

# **Text Mining**

**Bok, Jong Soon**

**[javaexpert@nate.com](mailto:javaexpert@nate.com)**

**<https://github.com/swacademy>**

# Text Mining



# Text Mining개요

- Text Data 접근(Text-as-Data Approach)
  - Digital화된 Text를 수치형 Data로 간주하여 Algorithm을 적용해서 분석/요약하는 통계기법
- 위계적 질서를 갖는 Text를 Row와 Column의 Matrix 형태를 갖는 Data로 개념화.
  - 문자(Letter) → 형태소(Morpheme) → 단어(Word) → 문장 (Sentence) → 단락(Paragraph) → 문서(Document) → 말뭉치 (Corpus)
- Text Data를 이해하고 모형화하는 작업
- Token을 근거로 문서에 나타난 주제(Topic)이나 감정 (Sentiment)을 추정.

# Text Mining개요(Cont.)

## ● Data Mining

- 대규모 Big Data(정형화된 Data)에서 가치 있는/의미 있는 정보를 추출하는 기술
- 사람이 예측할 수 없는 의미 있는 경향과 규칙까지 발견하기 위해서 대량의 Big Data로부터 자동화 또는 반자동화 도구를 활용해 탐색하고 분석하는 과정
- 고급 통계 분석과 Modeling 기법을 적용하여 Data 안의 Pattern과 관계를 찾아내는 과정.

# Text Mining개요(Cont.)

## ● Text Mining

- 대규모의 Text 문서에서 의미 있는 추출하는 기술
- 비정형 Data(글자/텍스트)를 정형화 및 특징을 추출하는 과정이 필요.
- Text 덩어리 안에서 단어들을 분해해 단어의 출현빈도나 단어들 간의 관계성을 파악하여 의미 있는 정보를 추출해내는 기술
- 정보 검색, Data Mining, Machine Learning, 통계학, Computer 언어학(Computational linguistics) 및 Computer 공학 등이 결합된 분야
- 특히 분석 대상이 형태가 일정하지 않고 다루기 힘든 비정형 데이터이므로 인간의 언어를 Computer가 인식해 처리하는 자연어 처리(NLP) 방법과 관련이 깊음.

# Text Mining개요(Cont.)

- 전통적 내용분석 방식이 유지되기 어려운 이유
  - Social Media, Online 공간에서 기존과는 다른 새로운 Text가 넘쳐나고 있다.
  - 내용분석을 실시할 coder들을 고용하고 교육시키기 위한 비용과 시간이 더욱 많이 필요.
  - Coder들은 인간이기 때문에 검사-재검사 신뢰도(Test-Retest Reliability) 낮다.
  - 방대한 Text를 분석하는 데 엄청난 시간과 비용이 소요.

# Text Mining 개요(Cont.)

- “It is good” vs “It is bad”

|              | bad | good | is | it | Sentiment |
|--------------|-----|------|----|----|-----------|
| “It is good” | 0   | 1    | 1  | 1  | ?         |
| “It is bad”  | 1   | 0    | 1  | 1  | ?         |

- Text-as-Data

- Data Matrix
- Numerical Array : 분석 단위 X 단어
- Document-Term Matrix(DTM) : 문서 X 단어 행렬

# Text Mining개요(Cont.)

- Text Data / DTM(문서 X 단어 행렬, Document-Term Matrix)를 분석하는 방법
  - Dictionary-based Approach
    - 사전(事前, A Priori)에 규정된 단어의 의미를 기반으로 Text의 의미 추정.
    - Computer Algorithm을 이용한 Text Mining
    - 감정분석(Sentiment Analysis)
  - Machine Learning을 이용한 기법
    - 단어의 의미는 기계학습 사후(事後, A Post Hoc)에 추정.

# Text Mining개요(Cont.)

## ● 지도 기계학습(Supervised Machine Learning)

- 기계학습을 위해 사용되는 Text Data 중 일부에서 Text의 의미가 알려져 있는 경우
- 훈련 Data는 예측변수와 결과변수로 구성
- 예측변수와 단어들의 속성(예:빈도 ; Frequency 등)과 이 예측변수를 통해 예측되는 결과변수인 Text Data의 분석 단위에 나타난 의미(예:Topic이나 Sentiment 등)으로 구성.
- 예측변수와 결과변수의 관계를 최적으로 설명할 수 있는 함수 추출.
- 문서에 표출된 감정이 어떤지에 대한 인간의 판단과 문서에 등장하는 단어들의 관계를 지도학습을 통해 추정한 후, 새로운 문서에 지도학습으로 추정된 문서분류모형을 적용
- Logistic Regression, Naïve Bayes Classification, SVM, Boosting, Deep Learning

# Text Mining개요(Cont.)

## ● 비지도 기계학습(Unsupervised Machine Learning)

- Text Data의 의미가 전혀 알려져 있지 않고 기계학습을 통해 Text Data의 의미를 추정하는 경우.
- PCA, Cluster Analysis
- 연구자가 분석 단계마다 자신의 주관적 판단의 개입이 필요.

# 자연어 처리와 Text Mining

## ● 자연어 처리 학습 주제

- 텍스트 전처리 : 토큰화, 정제, 형태소 분석, 불용어 처리. Label Encoding
- 개수 기반 단어 표현 : 문장 내에서 단어들의 빈도수를 측정해서 이를 기반으로 데이터를 분석할 수 있는 형태로 만드는 것
- 문서 유사도(Document Similarity) : 단어들을 수치화 한 후 이를 기반으로 단어들 사이의 거리를 계산해서 문서 간의 단어들의 차이를 계산하는 것
- Topic Modeling : 텍스트 본문의 숨겨진 의미 구조를 발견하기 위해 사용되는 Text Mining 기법
- 연관 분석(Association) : 문서 내의 단어들을 이용해서 연관 분석을 실시

# 자연어 처리와 Text Mining (Cont.)

## ● 자연어 처리 학습 주제

- Deep Learning을 이용한 자연어 처리 : RNN, LSTM 등의 인공신경망 Algorithm을 이용해서 Deep Learning으로 자연어 처리
- Word Embedding : Word2vec 패키지를 이용해서 단어를 벡터로 표현하는 방법으로 희소 표현에서 밀집 표현으로 변환하는 것
- 텍스트 분류(Text Classification) : 텍스트를 입력으로 받아, 텍스트가 어떤 종류의 범주(Class)에 속하는지를 구분하는 작업
- Tagging : 각 단어가 어떤 유형에 속해 있는지를 알아내는 것
- 번역(Translation) : 챗봇(Chatbot) 또는 기계 번역(Machine Translation)

# 자연어(Natural Language Processing, NLP)

- 사람이 사용하는 언어를 Computer로 처리하는 각종 방법
- 크게 2가지로 나눈다
  - 통계적 자연어 처리
  - 언어학 자연어 처리

# NLTK 자연어 처리 Package

# NLTK(Natural Language Toolkit) Package

- 교육용으로 개발된 자연어 처리와 문서 분석용 Python Package
- <https://www.nltk.org>
- NLTK 패키지가 제공하는 주요 기능
  - 말뭉치(corpus)
  - 토큰 생성(tokenizing)
  - 형태소 분석(morphological analysis)
  - 품사 태깅(POS tagging)
- Cross-Platform Open Source

# NLTK(Natural Language Toolkit) Package (Cont.)

| 언어 처리 분야       | NLTK Module            | 기능   |
|----------------|------------------------|--|
| Corpora 접근     | corpus                 | corpora와 lexicon에 대한 표준화된 interface                          |
| 문자열 처리         | tokenize, stem         | Tokenizing, 어간 추출(Stemming)                                  |
| Collocation 발견 | collocations           | t-test, chi-squared, point-wise                              |
| 품사 Tagging     | tag                    | n-gram, backoff, Brill, HMM, TnT                             |
| 기계 학습          | classify, cluster, tbl | Decision Making Tree, Naïve Bayes, k-means 등                 |
| Chunking       | chunk                  | 정규식, n-gram, named-entity                                    |
| Parsing        | parse, ccg             | Chart, 특징 기반, 단일화, 확률론적 의존성                                  |
| 구문 해석          | sem, inference         | Lambda 미적분학, 1차 논리, 모델 검사                                    |
| 평가 지표          | metrics                | 정밀도(precision), 재현율(recall), 계수 일치도 (Agreement coefficients) |
| 확률 및 추정        | probability            | 빈도 분포, 확률 분포   |
| 응용프로그램         | app, chat              | 용어 색인기(Concordancer), Parser, WorldNet Browser, Chatbot      |
| 언어 현장 조사       | toolbox                | SIL, Toolbox 형식으로 데이터 조작                                     |

