



# 언어인지 알고리즘

2019.10

허진경

[hjk7902@gmail.com](mailto:hjk7902@gmail.com)

<http://bit.ly/30qrNNU>

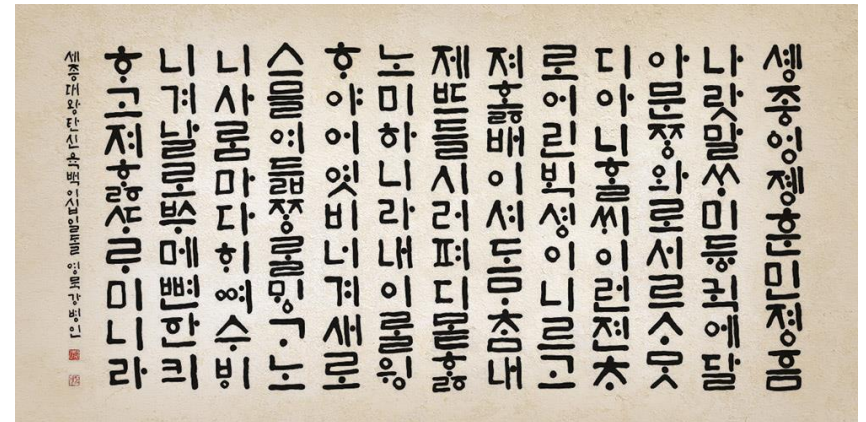
# CONTENTS

## 머신러닝을 이용한 자연어 처리

- 1 텍스트 마이닝
- 2 NLTK 자연어 처리 패키지
- 3 한글 형태소 분석을 위한 KoNLPy
- 4 워드 클라우드
- 5 연관분석
- 6 트랜잭션 데이터
- 7 뉴스기사 연관분석 실습
- 8 워드 임베딩

## 딥러닝을 이용한 자연어 처리

- 1 Keras 사용하기
- 2 RNN
- 3 LSTM
- 4 GRU



# 자연어 처리의 정의

**Natural language processing (NLP)** is a field of **computer science**, **artificial intelligence**, and **computational linguistics** concerned with the interactions between computers and human (natural) languages and, in particular, concerned with programming computers to fruitfully process large **natural language corpora**. Challenges in natural language processing frequently involve **natural language understanding**, **natural language generation**(frequently from formal, machine-readable logical forms), connecting language and machine perception, managing human-computer dialog systems, or some combination thereof.

- Wikipedia

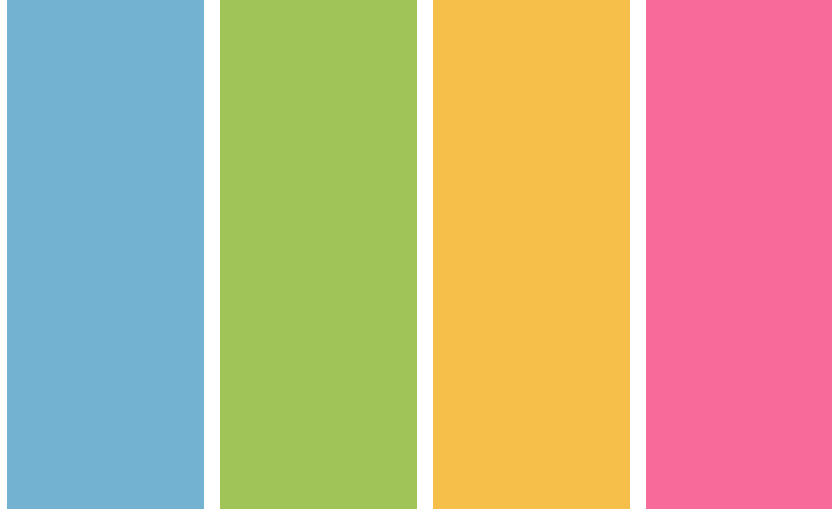
- 컴퓨터 과학, 인공지능, 컴퓨터 언어학의 결합 → 학제간 연구 필요
- 컴퓨터로 하여금 인간이 생산한 텍스트(비정형 데이터)에 담긴 정보를 처리하기 위한 알고리즘
- 자연어 이해(Natural Language Understanding)를 위한 필수 단계 → from morphology, syntax to semantics & pragmatics
- 인지 컴퓨팅(cognitive computing)의 필수 요소 → 지능형 질의응답/대화 시스템 등

[자연언어처리 기술의 필요성]

- 수많은 정보가 [비정형 데이터]에 포함되어 있으며 모든 데이터를 수작업으로 분석할 수 없음.
- 컴퓨터와 인간이 상호작용하기 위한 매개체 역할.

# 자연어 처리의 기술

기술명	설명
토큰 분리	미리 정의된 심볼을 기준으로 토큰 분리. 정의된 심볼이 언제나 토큰 경계는 아님.
어간추출	활용(conjugation)에 의해 변형된 단어의 원형을 복원. ex) running -> run
문장경계	인식 입력된 텍스트를 문장 단위로 분리(SBD: Sentence Boundary Detection)
띄어쓰기 복원	띄어쓰기가 없거나 잘못된 문장의 띄어쓰기를 수정
오류 단어 수정	의도와 달리 철자가 잘못 입력된 단어를 수정
형태소 분석	자연어 문장을 최소 의미 단위인 형태소로 분할, 형태소 태그 부착
구문 분석	자연어 문장을 구성하는 구(phrase) 사이의 계층적 관계를 트리로 재현
개체명 인식	문장의 고유명사나 장소/시간 표현의 범위 인식 및 클래스 태그 부착
의미역 인식	문장에 출현한 명사나 서술어의 의미적 역할 클래스 태그 부착
문장, 문서 군집화	텍스트의 표면적 형태나 의미값이 유사한 인스턴스들을 그룹화
문장, 문서 분류	입력된 문장, 문서의 범주 클래스 태그를 부착
문서 요약	입력된 텍스트의 핵심을 파악하여 짧은 형태의 요약문을 출력
이벤트 인식	문장이 기술하는 사건 정보를 객체 형태로 추출
오피니언 마이닝	대량 수집된 데이터에 포함된 사람들의 사건, 사물에 대한 의견을 추출, 통계 모델화, 예측 서비스 제공
의미 이해	형태소, 구문 분석 레벨을 넘어 메시지가 갖는 복합적인 의미 정보, 의도 추출
자동번역 시스템	소스 언어 텍스트를 동일/유사한 의미를 갖는 타겟 언어 텍스트로 변환
질의응답 시스템	자연어 질의에 대한 정답을 찾아 제공
검색 시스템	키워드 또는 자연어 질의로 찾고자 하는 문서의 리스트를 랭킹하여 제공
대화 시스템	형식언어가 아닌 자연언어로 컴퓨터와 메시지를 교환하면서 문제 해결, 정보 검색, 추천, 스케줄링, 상담 서비스 제공



