# Text/Web Application

**Dr. Saerom Park**
**Statistical Learning and Computational Finance Lab.**
**Department of Industrial Engineering**
*psr6275@snu.ac.kr*
http://slcf.snu.ac.kr

# Reference

- **Reading:** [Raschka. (2017), chapter 8 & 9], [Müller. (2016), chapter 7].

# TEXT APPLICATION

# Text Application

- 텍스트 데이터의 머신러닝
  - 현대 기술의 발달로 인해 데이터를 모으고 분석하는 것이 편해짐.
  - Natural Language Processing(NLP) & Sentiment analysis
  - IMDb(Internet Movie Database)를 바탕으로 다음을 진행한다.
    - 데이터 준비 및 가공 (Cleaning and preparing text data)
    - 문서의 벡터화 (Building feature vectors from text documents)
    - 분류 모델 학습 (Training a machine learning claasification model)
    - 문서 분류 (Inferring topics from document collections for categorization)

# IMDb data

- IMDb movie review data
  - http://ai.Stanford.edu/~amaas/data/sentiment/
  - 리뷰(Review) + 감성(Sentiment)
  - 리뷰는 특정 영화에 대한 리뷰 전체
  - 감성은 긍정이면 1, 부정이면 0

| | review | sentiment |
|---|---|---|
| 2 | In 1974, the teenager Martha Moxley (Maggie Grace) moves to the high-class | 1 |
| 3 | OK... so... I really like Kris Kristofferson and his usual easy going delivery of line | 0 |
| 4 | ***SPOILER*** Do not read this, if you think about watching that movie, althou | 0 |
| 5 | hi for all the people who have seen this wonderful movie im sure thet you wo | 1 |
| 6 | I recently bought the DVD, forgetting just how much I hated the movie versio | 0 |
| 7 | Leave it to Braik to put on a good show. Finally he and Zorak are living their | 1 |
| 8 | Nathan Detroit (Frank Sinatra) is the manager of the New York's longest- esta | 1 |
| 9 | To understand "Crash Course" in the right context, you must understand the | 1 |
| 10 | I've been impressed with Chavez's stance against globalisation for sometime n | 1 |
| 11 | This movie is directed by Renny Harlin the finnish miracle. Stallone is Gabe Wa | 1 |
| 12 | I once lived in the u.p and let me tell you what. I didn't have the foggyest ide | 0 |
| 13 | Hidden Frontier is notable for being the longest running internet-based Star T | 1 |
| 14 | It's a while ago, that I have seen Sleuth (1972) with two great actors Michael | 0 |
| 15 | What is it about the French? First, they (apparently) like Jerry Lewis a lot more | 0 |
| 16 | This very strange movie is unlike anything made in the west at the time. With | 1 |
| 17 | I saw this movie on the strength of the single positive review and I can only ir | 0 |
| 18 | There are some great philosophical questions. What is the purpose of life? Wh | 0 |
| 19 | I was cast as the Surfer Dude in the beach scenes. Almost got cast as the mus | 1 |
| 20 | I had high hopes for this one until they changed the name to 'The Shepherd : | 0 |
| 21 | Set in and near a poor working class town in the mountains of rural Italy, it's | 1 |
| 22 | Opulent sets and sumptuous costumes well photographed by Theodor Sparku | 0 |
| 23 | i saw the film and i got screwed, because the film was foolish and boring. i th | 0 |
| 24 | I'm getting a little tired of people misusing God's name to perpetuate their ov | 0 |
| 25 | How offensive! Those who liked this movie have probably never opened a bibl | 0 |
| 26 | What else can you say about this movie,except that it's plain awful.Tina Louise | 0 |
| 27 | Certain aspects of Punishment Park are less than perfect, specifically some of t | 1 |
| 28 | First of all, I'd like to tell you that I'm into comics, anime, animation and such | 0 |
| 29 | You should not take what I am about to say lightly. I've seen many, many film | 1 |
| 30 | I love the Jurassic Park movies, they are three of my all time favorite movies.< | 0 |

movie_data

# Cleaning and Preparing text data

## 특수문자 처리

- 주어진 IMDb data를 그대로 다루게 될 경우, 쉼표(,)나 마침표(.) 등의 특수기호는 물론이고 :)와 같은 이모티콘 역시 모델 학습에 포함된다.
- 특수기호가 유의미한 경우도 있겠지만, 대부분의 경우 이는 모델을 학습하는 데에 Noise가 될 수 있으므로 보통 제거하여 사용한다.
- Python에서는 특수기호를 포함한 문자를 다루기 위해 정규표현식(regular expression, regex)이라는 함수를 제공하며, 이를 사용하면 특수기호나 특정 문자를 제거할 수 있다.