# 말뭉치(Corpus)

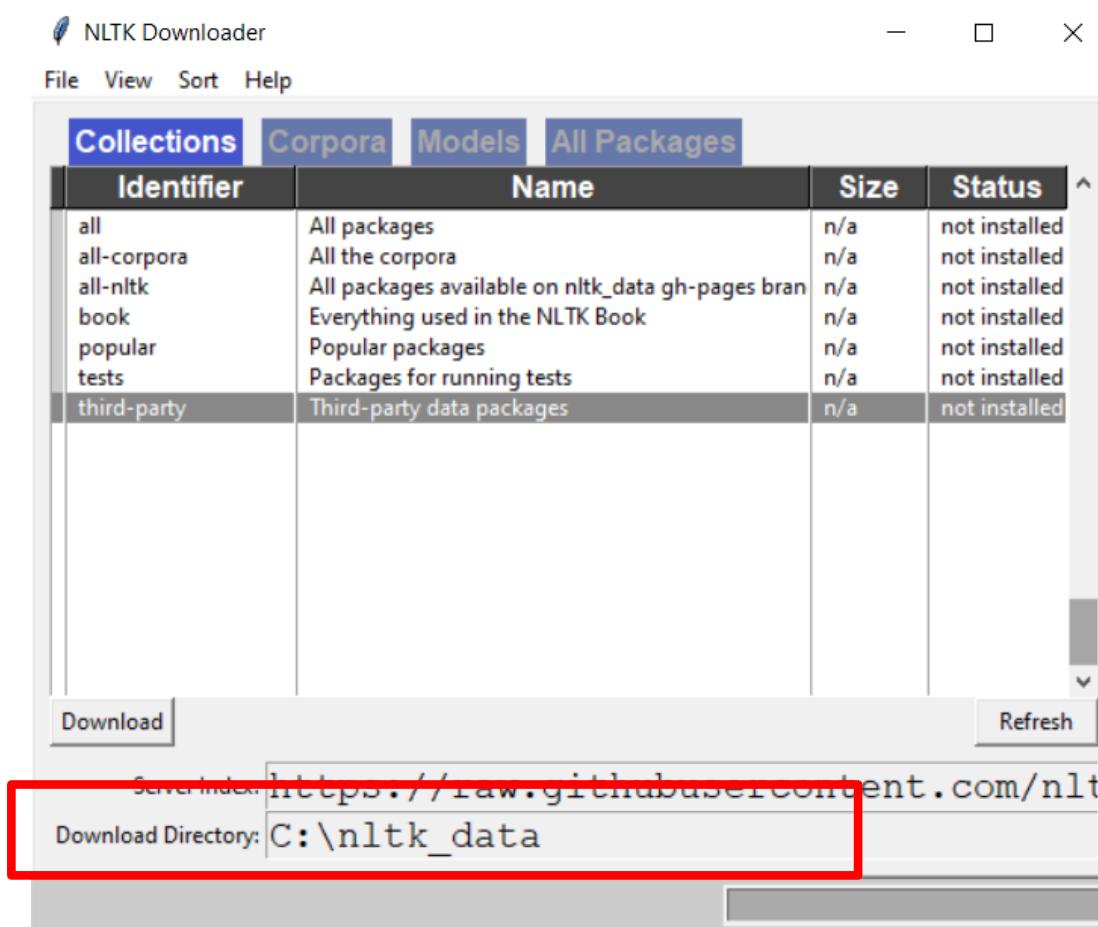
- 연구 대상 분야의 언어 현실을 총체적으로 보여 주는 자료의 집합
- Computer를 이용해 자연어 분석 작업을 할 수 있도록 만든 문서 집합
- NLTK에서는
  - 단순히 소설, 신문 등의 문서
  - 품사, 형태소 등의 보조적 의미를 추가
  - 쉬운 분석을 위해 구조적인 형태로 정리
- NLTK에서 corpus module로 학습용 말뭉치 제공
- **nltk.download()** 함수로 download 해야 함.

# 말뭉치(Corpus) (Cont.)

```
import nltk
```

```
nltk.download()
```

showing info [https://raw.githubusercontent.com/nltk/nltk\\_data/gh-pages/index.xml](https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)



# Penn Treebank Data

- NLTK를 설명할 목적으로 사용
- Tokenizing, Tagging, Chunking, Parsing 등에 사용
- Treebank
  - 구문 주석 말뭉치
  - 구조 분석을 좀 더 정교하게 만든 Tree Structure의 집합
- 400만 어절 규모의 Treebank 중 가장 유명한 것
- 주로 WallStreet Journal의 문장들로 구성.
- **nltk.download('treebank')**

```
[nltk_data] Downloading package treebank to C:\nltk_data...
[nltk_data] Package treebank is already up-to-date!
```

# Penn Treebank Data (Cont.)

NLTK Downloader

File View Sort Help

Collections Corpora Models All Packages

| Identifier            | Name   | Size     | Status        |
|-----------------------|--|----------|---------------|
| shakespeare           | Shakespeare XML Corpus Sample                    | 464.3 KB | not installed |
| sinica_treebank       | Sinica Treebank Corpus Sample                    | 878.2 KB | not installed |
| smultron              | SMULTRON Corpus Sample                           | 162.3 KB | not installed |
| state_union           | C-Span State of the Union Address Corpus         | 789.8 KB | not installed |
| stopwords             | Stopwords Corpus                                 | 22.6 KB  | not installed |
| subjectivity          | Subjectivity Dataset v1.0                        | 509.4 KB | not installed |
| swadesh               | Swadesh Wordlists                                | 22.3 KB  | not installed |
| switchboard           | Switchboard Corpus Sample                        | 772.6 KB | not installed |
| timit                 | TIMIT Corpus Sample                              | 21.2 MB  | not installed |
| toolbox               | Toolbox Sample Files                             | 244.7 KB | not installed |
| treebank              | Penn Treebank Sample                             | 1.7 MB   | installed     |
| twitter_samples       | Twitter Samples                                  | 15.3 MB  | not installed |
| udhr                  | Universal Declaration of Human Rights Corpus     | 1.1 MB   | not installed |
| udhr2                 | Universal Declaration of Human Rights Corpus (Ur | 1.6 MB   | not installed |
| unicode_samples       | Unicode Samples                                  | 1.2 KB   | not installed |
| universal_treebanks_v | Universal Treebanks Version 2.0                  | 24.7 MB  | not installed |

Download Refresh

Server Index: [https://raw.githubusercontent.com/nltk/nltk\\_data/3.2.5/corpora/treebank.zip](https://raw.githubusercontent.com/nltk/nltk_data/3.2.5/corpora/treebank.zip)