# 머신러닝을 이용한 자연어 처리

언어인지 알고리즘



---

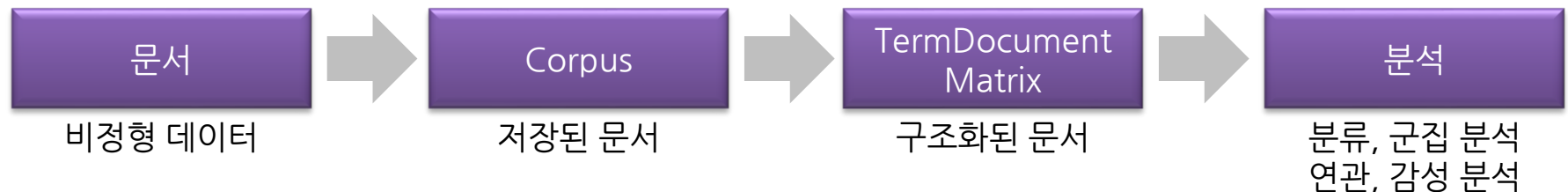
# 1절. 텍스트 마이닝



# 1.1. 텍스트 마이닝 개요

## 1절. 텍스트 마이닝

- 텍스트 마이닝(Text Mining)은 데이터 마이닝(Data Mining)의 한 분야에 속하며 뉴스 기사, SNS 글 등 **자연어에서 의미 있는 정보**를 찾는 것
- 텍스트 마이닝은 다양한 형태의 비정형 문서 데이터로부터 문서별 단어의 행렬(Matrix)을 만든 후, 여러 가지 분석 기법과 데이터 마이닝 기법을 사용하여 **통찰(Insight)을 얻거나 의사결정을 지원**하기 위해 사용



# 1.2 자연어 처리와 텍스트 마이닝

## 1절. 텍스트 마이닝

### ● 자연어 처리 학습 주제

- 텍스트 전처리 : 토큰화, 정제, 형태소 분석, 불용어 처리. 레이블 인코딩
- 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
- 토픽 모델링(Topic Modeling) : 텍스트 본문의 숨겨진 의미 구조를 발견하기 위해 사용되는 텍스트 마이닝 기법
- 연관 분석(Association) : 문서 내의 단어들을 이용해서 연관 분석을 실시
- 딥러닝을 이용한 자연어 처리 : RNN, LSTM 등의 인공신경망 알고리즘을 이용해서 딥러닝으로 자연어 처리
- 워드 임베딩(Word Imbedding) : 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
- 태깅(Tagging) : 각 단어가 어떤 유형에 속해 있는지를 알아내는 것
- 번역(Translation) : 챗봇(Chatbot) 또는 기계 번역(Machine Translation)





---



## 2절. NLTK 자연어 처리 패키지

# NLTK(Natural Language Toolkit) 패키지

2절. NLTK 자연어 처리 패키지

- 교육용으로 개발된 자연어 처리와 문서 분석용 파이썬 패키지
- NLTK 패키지가 제공하는 주요 기능
  - 말뭉치(corpus)
  - 토큰 생성(tokenizing)
  - 형태소 분석(morphological analysis)
  - 품사 태깅(POS tagging)
- 한글 형태소 분석기를 사용할 것이므로 이 절은 설명하지 않습니다.



---

## 3절. 한글 형태소 분석

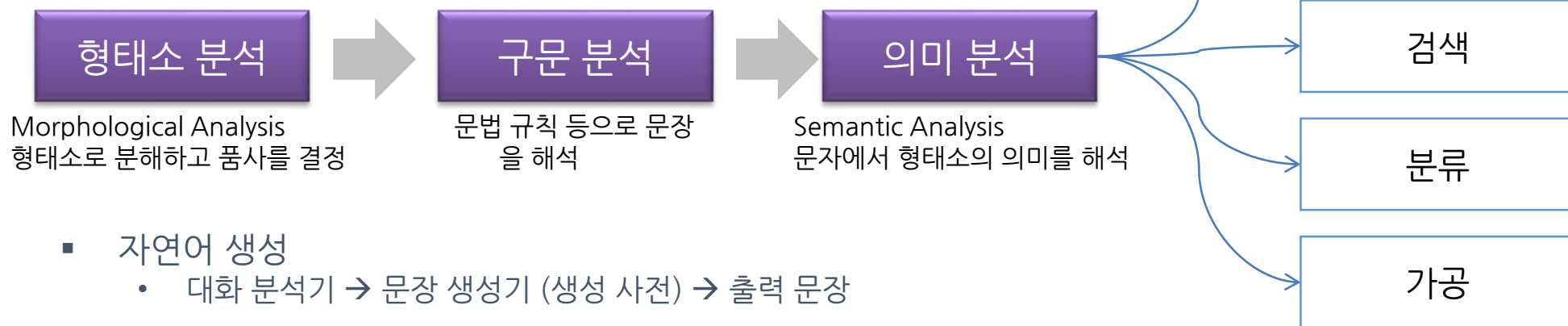


# 3.1. 자연어 처리

3절. 한글 형태소 분석

- 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

3절. 한글 형태소 분석

용어	상세
형태소(Morpheme)	의미를 가진 최소 단위입니다.
용언	꾸미는 말(동사, 형용사)입니다. 용언은 어근 + 어미로 구성됩니다.
어근(Stem)	용언이 활용할 때, 원칙적으로 모양이 변하지 않는 부분입니다.
어미	용언이 활용할 때, 변하는 부분으로 문법적 기능 수행합니다. 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.