- 파이썬 2.7 : https://docs.python.org/2/library/re.html

- 파이썬 3.6 : https://docs.python.org/3.6/library/re.html

# Cleaning and Preparing text data

## Documents into tokens

- 문서를 분석하기 위해서는 최소 분석 단위(token)을 정해야 한다. Token은 음절, 단어 등 여러 단위가 될 수 있으며, 그 중에서도 많이 쓰이는 것은 단어 단위의 token이다.
- 문서를 token으로 나누는 과정을 tokenize라고 하는데, 단어를 token으로 설정할 경우 띄어쓰기나 특수문자 등을 기준으로 tokenize한다.

> “Runners like running and thus they run”
> → [‘runners’, ‘like’, ‘running’, ‘and’, ‘thus’, ‘they’, ‘run’]

> “Hello, nice to meet you.”
> → [‘hello’, ‘nice’, ‘to’, ‘meet’, ‘you’]

# Cleaning and Preparing text data

## Stemming & Stop-words

- 보다 정확한 분석을 위해 tokenize된 token들을 원형(ex. Running → Run)으로 만들어주는 stemming이란 기술도 많이 활용된다. Stemmer의 방법 및 어휘에 따라 Krovetz Stemmer, Porter Stemmer 등 많은 Stemmer가 존재다.
- 또 다른 문서 분석을 위한 도구 중에는 Stop-words라는 기술이 있는데, 이는 불필요한 단어를 사전 제거하여 분석을 깔끔하게 하는 기술이다.
- 문서에 따라 다르게 설정하는 것이 보통이나, 일반적으로 영문 문서에서 'the', 'is', 'a' 등은 많이 사용되지만 그 의미가 크지 않으므로 Stop-words로 설정하여 미리 제거해주는 것이 좋다.
- Python의 Natural Language Toolkit(NLTK)에서는 Porter Stemmer와 함께 Stop-words도 제공하여 보다 정확한 문서 분석 도구를 제공한다.
- http://www.nltk.org/

# Processing text data

- Bag-of-words model
  - Text to numerical feature vectors
  - 단어와 같은 token을 바탕으로 전체 문서(document)에 대한 단어장(vocabulary)을 만든 후, 각 문서마다 특정 token을 가지고 있는 개수로 벡터화한다.


https://machinelearnings.co/text-classification-using-neural-networks-f5cd7b8765c6

# Processing text data

- TF-IDF(Term Frequency-inverse document frequency)
  - 단어의 개수로만 벡터화할 경우, 특정 분류의 문서에만 등장하는 단어를 잡아낼 수 없는 단점이 발생한다.
  - 따라서 특정 단어가 얼마나 많이 등장하느냐 뿐만 아니라, 얼마나 많은 문서에 등장하는가 살펴볼 필요가 있다.
  - TF(Term Frequency) : 단어의 등장 횟수
  - DF(Document Frequency) : 단어가 등장하는 문서의 개수
  - IDF(Inverse Document Frequency) : $\log\frac{1+n_d}{1+DF}$ ( $n_d$: 전체 문서의 개수 )
  - TF-IDF(Term Frequency-inverse document frequency) : TF×IDF

# Training Model

## Vectorizers

- sklearn.feature_extraction.text.CountVectorizer(input='content', encoding='utf-8', ..., lowercase=True, stop_words=None, ...) / TfidfVectorizer

| Parameters | |
|---|---|
| Input | Input 데이터 |
| encoding | Input 데이터의 encoding 형태 |
| lowercase | 대문자를 소문자로 모두 바꿀 것인가 여부 |
| stop_words | 분석에서 제거할 단어의 집합 |

| Attributes | |
|---|---|
| vocabulary_ | 각 단어를 변환한 결과를 딕셔너리로 반환 |
| stop_words_ | 부석에서 제거된 단어의 집합 |

- CounterVectorizer : http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
- TfidfVecotrizer : http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

# Training Model

## Vectorizers

- Scikit learn에서 제공하는 Vectorizer는 Bag-of-words에 해당하는 CounterVectorizer와 TF-IDF를 도입한 TfidfVectorizer이 대표적이다. 이들을 이용하면 주어진 IMDb 문서를 아래와 같이 벡터화시킬 수 있다.

```
count = CountVectorizer()
docs = np.array([
        'The sun is shining',
        'The weather is sweet',
        'The sun is shining, the weather is sweet, and one and one is two'])
bag = count.fit_transform(docs)
```