Download Directory: C:\nltk\_data

Finished installing treebank

# Penn Treebank Data (Cont.)

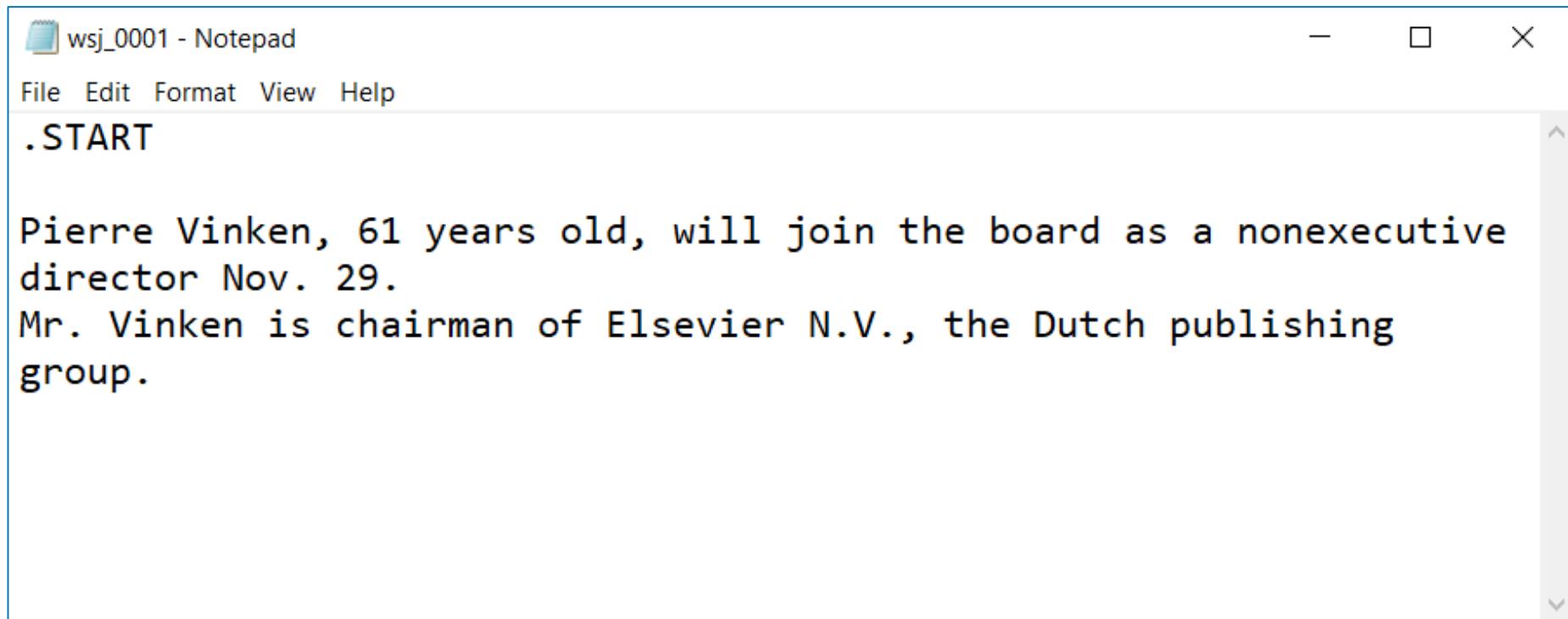
## ■ File들의 id 확인

```
from nltk.corpus import treebank  
print(treebank.fields())
```

```
['wsj_0001.mrg', 'wsj_0002.mrg', 'wsj_0003.mrg', 'wsj_0004.mrg', 'wsj_0005.mrg', 'wsj_0006.mrg', 'wsj_0007.mrg', 'wsj_0008.mrg', 'wsj_0009.mrg', 'wsj_0010.mrg', 'wsj_0011.mrg', 'wsj_0012.mrg', 'wsj_0013.mrg', 'wsj_0014.mrg', 'wsj_0015.mrg', 'wsj_0016.mrg', 'wsj_0017.mrg', 'wsj_0018.mrg', 'wsj_0019.mrg', 'wsj_0020.mrg', 'wsj_0021.mrg', 'wsj_0022.mrg', 'wsj_0023.mrg', 'wsj_0024.mrg', 'wsj_0025.mrg', 'wsj_0026.mrg', 'wsj_0027.mrg', 'wsj_0028.mrg', 'wsj_0029.mrg', 'wsj_0030.mrg', 'wsj_0031.mrg', 'wsj_0032.mrg', 'wsj_0033.mrg', 'wsj_0034.mrg', 'wsj_0035.mrg', 'wsj_0036.mrg', 'wsj_0037.mrg', 'wsj_0038.mrg', 'wsj_0039.mrg', 'wsj_0040.mrg', 'wsj_0041.mrg', 'wsj_0042.mrg', 'wsj_0043.mrg', 'wsj_0044.mrg', 'wsj_0045.mrg', 'wsj_0046.mrg', 'wsj_0047.mrg', 'wsj_0048.mrg', 'wsj_0049.mrg', 'wsj_0050.mrg', 'wsj_0051.mrg', 'wsj_0052.mrg', 'wsj_0053.mrg', 'wsj_0054.mrg', 'wsj_0055.mrg', 'wsj_0056.mrg', 'wsj_0057.mrg', 'wsj_0058.mrg', 'wsj_0059.mrg', 'wsj_0060.mrg', 'wsj_0061.mrg', 'wsj_0062.mrg', 'wsj_0063.mrg', 'wsj_0064.mrg', 'wsj_0065.mrg', 'wsj_0066.mrg', 'wsj_0067.mrg', 'wsj_0068.mrg', 'wsj_0069.mrg', 'wsj_0070.mrg', 'wsj_0071.mrg', 'wsj_0072.mrg', 'wsj_0073.mrg', 'wsj_0074.mrg', 'wsj_0075.mrg', 'wsj_0076.mrg', 'wsj_0077.mrg', 'wsj_0078.mrg', 'wsj_0079.mrg', 'wsj_0080.mrg', 'wsj_0081.mrg', 'wsj_0082.mrg', 'wsj_0083.mrg', 'wsj_0084.mrg', 'wsj_0085.mrg', 'wsj_0086.mrg', 'wsj_0087.mrg', 'wsj_0088.mrg', 'wsj_0089.mrg', 'wsj_0090.mrg', 'wsj_0091.mrg', 'wsj_0092.mrg', 'wsj_0093.mrg', 'wsj_0094.mrg', 'wsj_0095.mrg', 'wsj_0096.mrg', 'wsj_0097.mrg', 'wsj_0098.mrg', 'wsj_0099.mrg', 'wsj_0100.mrg', 'wsj_0101.mrg', 'wsj_0102.mrg', 'wsj_0103.mrg', 'wsj_0104.mrg', 'wsj_0105.mrg', 'wsj_0106.mrg', 'wsj_0107.mrg', 'wsj_0108.mrg', 'wsj_0109.mrg', 'wsj_0110.mrg', 'wsj_0111.mrg', 'wsj_0112.mrg', 'wsj_0113.mrg', 'wsj_0114.mrg', 'wsj_0115.mrg', 'wsj_0116.mrg', 'wsj_0117.mrg', 'wsj_0118.mrg', 'wsj_0119.mrg', 'wsj_0120.mrg', 'wsj_0121.mrg', 'wsj_0122.mrg', 'wsj_0123.mrg', 'wsj_0124.mrg', 'wsj_0125.mrg', 'wsj_0126.mrg', 'wsj_0127.mrg', 'wsj_0128.mrg', 'wsj_0129.mrg', 'wsj_0130.mrg', 'wsj_0131.mrg', 'wsj_0132.mrg', 'wsj_0133.mrg', 'wsj_0134.mrg', 'wsj_0135.mrg', 'wsj_0136.mrg', 'wsj_0137.mrg', 'wsj_0138.mrg', 'wsj_0139.mrg', 'wsj_0140.mrg', 'wsj_0141.mrg', 'wsj_0142.mrg', 'wsj_0143.mrg', 'wsj_0144.mrg', 'wsj_0145.mrg', 'wsj_0146.mrg', 'wsj_0147.mrg', 'wsj_0148.mrg', 'wsj_0149.mrg', 'wsj_0150.mrg', 'wsj_0151.mrg', 'wsj_0152.mrg', 'wsj_0153.mrg', 'wsj_0154.mrg', 'wsj_0155.mrg', 'wsj_0156.mrg', 'wsj_0157.mrg', 'wsj_0158.mrg', 'wsj_0159.mrg', 'wsj_0160.mrg', 'wsj_0161.mrg', 'wsj_0162.mrg', 'wsj_0163.mrg', 'wsj_0164.mrg', 'wsj_0165.mrg', 'wsj_0166.mrg', 'wsj_0167.mrg', 'wsj_0168.mrg', 'wsj_0169.mrg', 'wsj_0170.mrg', 'wsj_0171.mrg', 'wsj_0172.mrg', 'wsj_0173.mrg', 'wsj_0174.mrg', 'wsj_0175.mrg', 'wsj_0176.mrg', 'wsj_0177.mrg', 'wsj_0178.mrg', 'wsj_0179.mrg', 'wsj_0180.mrg', 'wsj_0181.mrg', 'wsj_0182.mrg', 'wsj_0183.mrg', 'wsj_0184.mrg', 'wsj_0185.mrg', 'wsj_0186.mrg', 'wsj_0187.mrg', 'wsj_0188.mrg', 'wsj_0189.mrg', 'wsj_0190.mrg', 'wsj_0191.mrg', 'wsj_0192.mrg', 'wsj_0193.mrg', 'wsj_0194.mrg', 'wsj_0195.mrg', 'wsj_0196.mrg', 'wsj_0197.mrg', 'wsj_0198.mrg', 'wsj_0199.mrg']
```

# Penn Treebank Data (Cont.)

C:\nltk\_data\corpora\treebank\raw\wsj\_0001



The screenshot shows a Windows Notepad window titled "wsj\_0001 - Notepad". The window contains the following text:

```
.START

Pierre Vinken, 61 years old, will join the board as a nonexecutive
director Nov. 29.
Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing
group.
```

# Penn Treebank Data (Cont.)

- Tokenizing, Tagging in Penn Treebank
- sents() 이용
- 문장의 token化한 결과

```
1 treebank.sents('wsj_0001.mrg')
```

```
[['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'board', 'as', 'a', 'nonexecutive', 'director', 'No  
v.', '29', '.'], ['Mr.', 'Vinken', 'is', 'chairman', 'of', 'Elsevier', 'N.Y.', ',', 'the', 'Dutch', 'publishing', 'group', '.']]
```

# Penn Treebank Data (Cont.)

- 원문 보기
- join()

```
1 wsj_0001 = treebank.sents('wsj_0001.mrg')
2 for line in wsj_0001:
3     print(''.join(line))
```

Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29 .  
Mr. Vinken is chairman of Elsevier N.Y. , the Dutch publishing group .

# Penn Treebank Data (Cont.)

- 품사 Tagging
- tagged\_words()

```
1 treebank.tagged_words('wsj_0001.mrg')  
[('Pierre', 'NNP'), ('Vinken', 'NNP'), ('.', '.', '.', '.'), ...]
```

# Penn Treebank Data (Cont.)

- Data parsing 및 각 단어들의 상세 tagging
- parsed\_sents()

```
1 treebank.parsed_sents('wsj_0001.mrg')[0]
```

The Ghostscript executable isn't found.  
See <http://web.mit.edu/ghostscript/www/Install.htm>

If you're using a Mac, you can try installing  
<https://docs.brew.sh/Installation> then `brew install ghostscript`

```
Tree('S', [Tree('NP-SBJ', [Tree('NP', [Tree('NNP', ['Pierre']), Tree('NNP', ['Vinken'])]), Tree(',', ['.', '.']), Tree('ADJP', [Tree('NP', [Tree('CD', ['61']), Tree('NNS', ['years'])]), Tree('JJ', ['old'])]), Tree(',', ['.', '.']), Tree('VP', [Tree('MD', ['will']), Tree('VP', [Tree('VB', ['join']), Tree('NP', [Tree('DT', ['the']), Tree('NN', ['board'])]), Tree('PP-CLR', [Tree('IN', ['as']), Tree('NP', [Tree('DT', ['a']), Tree('JJ', ['nonexecutive'])]), Tree('NN', ['director'])])]), Tree('NP-TMP', [Tree('NNP', ['Nov.']), Tree('CD', ['29'])])]), Tree('.', ['.'])])])])
```