## 3.2. 형태소

3절. 한글 형태소 분석

### ● 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)

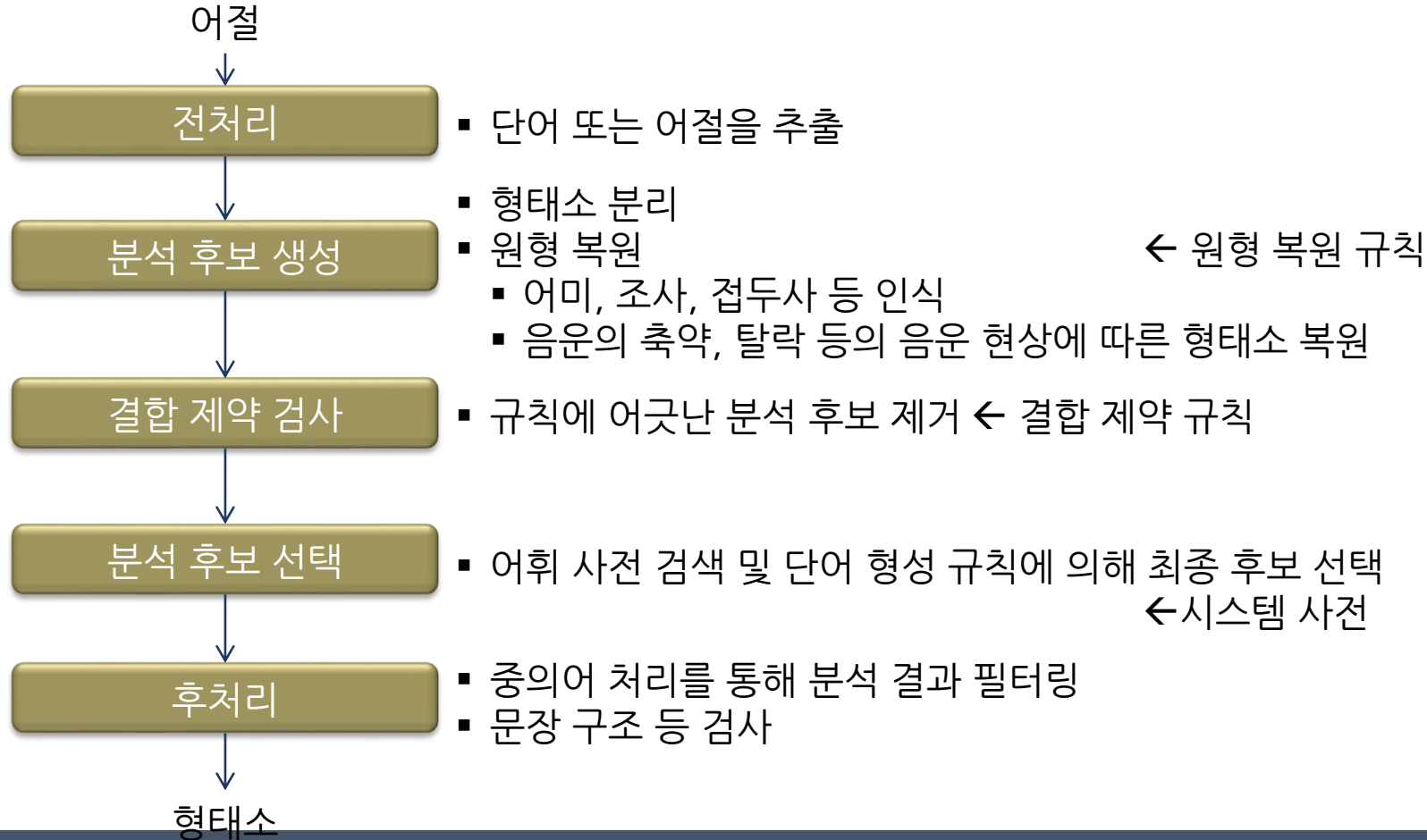
### ● 형태소의 종류

- 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

## 3.3. 형태소 분석

3절. 한글 형태소 분석

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



# 형태소 분석 엔진

3절. 한글 형태소 분석 > 3.2. 형태소 분석

- KoNLPy
  - Python용 형태소 분석기(Korean NLP in Python)(GPL v3+라이선스)
  - <http://konlpy.readthedocs.org/>
- KOMORAN
  - 자바로 만든 오픈소스 형태소 분석기(Apache v2 라이선스)
  - <https://shineware.tistory.com/tag/KOMORAN/>
- HanNanum
  - 자바로 만들어진 형태소 분석기(GPL v3 라이선스)
  - <http://semanticweb.kaist.ac.kr/home/index.php/HanNanum>
- KoNLP
  - R용 형태소 분석기(Korean NLP)(GPL v3 라이선스)
  - <https://github.com/haven-jeon/KoNLP>



## 3.4. KoNLPy 패키지

3절. 한글 형태소 분석

- 한국어 정보처리를 위한 파이썬 패키지
- 오픈 소스 소프트웨어이며 [GPL v3 이상] 라이선스로 배포
- 공식 사이트는 다음과 같습니다.
  - <http://konlpy.org/en/latest/>
  - <https://github.com/konlpy/konlpy>
- KoNLPy 패키지 설치
  - KoNLPy는 JPytype1 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

## 3.5. 형태소 분석

3절. 한글 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>

## 3.6. 품사 태깅

3절. 한글 형태소 분석

- 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 + ㄴ 다/EFN

### 3) Komoran

3절. 한글 형태소 분석 > 3.6. 품사 태깅

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

메서드	설명
<code>morphs(phrase)</code>	형태소 분석 문구를 반환합니다.
<code>nouns(phrase)</code>	명사를 추출합니다.
<code>pos(phrase, flatten=True)</code>	<code>flatten</code> 이 <code>False</code> 이면 어절을 보존합니다.

### 3) Komoran

3절. 한글 형태소 분석 > 3.6. 품사 태깅

#### ● 코모란 형태소 분석기를 이용해 태깅하는 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', '입니다', '.']

```
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'), ('입니다', 'EF'), ('.', 'SF')]

## 3.7. 말뭉치

3절. 한글 형태소 분석

- 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

## 3.7. 말뭉치

3절. 한글 형태소 분석

```
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 법  
통과 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



---

## 4절. 워드 클라우드





## 4절. 워드클라우드

#### 4절. 워드클라우드

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



## 4.1. 샘플을 이용한 워드클라우드 그리기

4절. 워드클라우드

```
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 형태소  
분석 후 명사만 추출함

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

['대한민국', '헌법', '유구', '한', '역사', '전통', '국민', '운동', '건  
립', '대한민국', '임시', '정부', '법통', '불의', '항거', '민주', '이념',  
'계승', '조국', '민주개혁', '평화', '통일', '사명', '입각', '정의', '인  
도', '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', '자  
소', '아저', '자유', '해본', '화부', '거', '다진', '녀', '7월 12일', '제

## 4.1. 샘플을 이용한 워드클라우드 그리기

### 4. 워드 클라우드

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

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

명사들을 공백으로 이어서 하나의 문장으로 만들

```
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 동포애 민족 단결 사회 폐습 불의 타파 자율 조화 바탕 자유 민주 기본 질서 정치 경제 사회 문화 영역 각인 기회 능력 최고 도로 발취 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 대한민국 국민 요건 법률 국가 법률 바 재외국민 보호 의무 대한민국 영토 한반도 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 보장 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'

## 4.1. 샘플을 이용한 워드클라우드 그리기

### 4. 워드 클라우드

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

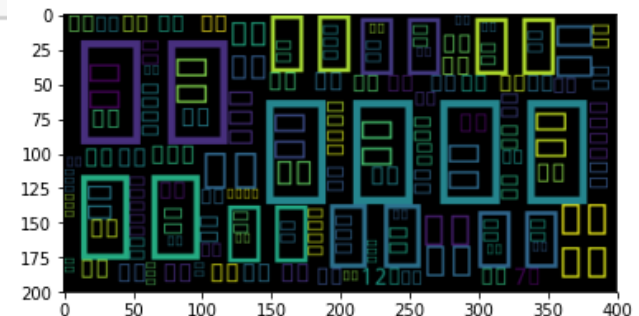
```
1 wordc = WordCloud()
2 wordc.generate(text)
```

<wordcloud.wordcloud.WordCloud at 0x1d532a20>

워드클라우드를 그렸지만 한  
글이 깨짐

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

<matplotlib.image.AxesImage at 0x1f2da6d8>



## 4.2. 한글 처리

### 4. 워드 클라우드

```
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)
```

워드클라우드에 한글을 표시  
하기 위해 워드클라우드 객체  
를 생성할 때 폰트를 지정

<wordcloud.wordcloud.WordCloud at 0x1d536cc0>

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



## 4.3. 전체 데이터를 이용한 워드클라우드 생성

### 4. 워드 클라우드

```
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')
```





#### 4.4. 불용어 사전 추가

#### 4. 워드 클라우드

```
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

## 4.5 마스크킹

#### 4. 워드 클라우드

```
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)
```

- ✓ 워드 클라우드를 지정된 마스크 이미지에 맞도록 표시
- ✓ 마스크 이미지는 <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")
```



## 4.6. 팔래트 변경

#### 4. 워드 클라우드

파일은 <http://coderby.com/img/15> 에서

```
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

%표현

```

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")