# Practice

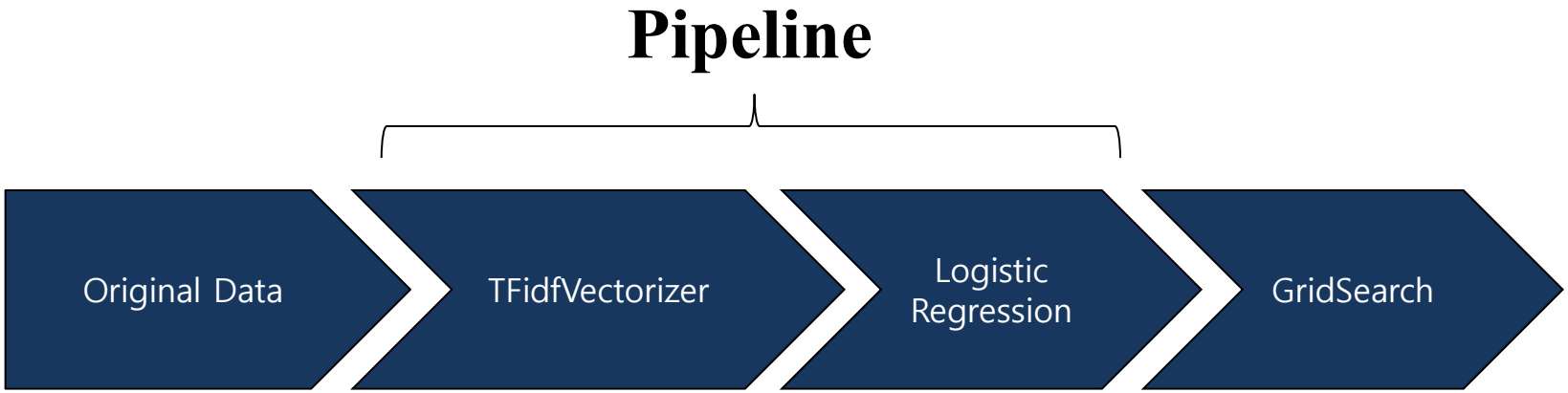
- Code for Cleaning and Preparing & Processing text data
  - Preparing the IMDb movie review data for text processing
  - Transforming documents into feature vectors
  - Assessing word relevancy via term frequency-inverse document frequency
  - Cleaning text data
  - Processing documents into tokens
  - Code 01 참고

# Training Model

- Training Model with logistic regression model for document classification
  - 벡터화된 문서를 바탕으로 Logistic regression을 통해 문서를 분류할 수 있다.
  - 이 때 Pipeline을 사용하게 되는데 Pipeline이란 입력된 리스트에 정의된 함수를 순차적으로 적용하는 함수이다. 인자는 ('이름', 함수)인 튜플로 저장되어야 한다.

```
lr_tfidf = Pipeline([('vect', tfidf), ('clf', LogisticRegression(random_state=0))])
```

- http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html



# Training Model

## Scikit-learn의 Pipeline

- sklearn.pipeline.Pipeline(steps, …)

| Parameters | |
|---|---|
| steps | List형 튜플으로 입력되는 함수의 집합 |

| Methods | |
|---|---|
| fit(X[,y]) | 모델을 학습시킴 |
| set_params(**kwargs) | Pipeline 안의 함수들의 변수를 세팅 |
| predict(X) | 학습시킨 모델로 주어진 X에 대해 예측치를 반환 |
| score(X[, y, sample_weight]) | 학습된 모델로 변환된 결과와 실제 값을 비교한 score를 반환 |

```
anova_filter = SelectKBest(f_regression, k=5)
clf = svm.SVC(kernel='linear')
anova_svm = Pipeline([('anova', anova_filter), ('svc', clf)])
anova_svm.set_params(anova__k=10, svc__C=.1).fit(X, y)
prediction = anova_svm.predict(X)
anova_svm.score(X, y)
```

# Training Model

- Training Model with logistic regression model for document classification(Cont')
  - 그 후 Scikit learn에서 제공하는 Logistic regression에 GridSearchCV을 도입하면 주어진 IMDb를 바탕으로 모델을 학습시킬 수 있다.

```
gs_lr_tfidf = GridSearchCV(lr_tfidf, param_grid,
                           scoring='accuracy',
                           cv=5, #cross validation
                           n_jobs=-1) #병행 여부
gs_lr_tfidf.fit(X_train, y_train)
```

  - http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

# Training Model

## Scikit-learn의 GridSearchCV

- sklearn.model_selection.GridSearchCV(estimator, param_grid, scoring=None, ...)