# book Data

- nltk.download('book')
- NLTK 책에서 사용하는 모든 Data downloads.

```
import nltk  
nltk.download('book', quiet=True)  
from nltk.book import *
```

```
1 import nltk  
2 nltk.download('book', quiet=True)  
True  
  
1 from nltk.book import *  
*** Introductory Examples for the NLTK Book ***  
Loading text1, ..., text9 and sent1, ..., sent9  
Type the name of the text or sentence to view it.  
Type: 'texts()' or 'sents()' to list the materials.  
text1: Moby Dick by Herman Melville 1851  
text2: Sense and Sensibility by Jane Austen 1811  
text3: The Book of Genesis  
text4: Inaugural Address Corpus  
text5: Chat Corpus  
text6: Monty Python and the Holy Grail  
text7: Wall Street Journal  
text8: Personals Corpus  
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

## book Data (Cont.)

- text1
- 허먼 멜빌(Herman Melville, 1819.8.1 ~ 1891.9.28, 미국)의 소설 Moby Dick

```
1 type(text1)
```

```
nltk.text.Text
```

```
1 text1
```

```
<Text : Moby Dick by Herman Melville 1851>
```

# Gutenberg Data

- nltk의 corpus module 중 parsing이 안된 원문 데이터
- Gutenberg 말뭉치는 저작권이 말소된 문학작품.

```
1 nltk.download('gutenberg')
```

```
[nltk_data] Downloading package gutenberg to C:\nltk_data...
[nltk_data] Package gutenberg is already up-to-date!
```

True

```
1 nltk.corpus.gutenberg.fileids()
```

```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

# Gutenberg Data (Cont.)

- 제인 오스틴(Jane Austen, 1775.12.16 ~ 1817.7.18, 영국의 소설가)의 소설 엠마(austen\_emma.txt)

```
1 emma = nltk.corpus.gutenberg.raw('austen-emma.txt')
2 emma[:289]
```

'[Emma by Jane Austen 1816] VOLUME I  
CHAPTER I  
Emma Woodhouse, handsome, clever, and rich, with a comfortable home and happy disposition, seemed to unite some of the best blessings of existence; and had lived nearly twenty-one years in the world without very little to distress or vex her.'

# Tokenizing

- 자연어 분석을 위해 문자열을 작은 단위(token)로 나누기
- Token → 나눈 작은 문자열 단위
- sent\_tokenize()
- PunktSentenceTokenizer의 NLTK의 instance

```
1 from nltk.tokenize import sent_tokenize  
2 sent_tokenize(emma[:1000])[3]
```

"Sixteen years had Miss Taylor been in Mr. Woodhouse's family, unless as a governess than a friend, very fond of both daughters, but particularly of Emma."

# Tokenizing (Cont.)

- 문자열을 입력받아 단어 단위의 Token list 출력하기
- word\_tokenize()

```
1 from nltk.tokenize import word_tokenize  
2 word_tokenize(emma[50:100])
```

```
['Emma',  
'Woodhouse',  
'',  
'handsome',  
'',  
'clever',  
'',  
'and',  
'rich',  
'',  
'with',  
'a']
```

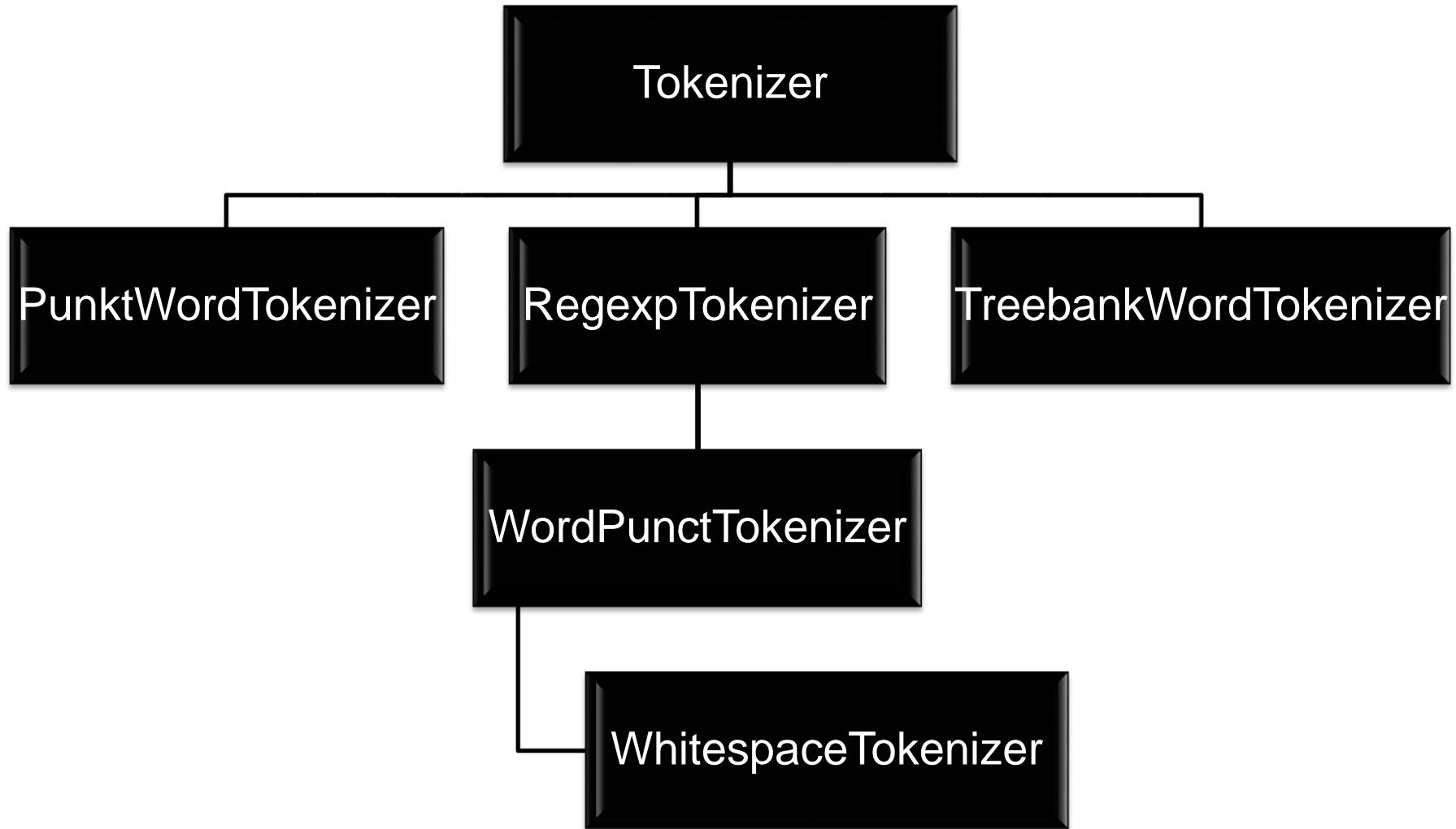
# Tokenizing (Cont.)

- Token化에 포함할 문자들을 정규표현식을 이용해서 지정하기
- RegexpTokenizer()

```
1 from nltk.tokenize import RegexpTokenizer  
2 retokenize = RegexpTokenizer('[\w]+')  
3 retokenize.tokenize(emma[50:100])
```

```
['Emma', 'Woodhouse', 'handsome', 'clever', 'and', 'rich', 'with', 'a']
```

# Tokenizer의 상속 Tree





# 한글 형태소 분석

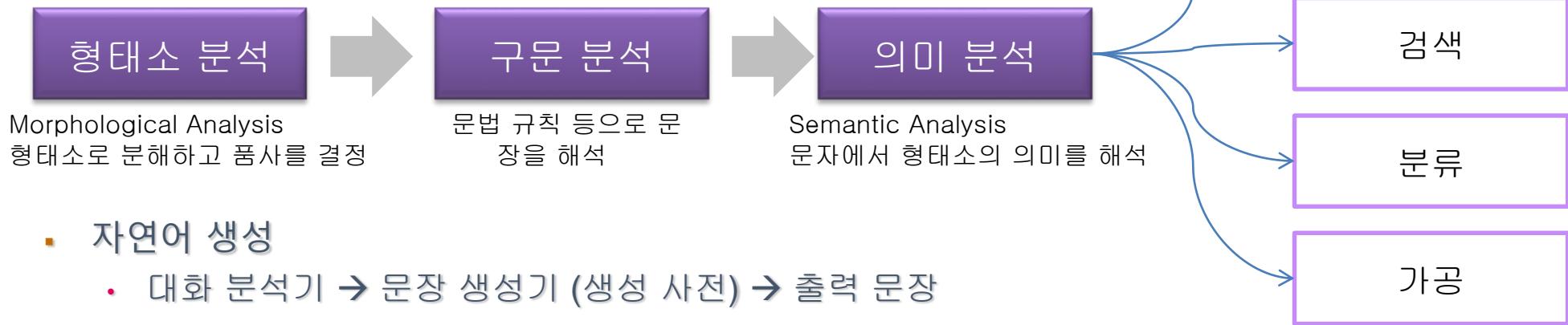
# 자연어 처리

## ● 자연어 처리 (Natural Language Processing, NLP)

- 자연어 : 사람들이 일상적으로 사용하는 언어
- 인공어 : Artificial Language, 사람들이 필요에 의해서 만든 언어  
에스페란토, C언어, Java 등