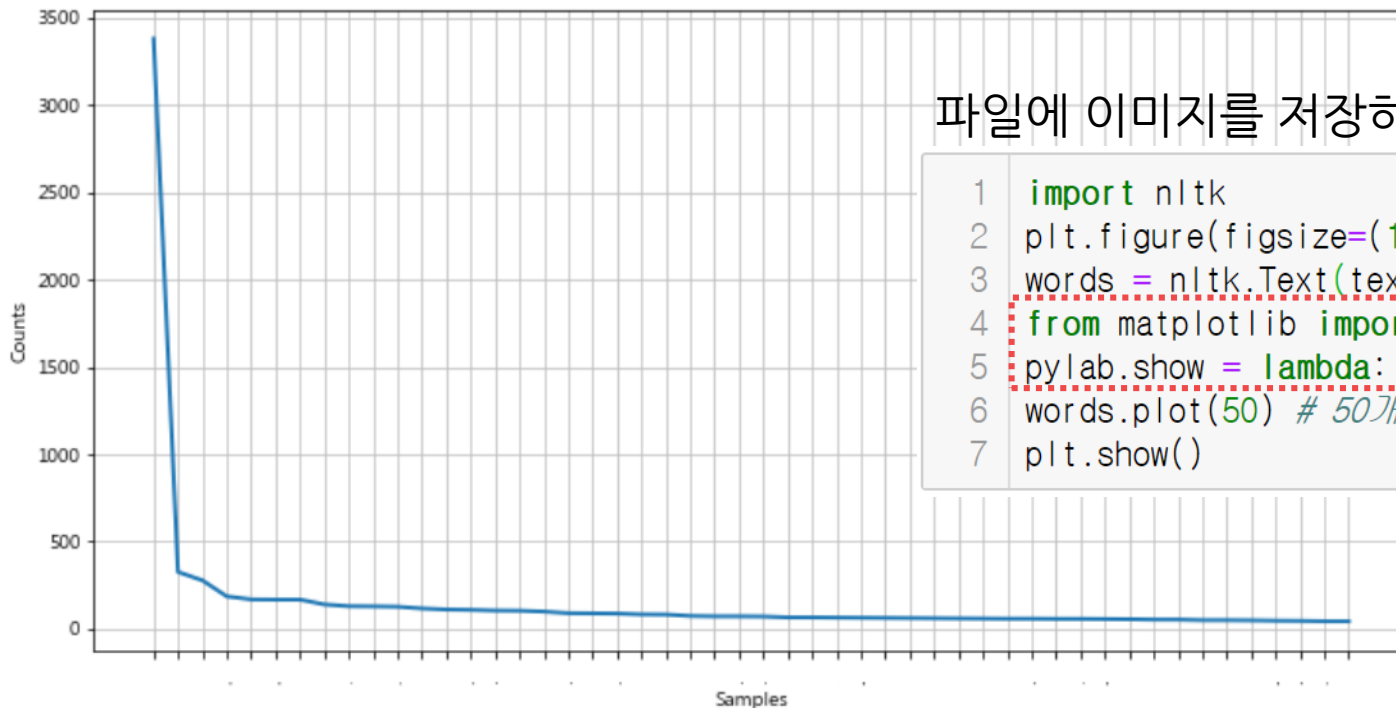
```



## 4.7. 단어 빈도수 계산

### 4. 워드 클라우드

```
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()
```



---



## 5절. 연관 분석(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

# 5.1. 연관 분석

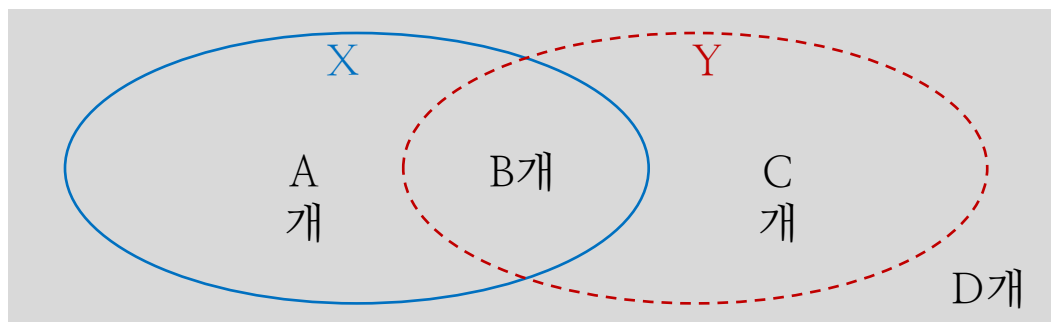
5절. 연관 분석(Association Analysis) 개요

- 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
- `apriori` 알고리즘이 비교적 구현이 간단하고 성능 또한 높음
- `pip install apyori`

## 5.2. 연관 분석 평가

5절. 연관 분석(Association Analysis) 개요

- 지지도(Support)는 전체 트랜잭션에서 항목 집합(X, Y)을 포함하는 트랜잭션의 비율을 의미
- 신뢰도(Confidence)는 X를 포함하는 트랜잭션 중 Y도 포함하는 트랜잭션의 비율을 의미



범례 : ○ X를 포함하는 트랜잭션

○ Y를 포함하는 트랜잭션

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

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

# 연관 분석 평가 척도

5절. 연관 분석(Association Analysis) 개요

평가 척도	설명
지지도 (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 (Confidence)	<ul style="list-style-type: none"><li>· 연관 규칙이 얼마나 믿을 수 있는지 평가</li></ul>
향상도 (Lift)	<ul style="list-style-type: none"><li>· <math>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 / Y의 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 (Phi coefficient)	<ul style="list-style-type: none"><li>· -1 : 완전 음의 상관관계, 0 : 상관관계 없음, 1 : 완전 양의 상관관계</li></ul>
IS 척도 (Interest-Support)	<ul style="list-style-type: none"><li>· 비대칭적 이항 변수에 적합한 척도로 값이 클수록 상관관계가 높음</li><li>· <math>P(X, Y) / P(X) * P(Y)</math></li></ul>



---

## 6절. 트랜잭션 데이터





## 6.1. CSV 파일로부터 트랜잭션 데이터 생성

6절. 트랜잭션 데이터

- 트랜잭션(Transaction)은 서로 관련 있는 데이터의 모음
- 연관분석 시 트랜잭션으로부터 연관 규칙을 도출
- 파이썬의 트랜잭션 데이터는 리스트의 리스트 형식

```
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
```

basket.csv

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

```
1 transactions
```

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



---



## 7절. 연관 규칙(Association Rule)

# 7.1. 연관 규칙 생성

7절. 연관 분석(Association Analysis)

- **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

7절. 연관 분석(Association Analysis) > 7.1. 연관 규칙 생성

- 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 RelationRecord와 OrderStatistic은 다음처럼 namedtuple로 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

## 7.2. 연관 규칙 조회

7절. 연관 분석(Association Analysis)

```
1 print("lhs Wtrhs WtWtsupportWtWtconfidenceWtWtlift")
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, "Wt{:>5.4f}Wt{:>5.4f}Wt{:>5.4f}".W
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

## 7.3. 연관 규칙 평가

7절. 연관 분석(Association Analysis)

- 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 0.833 이고 향상도가 1보다 작은 것은 음의 상관관계를 의미
- 그러므로 오렌지주스를 구매한 사람은 와인을 구매하지 않음



---



## 8절. 뉴스 기사 연관 분석 실습

## 8.1. 뉴스 RSS 서버에서 링크 주소 가져오기

8절. 뉴스 기사 연관 분석

- JTBC 경제분야 크롤링 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)'



## 8.2. KoNLPy 패키지

8절. 뉴스 기사 연관 분석

- 한국어 정보처리를 위한 파이썬 패키지
- 오픈 소스 소프트웨어이며 [GPL v3 이상] 라이선스로 배포
- 공식 사이트는 다음과 같습니다.
  - <http://konlpy.org/en/latest/>
  - <https://github.com/konlpy/konlpy>
- KoNLPy 패키지 설치
  - KoNLPy는 JPytype1 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