| Parameters | |
|---|---|
| estimator | GridSearchCV를 실행할 분류 모델 |
| param_grid | GridSearchCV를 진행할 parameter의 딕셔너리 셋 |
| scoring | 최적의 parameter를 선택하는 기준 |
| verbose | 실행 중의 모든 결과를 보여줄 것인가에 대한 여부 |

| Attributes | |
|---|---|
| best_estimator_ | 가장 성능이 좋은 parameter를 반환 |
| best_score_ | 가장 좋은 score를 반환 |

| Methods | |
|---|---|
| fit(X[,y,groups]) | 주어진 X(와 y)로 모델을 학습시킴 |
| predict(X) | X를 활용해 가장 좋은 estimator로 예측한 값을 반환 |

- http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

# Training Model

- Working with bigger data – online algorithms and out-of-core learning
  - 위의 GridSearchCV를 실행 시 50000개만의 리뷰를 사용했음에도 시간이 굉장히 오래 걸린다. (1시간 이상)
  - 이는 Stochastic gradient descent와 partial_fit, HashinVectorizer 함수를 활용하면 저장소 내의 문서를 mini-batch화 하여 logistic regression을 바로 학습시켜 시간을 단축시킬 수 있다.
  - CounterVectorizer의 경우 모든 작업을 메모리 상에서 실행하는 반면, HashingVectorizer는 해시 함수를 사용하여 단어에 인덱스 번호를 부여하기 때문에 메모리 및 실행시간을 줄일 수 있다.

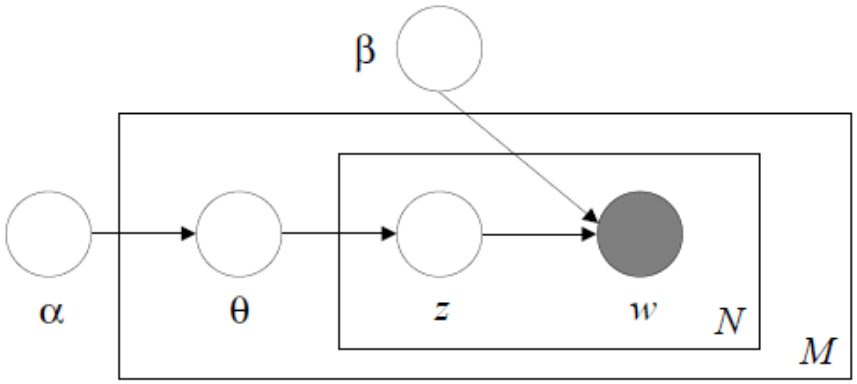
```
vect = HashingVectorizer(decode_error='ignore', n_features=2**21,
                         tokenizer=tokenizer)
X_train = vect.transform(X_train)
```

  - http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.HashingVectorizer.html

# Topic Modeling

## Topic Modeling

- Latent Dirichlet Allocation을 활용하여 특정 문서가 어떤 카테고리에 속하는 지 유추 가능하다.
- LDA(Latent Dirichlet Allocation) : 이미 알고 있는 주제별 단어 수 분포를 바탕으로 새로운 문서의 단어 수 분포를 통해 해당 문서의 주제를 예측하는 알고리즘. (단, 주제들은 Dirichlet분포를 따른다고 가정)




- 허나 이 과정을 통해 Topic을 컴퓨터가 스스로 지정해주는 것은 아니다. 비슷한 카테고리에 속할 것으로 예측되는 문서를 묶을 뿐, 그 카테고리의 이름은 사람이 지정해주어야 한다.

# Practice

## Code for Training Model

- Training a logistic regression model for document classification
- Working with bigger data - online algorithms and out-of-core learning
- Topic Modeling : Latent Dirichlet Allocation with scikit-learn
- Code 02 참고

# Web Application

## Serializing fitted scikit-learn estimators

- 학습시킨 classifier을 온라인에서 사용하기 위해서는 따로 저장할 필요가 있다.
- 이를 위해 python에서는 pickle이라는 패키지를 통해, classifier는 물론 여러 데이터를 pkl이라는 확장자를 가진 파일로 내보내는 기능을 제공한다.
- 저장할 때에는 pickle의 dump라는 함수를 사용하며, 읽어올 때는 pickle의 load라는 함수를 사용한다.

```
pickle.dump(clf, open(os.path.join(dest, 'classifier.pkl'), 'wb'), protocol=4)
pickle.load(open(os.path.join('pkl_objects', 'classifier.pkl'), 'rb'))
```

- Load된 classifier는 바로 사용 가능하다.

```
clf = pickle.load(open(os.path.join('pkl_objects', 'classifier.pkl'), 'rb'))
clf.predict(X)
```

# Reference


- [Raschka. (2017)] Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning*. Packt Publishing Ltd, 2017.
- [Müller. (2016)] Müller, Andreas C., and Sarah Guido. *Introduction to machine learning with Python: a guide for data scientists*. 2016.
- [GÉRON. (2017)] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* Sebastopol, CA: O, 2017.