## ● 자연어 처리 분야

- 자연어 이해 : 형태소 분석 → 의미 분석 → 대화 분석



- 자연어 생성
  - 대화 분석기 → 문장 생성기 (생성 사전) → 출력 문장

## ● 자연어 처리의 활용 분야

- 맞춤법 검사, 번역기, 검색 엔진, 키워드 분석 등

● 문서 (Document) → 문단 (Paragraph) → 문장 (Sentence) → 어절 (Word phrase) → 형태소 (Morpheme) → 음절 (Syllable) → 음소 (Phoneme)

# 자연어 처리 용어

| 용어            | 상세   |
|---------------|--|
| 형태소(Morpheme) | 의미를 가진 최소 단위입니다.   |
| 용언            | 꾸미는 말(동사, 형용사)입니다.<br>용언은 어근 + 어미로 구성됩니다.  |
| 어근(Stem)      | 용언이 활용할 때, 원칙적으로 모양이 변하지 않는 부분입니다.   |
| 어미            | 용언이 활용할 때, 변하는 부분으로 문법적 기능 수행합니다.<br>어미에는 연결 어미, 선어말 어미 + 종결 어미가 있습니다.             |
| 자모            | 문자 체계의 한 요소(자음, 모음)입니다.  |
| 품사            | 명사, 대명사, 수사, 동사, 형용사, 관형사, 부사, 감탄사, 조사가 있습니다.                                      |
| 어절 분류         | 명사 + 주격 조사, 명사 + 목적격 조사, 명사 + 관형격 조사, 동사 + 연결 어미 또는 동사 + 선어말 어미 + 종결 어미 등으로 분류합니다. |
| 불용어(Stopword) | 검색 등에서 의미가 없어 무시되도록 설정된 단어들입니다.  |
| n-gram        | 문자의 빈도와 문자간 관계를 의미합니다. "안녕하세요"를 2-gram으로 나누면, "안녕", "녕하", "하세", "세요"로 나눌 수 있습니다.   |

# 형태소

## ● 형태소

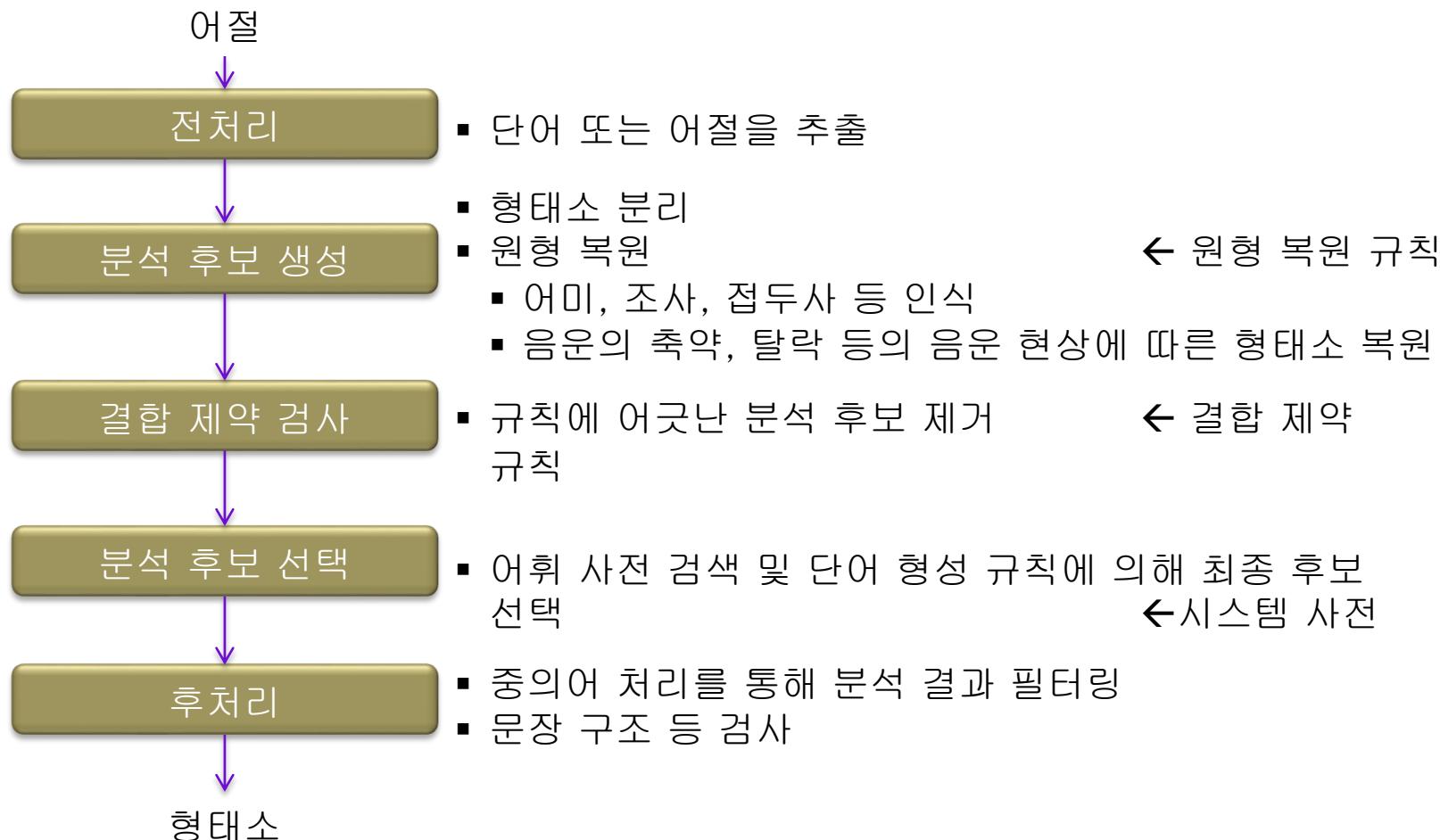
- 의미를 가진 최소의 단위
- 문법적, 관계적인 뜻을 나타내는 단어 또는 단어의 부분
- 체언 (명사, 대명사, 수사), 수식언 (관형사, 부사), 독립언 (감탄사), 관계언 (조사, 서술격조사),  
용언 (동사, 형용사, 조용보조어간), 부속어 (어미, 접미)

## ● 형태소의 종류

- 실질 형태소
  - 체언, 수식언, 감탄사, 용언의 어근
  - 구체적인 대상, 동작, 상태 등을 나타내는 형태소로 어휘 형태소
  - 검색 엔진에서 색인의 대상이 됩니다.
- 형식 형태소
  - 조사, 어말어미, 접미사, 접두사, 선어말어미
  - 형태소 간의 관계를 나타내는 형태소로 문법 형태소

# 형태소 분석

- 단어 또는 어절을 구성하는 각 형태소 분리
- 분리된 형태소의 기본형(어근) 및 품사 정보 추출
- 일반적인 형태소 분석 절차



# 형태소 분석 엔진

- KoNLPy

- Python용 형태소 분석기(Korean NLP in Python)(GPL v3+ License)
  - <http://konlpy.readthedocs.org/>

- KOMORAN

- Java로 만든 오픈소스 형태소 분석기(Apache v2 License)
  - <https://shineware.tistory.com/tag/KOMORAN/>

- HanNanum

- Java로 만들어진 형태소 분석기(GPL v3 License)
  - <http://semanticweb.kaist.ac.kr/home/index.php/HanNanum>

- KoNLP

- R용 형태소 분석기(Korean NLP)(GPL v3 License)
  - <https://github.com/haven-jeon/KoNLP>

# KoNLPy Package

- 한국어 정보처리를 위한 파이썬 패키지
- Open Source Software이며 [GPL v3 이상] 라이센스로 배포
- 공식 사이트는 다음과 같습니다.
  - <http://konlpy.org/en/latest/>
  - <https://github.com/konlpy/konlpy>
- KoNLPy 패키지 설치
  - KoNLPy는 JPyte1 패키지에 의존
  - (base) C:\Users\COM>pip install konlpy
- JDK를 설치해야 함
  - <http://jdk.java.net/12/> -> Builds -> Windows/x64 zip 다운로드
  - 압축 풀기
  - JAVA\_HOME 환경변수 설정

# 형태소 분석

- 형태소를 비롯하여, 어근, 접두사/접미사, 품사(POS, part-of-speech) 등 다양한 언어적 속성의 구조를 파악하는 것
- 품사 태그를 알아야 함
  - <https://konlpy-ko.readthedocs.io/ko/v0.4.3/morph/#comparison-between-pos-tagging-classes>
  - 엑셀파일 : <http://coderby.com/pds/14>

# 품사 Tagging

- 형태소의 뜻과 문맥을 고려하여 그것에 마크업을 하는 일
- 단어와 /(슬래시) 뒤에 품사가 표시되어 나누어지도록 하는 것
- 예)
  - 원문 : 가방에 들어가신다
  - 품사 태깅 : 가방/NNG + 에/JKM + 들어가/VV + 시/EPH + 냐다/EFN