## 8.3. 기사 수집 및 형태소 분석

8절. 뉴스 기사 연관 분석

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

```
1 #!/pip install konlpy
```

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

```
1 news = [] # transaction  
2 for link in link_list:  
3     news_url = link.text  
4     news_response = requests.get(news_url)  
5     news_soup = BeautifulSoup(news_response.content, "html.parser")  
6     news_content = news_soup.select_one("#articlebody > .article_content")  
7     news.append(list(filter(lambda word: len(word)>1,  
8                             kkma.nouns(news_content.text))))
```

## 8.4. 연관 분석

8절. 뉴스 기사 연관 분석

- 연관 규칙을 생성하고, 데이터프레임으로 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
```

## 8.5. 연관 분석 탐색

8절. 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



---

## 9절. Word Embedding



# 워드 임베딩(Word Embedding)

9절. Word Embedding

- 워드 임베딩(Word Embedding)

- 단어를 벡터로 표현하는 대표적인 방법으로 주로 희소 표현에서 밀집 표현(Dense Vector)으로 변환하는 것을 의미
- 이 밀집 벡터를 워드 임베딩 과정을 통해 나온 결과라고 하여 임베딩 벡터(Embedding Vector)라고 함

- 희소 표현(Sparse Representation)

- 원-핫 인코딩을 통해서 나온 원-핫 벡터들은 표현하고자 하는 단어의 인덱스의 값만 1이고, 나머지 인덱스에는 전부 0으로 표현되는 벡터 표현 방법
- 벡터 또는 행렬(Matrix)의 값이 대부분이 0으로 표현되는 방법
- 원-핫 벡터는 희소 벡터(Sparse Vector)
- 단어간 유사성을 표현할 수 없다는 단점

- 밀집 표현(Dense Representation) 또는 분산 표현(Distributed Representation)

- 단어의 '의미'를 다차원 공간에 벡터화하는 방법
- '비슷한 위치에서 등장하는 단어들은 비슷한 의미를 가진다'라는 가정
- 분산 표현을 이용하여 단어의 의미를 벡터화하는 작업은 워드 임베딩(Embedding) 작업에 속함
- 사용자가 설정한 값으로 모든 단어의 벡터 표현의 차원을 맞춤
- 이 과정에서 더 이상 0과 1만 가진 값이 아니라 실수값을 가지게 됨
- 벡터의 차원이 조밀해졌다고 하여 밀집 벡터(Dense Vector)라고 함

# Word2Vec

## 9절. Word Embedding

- 단어 간 유사성을 고려하기 위해서는 단어의 의미를 벡터화하기 위한 패키지
- 벡터화 방법
  - CBOW(Continuous Bag of Words)
    - 주변에 있는 단어들을 가지고, 중간에 있는 단어들을 예측하는 방법
  - Skip-Gram
    - 중간에 있는 단어로 주변 단어들을 예측하는 방법
- ***class Word2Vec(sentences=None, size=100, window=5, min\_count=5, workers=3)***
  - size : 특징 벡터의 차원
  - window : 문장 내 현재 단어와 예측 단어 사이의 최대 거리
  - min\_count : 총 빈도가 이보다 낮은 모든 단어를 무시함
  - workers : 모델을 훈련시킬 쓰레드의 수
  - <https://www.pydoc.io/pypi/gensim-3.2.0/autoapi/models/word2vec/index.html#models.word2vec.Word2Vec>
- 패키지 설치
  - pip install gensim

# 뉴스기사 워드 임베딩 하기

9절. Word Embedding

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

```
1 news = [] # transaction
2 for link in link_list:
3     news_url = link.text
4     news_response = requests.get(news_url)
5     news_soup = BeautifulSoup(news_response.content, "html.parser")
6     news_content = news_soup.select_one("#articlebody > .article_content")
7     news.append(list(filter(lambda word: len(word)>1,
8                             kkma.nouns(news_content.text))))
```



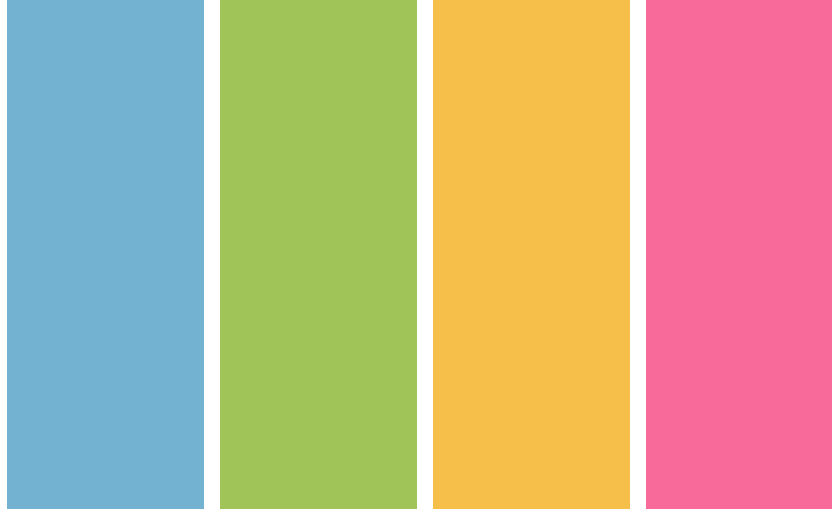
# 뉴스기사 워드 임베딩 하기

9절. Word Embedding

```
1 from gensim.models import Word2Vec
2 model = Word2Vec(news, size=100, window=5, min_count=2, workers=-1)
```

```
1 model.wv.most_similar("누진제")
```

```
[('이사회', 0.2872373163700104),
 ('신도시', 0.2726539075374603),
 ('협의체', 0.2622441053390503),
 ('투플러스', 0.23011885583400726),
 ('지방', 0.22897236049175262),
 ('전기', 0.2165725827217102),
 ('홍남기', 0.21561753749847412),
 ('수법', 0.20883645117282867),
 ('국토교통부', 0.20025959610939026),
 ('50만원', 0.19682154059410095)]
```



# 딥러닝을 이용한 자연어 처리

---

언어인지 알고리즘



---

# 1절. Keras



# Keras

Keras

- 사용자가 손쉽게 딥 러닝을 구현할 수 있도록 도와주는 상위 레벨의 인터페이스
- <https://keras.io/>
- Keras is compatible with: **Python 2.7-3.6.**



# Keras

# Sequential model

Keras

<https://keras.io/getting-started/sequential-model-guide/>

- 예

```
from keras.models import Sequential
from keras.layers import Dense
model=Sequential()
model.add(Dense(3, input_dim=4, activation='softmax'))
```

- 장점

- sequential API로 모델을 설계하는 것은 직관적이고 편리함

- 단점

- 단순히 층을 쌓는 것만으로는 모든 인공신경망을 구현할 수 없다. 즉, 복잡한 인공 신경망을 구현할 수 없다.

# Functional API

Keras

<https://keras.io/getting-started/functional-api-guide/>

- 레이어 인스턴스는 호출 가능하며(텐서에서), 텐서를 반환
- 다음 입력 텐서와 출력 텐서를 사용하여 모델을 정의 할 수 있음
- 예

