['Well done!',  
        'Good work',  
        'Great effort',  
        'nice work',  
        'Excellent',  
        'Weak',  
        'Poor effort!',  
        'not good',  
        'poor work',  
        'Could have done better.']  
labels = np.array([1,1,1,1,1,0,0,0,0,0])
```

- One hot encoding (50 vocab size)
- Padding (4 maxlen, post)
- CNN model
 - Embedding (8 output vector size)
 - Flatten, 1 Dense with sigmoid
- Compile
- Fit (epochs = 50)
- Evaluate (accuracy)

Exercise #12

스팸분류 문제

- Download the SMS Spam Collection Data Set from UCI ML repository
스팸데이터셋을 다운로드
- Read and explore the dataset 데이터 읽고 탐색

	label	body_text
0	ham	I've been searching for the right words to tha...
1	spam	Free entry in 2 a wkly comp to win FA Cup fina...
2	ham	Nah I don't think he goes to usf, he lives aro...
3	ham	Even my brother is not like to speak with me. ...
4	ham	I HAVE A DATE ON SUNDAY WITH WILL!!

Exercise #12

Preprocess data – punctuation, tokenize, stopwords, stemming, lemmatizer
전처리

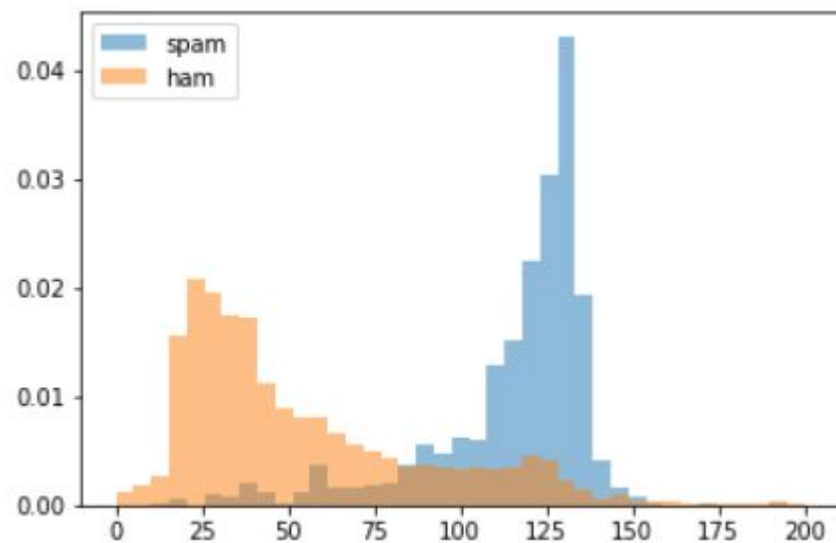
- Vectorize data – bag of words, bi-grams, TF-IDF 단어의 벡터화
- Add new columns – length of text message, % of punctuation in the text
새로운 컬럼 생성
- Draw the histogram of new columns respectively by labels 히스토그램

Exercise #12

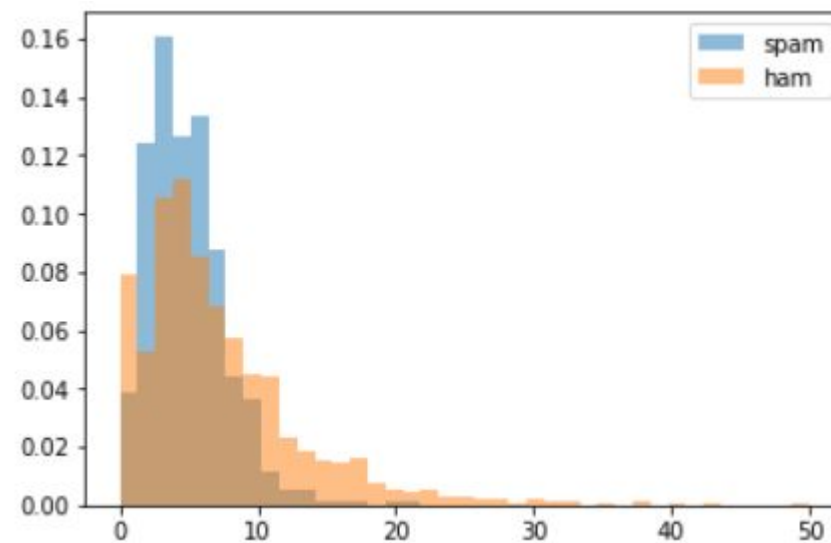
스팸분류 문제

- To decide if the text is spam or not, build:
 - a Random Forest Model
 - a Logistic Regression Model
 - Bayesian Model
 - A CNN Model
- Compare the results

Visualizing Features of Text



message body length



% of punctuation

Random Forest Model

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

rf = RandomForestClassifier()

param = {'n_estimators': [10, 150, 300],
         'max_depth': [30, 60, 90, None]}

gs = GridSearchCV(rf, param, cv=5, n_jobs=-1) # n_jobs=-1 for parallelizing search
gs_fit = gs.fit(X_count_feat, data['label'])
pd.DataFrame(gs_fit.cv_results_).sort_values('mean_test_score', ascending=False).head()
```

Logistic Model

```
#LOGISTIC REGRESSION  
from sklearn.linear_model import LogisticRegression  
classifier = LogisticRegression()  
classifier.fit(x_train, y_train)  
score = classifier.score(x_test, y_test)  
print("Accuracy:", score)
```


Exercise #12

- Model building with imdb

```
from keras.datasets import imdb
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=5000)
```

- Padding x_train, x_test (maxlen=400)

- CNN Model

- Embedding (embedding_dims = 50), Dropout(0.2), Flatten()
- Conv1D (250 filters, 3 kernel size, activation='relu'), MaxPooling1D()
- 250 Dense with activation='relu', Dropout(0.2)
- 1 Dense with sigmoid

- Compile and fit (epochs=2, batch=32)

- Predict and evaluation

Exercise #12

- Modeling building with yelp
 - X and y split
 - Encoding with Tokenizer (num_words=5000, vocab size = len(tokenizer.word_index) + 1) and padding (maxlen = 100, post)
 - Train and test split
 - Embedding (output_dim=50), Flatten, 10 Dense with relu, 1 Dense with sigmoid
 - Compile and fit (epochs = 10)
 - Evaluate (loss and accuracy)