# Komoran

- 2013년 Shineware에서 자바로 만든 오픈소스 한국어 형태소 분석기
- konlpy.tag.Komoran(jvmpath=None, dicpath=None)

| 메서드                          | 설명                       |
|------------------------------|--------------------------|
| morphs(phrase)               | 형태소 분석 문구를 반환.           |
| nouns(phrase)                | 명사를 추출.                  |
| pos(phrase,<br>flatten=True) | flatten이 False이면 어절을 보존. |

# Komoran (Cont.)

## ● Komoran 형태소 분석기를 이용해 Tagging하는 예

```
1 text = u"""아름답지만 다소 복잡하기도한 한국어는 전세계에서 13번째로 많이 사용되는 언어입니다."""
```

```
1 from konlpy.tag import Komoran  
2 komoran = Komoran()
```

```
1 print(komoran.morphs(text))
```

```
['아름답', '지만', '다소', '복잡', '하', '기', '도', '하', 'ㄴ', '한국어', '는', '전', '세계', '에서', '13',  
'번', '째', '로', '많이', '사용', '되', '는', '언어', '이', 'ㅂ니다', '.']
```

```
1 print(komoran.nouns(text))
```

```
['한국어', '전', '세계', '번', '사용', '언어']
```

```
1 print(komoran.pos(text))
```

```
[('아름답', 'VA'), ('지만', 'EC'), ('다소', 'MAG'), ('복잡', 'XR'), ('하', 'XSA'), ('기', 'ETN'), ('도', 'JX'),  
('하', 'VV'), ('ㄴ', 'ETM'), ('한국어', 'NNP'), ('는', 'JX'), ('전', 'NNG'), ('세계', 'NNG'), ('에서', 'JKB'),  
('13', 'SN'), ('번', 'NNB'), ('째', 'XSN'), ('로', 'JKB'), ('많이', 'MAG'), ('사용', 'NNG'), ('되', 'XSV'),  
('는', 'ETM'), ('언어', 'NNG'), ('이', 'VCP'), ('ㅂ니다', 'EF'), ('.', 'SF')]
```

# 말뭉치

- 컴퓨터를 이용해 자연어 분석 작업을 할 수 있도록 만든 문서 집합
- KoNLPy 패키지를 설치하고 사용할 수 있는 말뭉치
  - kolaw
    - 한국 법률 말뭉치
    - constitution.txt
  - kobill
    - 대한민국 국회 의안 말뭉치
    - 파일 ID는 의안 번호를 의미
    - 1809890.txt - 1809899.txt

# 말뭉치 (Cont.)

```
1 from konlpy.corpus import kolaw  
2 c = kolaw.open('constitution.txt').read()  
3 print(c[:100])
```

대한민국헌법

유구한 역사와 전통에 빛나는 우리 대한국민은 3·1운동으로 건립된 대한민국임시정부의 법통과 불의에 항거한 4·19민주이념을 계승하고, 조국의 민주개혁과 평화적 통일의

```
1 from konlpy.corpus import kobill  
2 d = kobill.open('1809890.txt').read()  
3 print(d[150:300])
```

초등학교 저학년의 경우에도 부모의 따뜻한 사랑과 보살핌이 필요  
한 나이이나, 현재 공무원이 자녀를 양육하기 위하여 육아휴직을 할  
수 있는 자녀의 나이는 만 6세 이하로 되어 있어 초등학교 저학년인  
자녀를 돌보기 위해서는 해당 부모님은 일자리를 그만 두어야



# Word Cloud

# Word Cloud

- 단어를 출현 빈도에 비례하는 크기로 단어의 빈도수를 시각화하는 기법
- 빈도분석이 된 데이터를 시각화하기 위해 Word Cloud를 사용
- `pip install wordcloud`

# Sample을 이용한 Word Cloud 그리기

```
1 from konlpy.corpus import kolaw  
2 data = kolaw.open('constitution.txt').read()
```

```
1 from konlpy.tag import Komoran  
2 komoran = Komoran()
```

헌법 말뭉치를 불러와 형태  
소 분석 후 명사만 추출함

```
1 print(komoran.nouns("%r"%data[0:1000]))
```

['대한민국', '헌법', '유구', '한', '역사', '전통', '국민', '운동', '건립', '대한민국', '임시', '정부', '법통', '불의', '항거', '민주', '이념', '계승', '조국', '민주개혁', '평화', '통일', '사명', '입각', '정의', '인도', '동포애', '민족', '단결', '사회', '폐습', '불의', '타파', '자율', '조화', '바탕', '자유', '민주', '기본', '질서', '정치', '경제', '사회', '문화', '영역', '각인', '기회', '능력', '최고', '도로', '발휘', '자유', '권리', '책임', '의무', '완수', '안', '국민', '생활', '균등', '향상', '밖', '항구', '세계', '평화', '인류', '공영', '이바지', '우리들의', '자소', '아저', '자유', '행복', '화부', '거', '다진', '녀', '7월 12일', '제

# Sample을 이용한 Word Cloud 그리기 (Cont.)

```
1 word_list = komoran.nouns("%r "%data[0:1000])
```

명사들을 공백으로 이어서  
하나의 문장으로 만듬

```
1 text = ' '.join(word_list)
```

```
1 text
```

'대한민국 헌법 유구 한 역사 전통 국민 운동 건립 대한민국 임시 정부 법통  
불의 항거 민주 이념 계승 조국 민주개혁 평화 통일 사명 입각 정의 인도 동  
포애 민족 단결 사회 폐습 불의 타파 자율 조화 바탕 자유 민주 기본 질서 정  
치 경제 사회 문화 영역 각인 기회 능력 최고 도로 발휘 자유 권리 책임 의무  
완수 안 국민 생활 균등 향상 밖 항구 세계 평화 인류 공영 이바지 우리들의  
자손 안전 자유 행복 확보 것 다짐 년 7월 12일 제정 차 개정 헌법 국회 의결  
국민 투표 개정 장 강 대한민국 민주공화국 대한민국 주권 국민 권력 국민 대  
한민국 국민 요건 법률 국가 법률 바 재외국민 보호 의무 대한민국 영토 한반  
도 부속 도서 대한민국 통일 지향 자유 민주 기본 질서 입각 평화 통일 정책  
수립 추진 대한민국 국제 평화 유지 노력 침략 전쟁 부인 국군 국가 안전 보  
장 국토방위 신성 의무 수행 사명 정치 중립 준수 헌법 체결 공포 조약 일반  
승인 국제 법규 국내법 효력 외국인 국제법 조약 바 지위 보장 공무원 국민  
전체 봉사자 국민 책임 공무원 신분 정치 중립 법률 바 보장 정당 설립 자유  
복수 정당'

# Sample을 이용한 Word Cloud 그리기 (Cont.)

```
1 import matplotlib.pyplot as plt  
2 %matplotlib inline  
3 from wordcloud import WordCloud
```

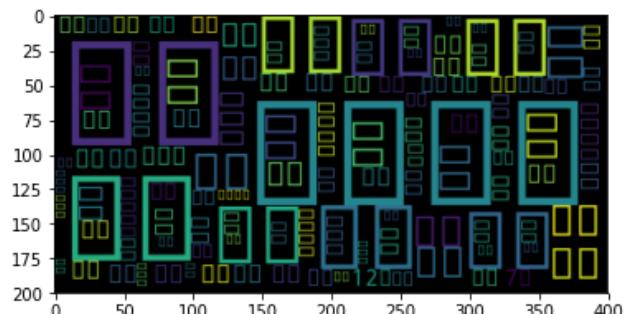
```
1 wordc = WordCloud()  
2 wordc.generate(text)
```

한글이 깨짐

```
<wordcloud.wordcloud.WordCloud at 0x1d532a20>
```

```
1 plt.figure()  
2 plt.imshow(wordc, interpolation="bilinear")
```

```
<matplotlib.image.AxesImage at 0x1f2da6d8>
```



# 한글 처리

```
1 wordc = WordCloud(background_color='white', max_words=20,  
2 font_path='c:/Windows/Fonts/malgun.ttf',  
3 relative_scaling=0.2)
```

```
1 wordc.generate(text)
```

```
<wordcloud.wordcloud.WordCloud at 0x1d536cc0>
```

```
1 plt.figure()  
2 plt.imshow(wordc, interpolation="bilinear")  
3 plt.axis('off')
```



# 전체 데이터를 이용한 Word Cloud 생성

```
1 word_list = komoran.nouns("%r %data")
2 text = ' '.join(word_list)
```

```
1 wordcloud = WordCloud(background_color='white', max_words=2000,
2 font_path='c:/Windows/Fonts/malgun.ttf',
3 relative_scaling=0.2)
```

```
1 wordcloud.generate(text)
```

```
<wordcloud.wordcloud.WordCloud at 0x203626a0>
```

```
1 plt.figure(figsize=(15, 10))
2 plt.imshow(wordcloud, interpolation="bilinear")
3 plt.axis('off')
```