```
from keras.models import Model
from keras.layers import Input, Dense
```

```
a = Input(shape=(32,))
b = Dense(32)(a)
model = Model(inputs=a, outputs=b)
```

```
x = Dense(64)(input)
위의 코드는 아래의 코드와 같음
x = Dense(64)
x(input)
```

- 다중 입력 및 다중 출력 모형 생성 가능
  - `model = Model(inputs=[a1, a2], outputs=[b1, b2, b3])`

# Tensorflow에서 설명하는 예

Keras

```
1. import tensorflow as tf
2. mnist = tf.keras.datasets.mnist

3. (x_train, y_train), (x_test, y_test) = mnist.load_data()
4. x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
5. model = tf.keras.models.Sequential([
6.     tf.keras.layers.Flatten(input_shape=(28, 28)),
7.     tf.keras.layers.Dense(360, activation='relu'),
8.     tf.keras.layers.Dropout(0.1),
9.     tf.keras.layers.Dense(128, activation='relu'),
10.    tf.keras.layers.Dropout(0.2),
11.    tf.keras.layers.Dense(10, activation='softmax')
12. ])
```

모델 정의

- Dense는 뉴런의 수, 활성화 함수를 지정
- Dense의 수는 레이어의 수

```
13. model.compile(optimizer='SGD',
14.                 loss='sparse_categorical_crossentropy',
15.                 metrics=['accuracy'])
```

옵티마이저, 손실함수, 평가 함수를 정의

```
16. model.fit(x_train, y_train, batch_size=100, epochs=5, verbose=0)
17. model.evaluate(x_test, y_test)
```

batch\_size : 그래디언트를 갱신할 때마다 전달하는 데이터의 수. default 32.  
epochs : 학습횟수 지정  
verbose : 로그레벨 지정(0=silent, 1=progress bar, 2=one line per epoch)

# Sequential 클래스를 이용한 모델 정의 방법

Keras

```
1. from keras.datasets import mnist
2. from keras.models import Sequential
3. from keras.layers import Flatten, Dense, Dropout
```

<https://keras.io/models/sequential/>

```
4. (x_train, y_train), (x_test, y_test) = mnist.load_data()
5. x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
6. model = Sequential([
7.     Flatten(input_shape=(28, 28)),
8.     Dense(360, activation='relu'),
9.     Dropout(0.1),
10.    Dense(128, activation='relu'),
11.    Dropout(0.2),
12.    Dense(10, activation='softmax')
13. ])
```



```
model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(360, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))
```

model.add() 함수를 이용해서 모델을 정의

```
14. model.compile(optimizer='SGD',
15.                loss='sparse_categorical_crossentropy',
16.                metrics=['accuracy'])
17. model.fit(x_train, y_train, batch_size=100, epochs=5, verbose=1)
18. model.evaluate(x_test, y_test)
```



# model.add()

Keras

```
1.  from keras.datasets import mnist
2.  from keras.models import Sequential
3.  from keras.layers import Flatten, Dense, Dropout

4.  (x_train, y_train), (x_test, y_test) = mnist.load_data()
5.  x_train, x_test = x_train / 255.0, x_test / 255.0

6.  model = Sequential()
7.  model.add(Flatten(input_shape=(28, 28)))
8.  model.add(Dense(360, activation='relu'))
9.  model.add(Dropout(0.1))
10. model.add(Dense(128, activation='relu'))
11. model.add(Dropout(0.2))
12. model.add(Dense(10, activation='softmax'))

13. model.compile(optimizer='SGD',
14.                loss='sparse_categorical_crossentropy',
15.                metrics=['accuracy'])

16. model.fit(x_train, y_train, batch_size=100, epochs=5, verbose=1)
17. model.evaluate(x_test, y_test)
```

# optimizer

Keras

옵티마이저	클래스(기본 파라미터)	설명
SGD	<code>keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)</code>	Stochastic gradient descent optimizer.
RMSprop	<code>keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)</code>	RMSProp optimizer.
Adagrad	<code>keras.optimizers.Adagrad(lr=0.01, epsilon=None, decay=0.0)</code>	Adagrad optimizer.
Adadelta	<code>keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=None, decay=0.0)</code>	Adadelta optimizer.
Adam	<code>keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)</code>	Adam optimizer.
Adamax	<code>keras.optimizers.Adamax(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0)</code>	Adamax optimizer from Adam paper's Section 7.
Nadam	<code>keras.optimizers.Nadam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, schedule_decay=0.004)</code>	Nesterov Adam optimizer.

# Activation functions

Keras

<https://keras.io/activations/>

이름	함수	설명
<b>softmax</b>	<code>keras.activations.softmax(x, axis=-1)</code>	Softmax activation function.
<b>elu</b>	<code>keras.activations.elu(x, alpha=1.0)</code>	Exponential linear unit.
<b>selu</b>	<code>keras.activations.selu(x)</code>	Scaled Exponential Linear Unit (SELU).
<b>softplus</b>	<code>keras.activations.softplus(x)</code>	Softplus activation function.
<b>softsign</b>	<code>keras.activations.softsign(x)</code>	Softsign activation function.
<b>relu</b>	<code>keras.activations.relu(x, alpha=0.0, max_value=None, threshold=0.0)</code>	Rectified Linear Unit.
<b>tanh</b>	<code>keras.activations.tanh(x)</code>	Hyperbolic tangent.
<b>sigmoid</b>	<code>keras.activations.sigmoid(x)</code>	Sigmoid activation function.
<b>hard_sigmoid</b>	<code>keras.activations.hard_sigmoid(x)</code>	Faster to compute than sigmoid activation.
<b>exponential</b>	<code>keras.activations.exponential(x)</code>	Exponential (base e) activation function.
<b>linear</b>	<code>keras.activations.linear(x)</code>	Linear (i.e. identity) activation function.

# Advanced Activation functions

Keras

<https://keras.io/layers/advanced-activations/>

이름	함수	설명
LeakyReLU	<code>keras.layers.LeakyReLU(alpha=0.3)</code> Rectifier Nonlinearities Improve Neural Network Acoustic Models ( <a href="https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf">https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf</a> )	Leaky version of a Rectified Linear Unit.
PReLU	<code>keras.layers.PReLU(alpha_initializer='zeros', alpha_regularizer=None, alpha_constraint=None, shared_axes=None)</code> Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification( <a href="https://arxiv.org/abs/1502.01852">https://arxiv.org/abs/1502.01852</a> )	Parametric Rectified Linear Unit.
ELU	<code>keras.layers.ELU(alpha=1.0)</code> Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)( <a href="https://arxiv.org/abs/1511.07289v1">https://arxiv.org/abs/1511.07289v1</a> )	Exponential Linear Unit.
ThresholdedReLU	<code>keras.layers.ThresholdedReLU(theta=1.0)</code> Zero-Bias Autoencoders and the Benefits of Co-Adapting Features ( <a href="https://arxiv.org/abs/1402.3337">https://arxiv.org/abs/1402.3337</a> )	Thresholded Rectified Linear Unit.
Softmax	<code>keras.layers.Softmax(axis=-1)</code>	Softmax activation function.
ReLU	<code>keras.layers.ReLU(max_value=None, negative_slope=0.0, threshold=0.0)</code>	Rectified Linear Unit activation function.

# 배치 정규화(Batch Normalization)

Keras

- Gradient Vanishing/Gradient Exploding 이 일어나지 않도록 하는 아이디어 중 하나
- 지금까지는 간접적인 방법을 사용 했음
  - Activation 함수의 변화 (ReLU 등)
  - Careful Weight Initialization(mean 0, std 1, Gaussian Distribution)
  - Small Learning Rate 등으로 해결
- 불안정화가 일어나는 이유가 ‘Internal Covariance Shift’ 라고 주장
  - Internal Covariance Shift라는 현상은 Network의 각 층이나 Activation 마다 input의 distribution이 달라지는 현상
  - Training 하는 과정 자체를 전체적으로 안정화하여 학습 속도를 가속시킬 수 있는 근본적인 방법을 사용
  - ICS 현상을 막기 위해서 간단하게 각 층의 input의 distribution을 평균 0, 표준편차 1인 input으로 normalize 시키는 방법(Keras의 BatchNormalization)
- Batch Normalization은 Whitening이라는 방법으로 해결할 수 있음
  - Whitening은 기본적으로 들어오는 input의 feature들을 uncorrelated 하게 만들어주고, 각각의 variance를 1로 만들어주는 작업
  - Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (<https://arxiv.org/abs/1502.03167>)
- <https://keras.io/layers/normalization/>

# 배치 정규화(Batch Normalization)

Keras

- `keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001, center=True, scale=True, beta_initializer='zeros', gamma_initializer='ones', moving_mean_initializer='zeros', moving_variance_initializer='ones', beta_regularizer=None, gamma_regularizer=None, beta_constraint=None, gamma_constraint=None)`

# loss function

Keras

- 케라스 모델의 컴파일을 위한 파라미터 중 하나
- 손실함수를 정의
- <https://keras.io/losses/>