# 불용어 사전 추가

```
1 from wordcloud import STOPWORDS  
2 from sklearn.feature_extraction.stop_words import ENGLISH_STOP_WORDS  
3  
4 불용어 = STOPWORDS | ENGLISH_STOP_WORDS | set(["대통령", "국가"])
```

```
1 wordcloud = WordCloud(background_color='white', max_words=2000,  
2                      stopwords=불용어,  
3                      font_path='c:/Windows/Fonts/malgun.ttf',  
4                      relative_scaling=0.2)  
5 wordcloud.generate(text)
```

```
<wordcloud.wordcloud.WordCloud at 0x204a2f28>
```

```
1 plt.figure(figsize=(15, 10))  
2 plt.imshow(wordcloud, interpolation="bilinear")  
3 plt.axis('off')
```



“대통령”, “국가” 단어를 불용어로 지정

# Masking

```
1 from PIL import Image  
2 import numpy as np  
3 img = Image.open("south_korea.png").convert("RGBA")  
4 mask = Image.new("RGB", img.size, (255,255,255))  
5 mask.paste(img, img)  
6 mask = np.array(mask)
```

- ✓ Word Cloud를 지정된 Mask Image에 맞도록 표시
- ✓ Mask Image는 <http://coderby.com/img/16>



```
1 wordcloud = WordCloud(background_color='white', max_words=2000,  
2 font_path='C:/Windows/Fonts/malgun.ttf',  
3 mask=mask, random_state=42)  
4 wordcloud.generate(text)  
5 wordcloud.to_file("result1.png")
```

# Pallet 변경

```
1 import random
2 def grey_color(word, font_size, position, orientation,
3                 random_state=None, **kwargs):
4     return 'hsl(0, 0%, %d%)' % random.randint(50, 100)
%표현
```

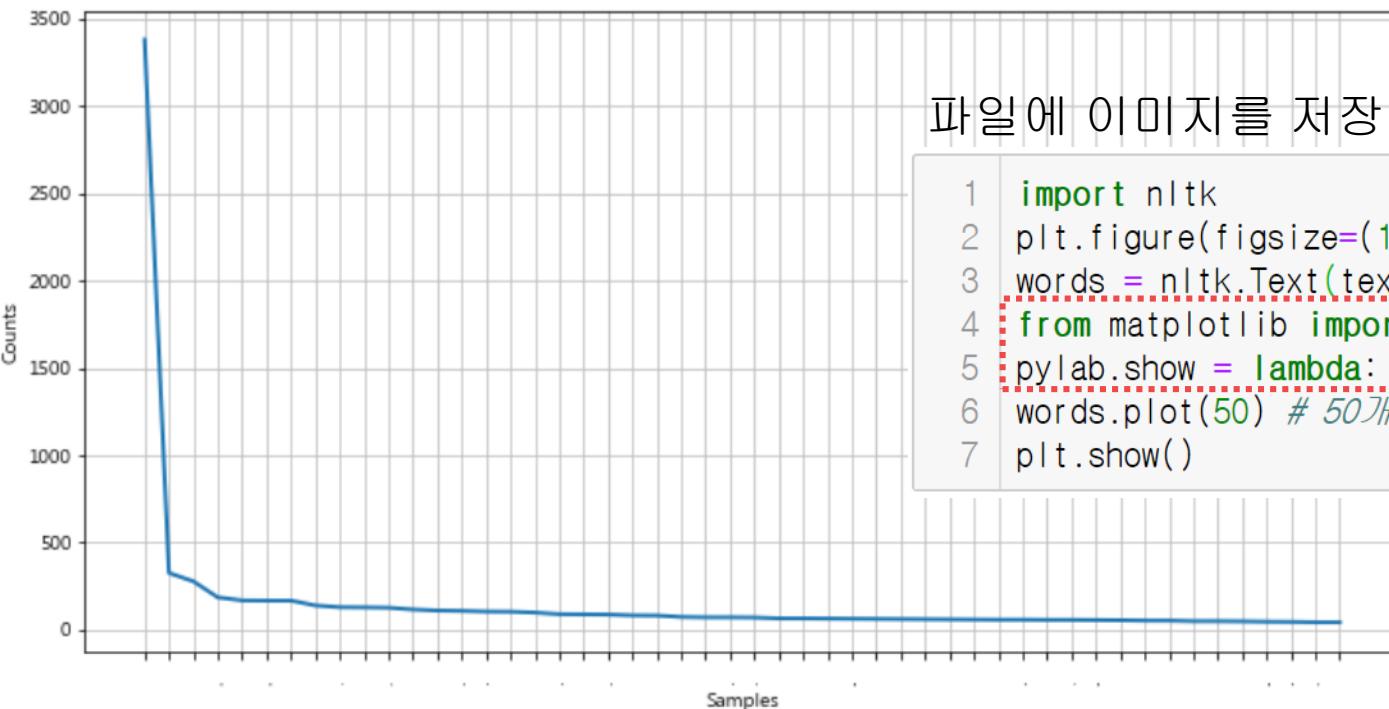
파일은 <http://coderby.com/img/15>에서



```
1 from PIL import Image
2 img = Image.open("south_korea_4x.png").convert("RGBA")
3 mask = Image.new("RGB", img.size, (255,255,255))
4 mask.paste(img, img)
5 mask = np.array(mask)
6 wordcloud = WordCloud(background_color='black', max_words=2000,
7                       font_path='C:/Windows/Fonts/malgun.ttf',
8                       mask=mask, random_state=42)
9 wordcloud.generate(text)
10 wordcloud.recolor(color_func=grey_color, random_state=3)
11 wordcloud.to_file("result2.png")
```

# 단어 빈도수 계산

```
1 import nltk  
2  
3 plt.figure(figsize=(12,6))  
4 words = nltk.Text(text)  
5 words.plot(50) # 50개만  
6 plt.show()
```



파일에 이미지를 저장하고 싶을 경우...

```
1 import nltk  
2 plt.figure(figsize=(12,6))  
3 words = nltk.Text(text)  
4 from matplotlib import pylab  
5 pylab.show = lambda: pylab.savefig('word_count.png')  
6 words.plot(50) # 50개만  
7 plt.show()
```



# Association Analysis

오렌지주스를 구매하는 사람은 와인을 구매할까?

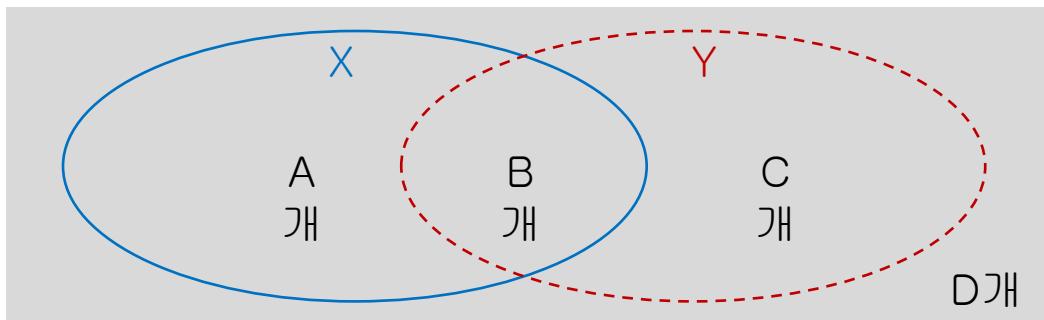
- 1 소주, 콜라, 와인
- 2 소주, 오렌지주스, 콜라
- 3 콜라, 맥주, 와인
- 4 소주, 콜라, 맥주
- 5 오렌지주스, 와인

# 연관 분석

- 연관 분석(Association Analysis)은 데이터들 사이에서 '자주 발생하는 속성을 찾는' 그리고 그 속성들 사이에 '연관성이 어느 정도 있는지'를 분석하는 방법
- 연관 규칙을 찾는 알고리즘으로는 apriori, FP-growth, DHP 알고리즘 등이 있음
- aprioir 알고리즘이 비교적 구현이 간단하고 성능 또한 높음
- pip install apyori

# 연관 분석 평가

- 지지도(Support)는 전체 Transaction에서 항목 집합(X, Y)을 포함하는 Transaction의 비율을 의미
- 신뢰도(Confidence)는 X를 포함하는 Transaction 중 Y도 포함하는 Transaction의 비율을 의미



범례 :  X를 포함하는 트랜잭션

Y를 포함하는 트랜잭션

$$\text{지지도} = B\text{개} / (A\text{개} + B\text{개} + C\text{개} + D\text{개})$$

$$\text{신뢰도} = B\text{개} / (A\text{개} + B\text{개})$$

# 연관 분석 평가 측도

| 평가 측도                       | 설명  |
|-----------------------------|---|
| 지지도<br>(Support)            | <ul style="list-style-type: none"><li>. 연관 규칙이 얼마나 중요한지에 대한 평가</li><li>. 낮은 지지도의 연관 규칙은 우연히 발생한 트랜잭션에서 생성된 규칙일 수 있음</li></ul>   |
| 신뢰도<br>(Confidence)         | <ul style="list-style-type: none"><li>. 연관 규칙이 얼마나 믿을 수 있는지 평가</li></ul>  |
| 향상도<br>(Lift)               | <ul style="list-style-type: none"><li>. <math>\text{Lift} = P(Y X) / P(Y) = P(X \cap Y) / P(X)P(Y) = \text{신뢰도} / P(Y)</math></li><li>. <math>X \rightarrow Y</math>의 신뢰도 / <math>Y</math>의 지지도</li><li>. 1 보다 작으면 음의 상관관계(설사약과 변비약), 1이면 상관관계 없음(과자와 후추), 1 보다 크면 양의 상관관계(빵과 버터)</li></ul> |
| 파이 계수<br>(Phi coefficient)  | <ul style="list-style-type: none"><li>. -1 : 완전 음의 상관관계, 0 : 상관관계 없음, 1 : 완전 양의 상관관계</li></ul>  |
| IS 측도<br>(Interest-Support) | <ul style="list-style-type: none"><li>. 비대칭적 이항 변수에 적합한 측도로 값이 클수록 상관관계가 높음</li><li>. <math>P(X, Y) / P(X) * P(Y)</math></li></ul>  |



# Transaction Data

# CSV file로부터 Transaction Data 생성

- 트랜잭션(Transaction)은 서로 관련 있는 데이터의 모음
- 연관분석 시 Transaction으로부터 연관 규칙을 도출
- Python의 Transaction Data는 list형식

```
1 import csv
2 with open('basket.csv', 'r', encoding='UTF8') as cf:
3     transactions = []
4     r = csv.reader(cf)
5     for row in r:
6         transactions.append(row)
7 
```

```
1 transactions
```

```
[['소주', '콜라', '와인'],
 ['소주', '오렌지주스', '콜라'],
 ['콜라', '맥주', '와인'],
 ['소주', '콜라', '맥주'],
 ['오렌지주스', '와인']]
```

basket.csv

|   |             |
|---|-------------|
| 1 | 소주,콜라,와인    |
| 2 | 소주,오렌지주스,콜라 |
| 3 | 콜라,맥주,와인    |
| 4 | 소주,콜라,맥주    |
| 5 | 오렌지주스,와인    |

# 연관 규칙 생성

- apriori() 연역적 알고리즘(A Priori Algorithm)을 사용하여 연관 규칙을 알아내는 함수

```
1 from apyori import apriori  
2 rules = apriori(transactions, min_support=0.1, min_confidence=0.1)  
3 results = list(rules)  
4 type(results)
```

list

```
1 results[0]
```

```
RelationRecord(items=frozenset({'맥주'}), support=0.4, ordered_statistics=[OrderedStatistic(items_base=frozenset(), items_add=frozenset({'맥주'}), confidence=0.4, lift=1.0)])
```

```
1 results[10]
```

```
RelationRecord(items=frozenset({'콜라', '소주'}), support=0.6, ordered_statistics=[OrderedStatistic(items_base=frozenset({'소주'}), items_add=frozenset({'콜라'}), confidence=1.0, lift=1.25), OrderedStatistic(items_base=frozenset({'콜라'}), items_add=frozenset({'소주'}), confidence=0.7499999999999999, lift=1.2499999999999998)])
```

이렇게 생성한 규칙은  
namedtuple 형식으로  
저장되어 있음

# 연관 규칙 형식

- 연관 규칙은 *namedtuple* 형식으로 저장되어 있기 때문에 원하는 값을 찾거나 출력하기 위해서는 연관 규칙 결과가 어떤 형식으로 저장되어 있는지 이해해야 함
- apyori.py 파일에 정의되어 있는 **RelationRecord**와 **OrderStatistic**은 다음처럼 *namedtuple*로 정의되어 있음

```
SupportRecord = namedtuple( 'SupportRecord', ('items', 'support'))
```

```
RelationRecord = namedtuple( 'RelationRecord', SupportRecord._fields + ('ordered_statistics',))
```

```
OrderedStatistic = namedtuple( 'OrderedStatistic', ('items_base', 'items_add', 'confidence', 'lift',))
```

C:\Users\사용자\Anaconda3\Lib\site-packages\apyori.py

# 연관 규칙 조회

```
1 print("lhs rhs support confidence lift")
2 for row in results:
3     support = row[1]
4     ordered_stat = row[2]
5     for ordered_item in ordered_stat:
6         lhs = [x for x in ordered_item[0]]
7         rhs = [x for x in ordered_item[1]]
8         confidence = ordered_item[2]
9         lift = ordered_item[3]
10        print(lhs, " => ", rhs, "支持度{:>5.4f}置信度{:>5.4f}提升度{:>5.4f}".
11                  format(support, confidence, lift))
12
```

| lhs       | rhs       | support | confidence | lift   |
|-----------|-----------|---------|------------|--------|
| [] =>     | ['맥주']    | 0.4000  | 0.4000     | 1.0000 |
| [] =>     | ['소주']    | 0.6000  | 0.6000     | 1.0000 |
| [] =>     | ['오렌지주스'] | 0.4000  | 0.4000     | 1.0000 |
| [] =>     | ['와인']    | 0.6000  | 0.6000     | 1.0000 |
| [] =>     | ['콜라']    | 0.8000  | 0.8000     | 1.0000 |
| ['맥주'] => | ['소주']    | 0.2000  | 0.5000     | 0.8333 |
| ['소주'] => | ['맥주']    | 0.2000  | 0.3333     | 0.8333 |
| ['맥주'] => | ['와인']    | 0.2000  | 0.5000     | 0.8333 |

# 연관 규칙 평가

- 오렌지주스를 구매한 사람은 와인을 구매할까?
- 앞의 결과에서 오렌지주스와 와인의 연관관계를 보면 다음과 같음
  - ['오렌지주스'] => ['와인'], 0.2000, 0.5000, 0.8
  - ['와인'] => ['오렌지주스'], 0.2000, 0.3333, 0.8
- 오렌지주스와 와인을 구매한 사람은 순서에 상관없이 전체의 20%
- 오렌지주스를 산 고객의 50%가 와인을 구매
- 향상도는 0.833이고 향상도가 1보다 작은 것은 음의 상관관계를 의미
- 그러므로 오렌지주스를 구매한 사람은 와인을 구매하지 않음

# 뉴스 기사 연관 분석 실습



# 뉴스 RSS 서버에서 링크 주소 가져오기

- JTBC 경제분야 Crawling 후 연관 분석
- RSS 주소: <http://fs.jtbc.joins.com/RSS/economy.xml>

```
1 import requests  
2 rss_url = "http://fs.jtbc.joins.com/RSS/economy.xml"  
3 jtbc_economy = requests.get(rss_url)
```

```
1 from bs4 import BeautifulSoup  
2 economy_news_list = BeautifulSoup(jtbc_economy.content, "xml")  
3 link_list = economy_news_list.select("item > link")
```

```
1 len(link_list)
```

20

```
1 link_list[0].text
```

'[http://news.jtbc.joins.com/article/article.aspx?news\\_id=NB11822526](http://news.jtbc.joins.com/article/article.aspx?news_id=NB11822526)'

## 기사 수집 및 형태소 분석

- 기사 원 글을 수집해서 기사의 본문 내용을 형태소 분석 후 명사만 뽑음

```
1 #!pip install konfpy
```

```
1 from konlpy.tag import Kkma  
2 kkma = Kkma()
```

# 연관 분석

## ■ 연관 규칙을 생성하고, DataFrame으로 만듦

```
1 from apyori import apriori  
2 rules = apriori(news, min_support=0.3, min_confidence=0.2)  
3 results = list(rules)
```

```
1 import pandas as pd  
2 result_df = pd.DataFrame(None,  
3                           columns=["lhs", "rhs", "support",  
4                                     "confidence", "lift"])  
5 index = 0  
6 for row in results:  
7     support = row[1]  
8     ordered_stat = row[2]  
9     for ordered_item in ordered_stat:  
10         lhs = " ".join(x.strip() for x in ordered_item[0])  
11         rhs = " ".join(x.strip() for x in ordered_item[1])  
12         confidence = ordered_item[2]  
13         lift = ordered_item[3]  
14         result_df.loc[index] = [lhs, rhs, support, confidence, lift]  
15         index = index + 1
```

# 연관 분석 탐색

## ■ 뉴스 기사 연관 분석

```
1 result_df.loc[(result_df.lhs.str.contains("택시")) &  
2                 (result_df.rhs=="공유")].sort_values(by=["lift"],  
3                                               ascending=False)
```

|      |                     | lhs | rhs | support  | confidence | lift |
|------|---------------------|-----|-----|----------|------------|------|
| 4084 | 서비스 대표 택시 앵커 업계     | 공유  | 0.3 | 1.000000 | 3.333333   |      |
| 217  | 경제 택시               | 공유  | 0.3 | 1.000000 | 3.333333   |      |
| 4402 | 혁신 이재웅 이재 택시 업계     | 공유  | 0.3 | 1.000000 | 3.333333   |      |
| 4632 | 서비스 대표 경제 택시 앵커 업계  | 공유  | 0.3 | 1.000000 | 3.333333   |      |
| 4653 | 서비스 이재 경제 택시 앵커 대표  | 공유  | 0.3 | 1.000000 | 3.333333   |      |
| 4667 | 이재웅 서비스 경제 택시 앵커 대표 | 공유  | 0.3 | 1.000000 | 3.333333   |      |
| 4681 | 혁신 서비스 경제 택시 앵커 대표  | 공유  | 0.3 | 1.000000 | 3.333333   |      |