```
from keras import losses
```

```
model.compile(loss=losses.mean_squared_error, optimizer='sgd')
```

문자열을 이용해서 설정  
가능

y\_true와 y\_pred 파라미터를  
갖는 손실 함수를 이용해 지정

```
model.compile(loss='mean_squared_error', optimizer='sgd')
```

# loss function

Keras

이름	함수
mean_squared_error	<code>keras.losses.mean_squared_error(y_true, y_pred)</code>
mean_absolute_error	<code>keras.losses.mean_absolute_error(y_true, y_pred)</code>
mean_absolute_percentage_error	<code>keras.losses.mean_absolute_percentage_error(y_true, y_pred)</code>
mean_squared_logarithmic_error	<code>keras.losses.mean_squared_logarithmic_error(y_true, y_pred)</code>
squared_hinge	<code>keras.losses.squared_hinge(y_true, y_pred)</code>
hinge	<code>keras.losses.hinge(y_true, y_pred)</code>
categorical_hinge	<code>keras.losses.categorical_hinge(y_true, y_pred)</code>
logcosh	<code>keras.losses.logcosh(y_true, y_pred)</code>
categorical_crossentropy	<code>keras.losses.categorical_crossentropy(y_true, y_pred)</code>
sparse_categorical_crossentropy	<code>keras.losses.sparse_categorical_crossentropy(y_true, y_pred)</code>
binary_crossentropy	<code>keras.losses.binary_crossentropy(y_true, y_pred)</code>
kullback_leibler_divergence	<code>keras.losses.kullback_leibler_divergence(y_true, y_pred)</code>
poisson	<code>keras.losses.poisson(y_true, y_pred)</code>
cosine_proximity	<code>keras.losses.cosine_proximity(y_true, y_pred)</code>



# 모델 저장하기

Keras

```
1. from keras.datasets import mnist
2. from keras.models import Sequential
3. from keras.layers import Flatten, Dense, Dropout

4. (x_train, y_train), (x_test, y_test) = mnist.load_data()
5. x_train, x_test = x_train / 255.0, x_test / 255.0

6. model = Sequential()
7. model.add(Flatten(input_shape=(28, 28)))
8. model.add(Dense(360, activation='relu'))
9. model.add(Dense(128, activation='relu'))
10. model.add(Dropout(0.2))
11. model.add(Dense(10, activation='softmax'))

12. model.compile(optimizer='SGD',
13.                loss='sparse_categorical_crossentropy',
14.                metrics=['accuracy'])

15. model.fit(x_train, y_train, batch_size=100, epochs=5, verbose=1)
16. # model.evaluate(x_test, y_test)

17. model.save('mnist_keras_model.h5')
```

# 모델 불러오기

Keras

```
1. from keras.datasets import mnist
2. from keras.models import load_model
3. mnist = tf.keras.datasets.mnist
4. (x_train, y_train), (x_test, y_test) = mnist.load_data()
5. x_train, x_test = x_train / 255.0, x_test / 255.0

6. model = load_model('mnist_keras_model.h5')

7. print(model.evaluate(x_test, y_test))

8. import numpy as np
9. pred = model.predict(x_test)
10. print(np.argmax(pred, axis=1))
```

# Callback

Keras

- 학습 시 특정 조건이 되면 실행되는 객체
- fit() 함수의 callbacks 파라미터를 통해 학습할 때 지정할
- <https://keras.io/callbacks/>
- 콜백의 종류
  - Callback, BaseLogger, TerminateOnNaN, ProgbarLogger, History
  - ModelCheckpoint, EarlyStopping
  - RemoteMonitor, LearningRateScheduler
  - TensorBoard
  - ReduceLROnPlateau
  - CSVLogger
  - LambdaCallback

# Model Checkpoint Callback

Keras

- 학습을 할 때 모델을 체크포인트 할 콜백
- ModelCheckpoint 콜백을 지정하려면 fit() 함수에 validation\_split 매개변수를 추가해야 함
- `keras.callbacks.ModelCheckpoint(filepath, monitor='val_loss', verbose=0, save_best_only=False, save_weights_only=False, mode='auto', period=1)`
  - filepath: string, path to save the model file.
  - monitor: quantity to monitor.
  - verbose: verbosity mode, 0 or 1.
  - save\_best\_only: if save\_best\_only=True, the latest best model according to the quantity monitored will not be overwritten.
  - save\_weights\_only: if True, then only the model's weights will be saved (model.save\_weights(filepath)), else the full model is saved (model.save(filepath)).
  - mode: one of {auto, min, max}. If save\_best\_only=True, the decision to overwrite the current save file is made based on either the maximization or the minimization of the monitored quantity. For val\_acc, this should be max, for val\_loss this should be min, etc. In auto mode, the direction is automatically inferred from the name of the monitored quantity.
  - period: Interval (number of epochs) between checkpoints.
- <https://keras.io/callbacks/#modelcheckpoint>

# Model Checkpoint Callback

Keras

```
1. import tensorflow as tf
2. from keras.models import Sequential
3. from keras.layers import Flatten, Dense, Dropout
4. from keras.callbacks import ModelCheckpoint
5. mnist = tf.keras.datasets.mnist
6. (x_train, y_train), (x_test, y_test) = mnist.load_data()
7. x_train, x_test = x_train / 255.0, x_test / 255.0

8. MODEL_SAVE_FOLDER = './model/'
9. import os
10. if not os.path.exists(MODEL_SAVE_FOLDER):
11.     os.mkdir(MODEL_SAVE_FOLDER)

12. path = MODEL_SAVE_FOLDER + '{epoch:02d}-{val_acc:.4f}.hdf5'
13. checkpoint = ModelCheckpoint(filepath=path, monitor='val_acc', verbose=1, save_best_only=True)

14. model = Sequential()
15. model.add(Flatten(input_shape=(28, 28)))
16. model.add(Dense(360, activation='relu'))
17. model.add(Dense(128, activation='relu'))
18. model.add(Dropout(0.2))
19. model.add(Dense(10, activation='softmax'))
20. model.compile(optimizer='SGD', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
21. model.fit(x_train, y_train,
            validation_split=0.2, batch_size=1000, epochs=50, verbose=1, callbacks=[checkpoint])
22. # model.evaluate(x_test, y_test)
```

사용 가능한 monitor 매개변수 값  
['loss', 'acc', 'val\_loss', 'val\_acc']

fit() 함수에 validation\_split=0.2 매개변수  
또는 validation\_data=(x\_test, y\_test)  
가 포함되어 있어야 함

accuracy가 증가할 경우  
에만 모델을 저장함

콜백 지정

# Early Stopping Callback

Keras

- 반복문을 돌면서 최고 성능의 모델을 찾아낼 때, 초반에 최고 성능의 모델이 찾아져서 그 보다 더 좋은 성능의 모델이 더 이상 발견되지 않는 경우 학습을 중단시키는 콜백
- `keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=0, verbose=0, mode='auto', baseline=None, restore_best_weights=False)`
  - monitor: quantity to be monitored.
  - min\_delta: minimum change in the monitored quantity to qualify as an improvement, i.e. an absolute change of less than min\_delta, will count as no improvement.
  - patience: number of epochs with no improvement after which training will be stopped.
  - verbose: verbosity mode.
  - mode: one of {auto, min, max}. In min mode, training will stop when the quantity monitored has stopped decreasing; in max mode it will stop when the quantity monitored has stopped increasing; in automode, the direction is automatically inferred from the name of the monitored quantity.
  - baseline: Baseline value for the monitored quantity to reach. Training will stop if the model doesn't show improvement over the baseline.
  - restore\_best\_weights: whether to restore model weights from the epoch with the best value of the monitored quantity. If False, the model weights obtained at the last step of training are used.
- <https://keras.io/callbacks/#earlystopping>

# Early Stopping Callback

Keras

```
1. import tensorflow as tf
2. from keras.models import Sequential
3. from keras.layers import Flatten, Dense, Dropout
4. from keras.callbacks import ModelCheckpoint
5. from keras.callbacks import EarlyStopping

6. ... 생략 ...
7. model_path = MODEL_SAVE_FOLDER + '{epoch:05d}-{val_acc:.4f}.hdf5'
8. checkpoint = ModelCheckpoint(filepath=model_path, monitor='val_acc',
                               verbose=1, save_best_only=True)

9. early_stopping = EarlyStopping(monitor='val_loss', patience=3)

10. model = Sequential()
11. ... 생략 ...
12. model.compile(optimizer='SGD',
                 loss='sparse_categorical_crossentropy',
                 metrics=['accuracy'])

13. model.fit(x_train, y_train,
            validation_split=0.2, batch_size=1000, epochs=5000, verbose=0,
            callbacks=[checkpoint, early_stopping])
```

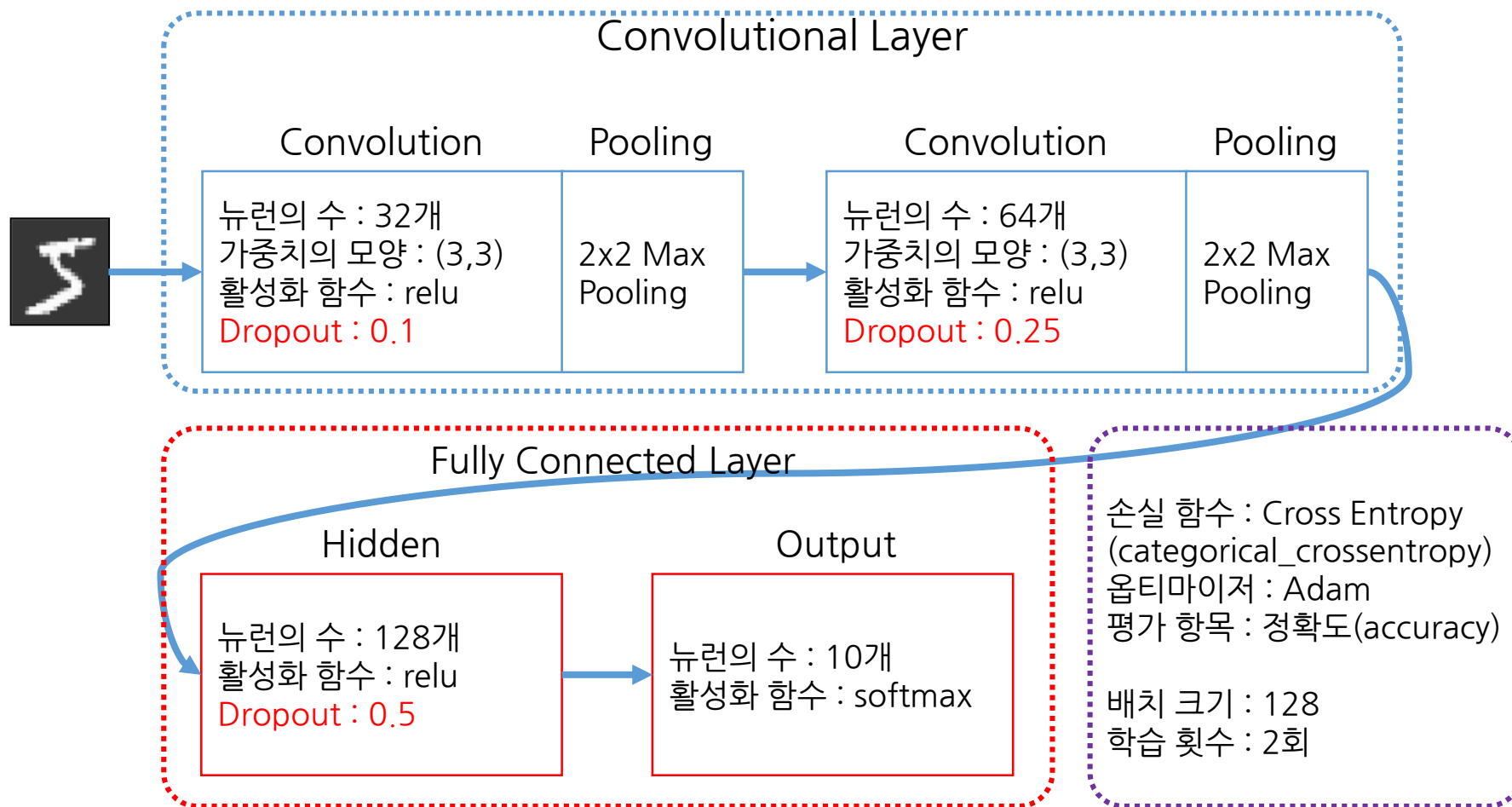
```
Epoch 00119: val_acc improved from 0.94842 to 0.94858, saving model to ./model/00119-0.9486.hdf5
Epoch 00120: val_acc improved from 0.94858 to 0.94883, saving model to ./model/00120-0.9488.hdf5
Epoch 00121: val_acc did not improve from 0.94883
Epoch 00122: val_acc improved from 0.94883 to 0.94950, saving model to ./model/00122-0.9495.hdf5
Epoch 00123: val_acc did not improve from 0.94950
Epoch 00124: val_acc did not improve from 0.94950
Epoch 00125: val_acc did not improve from 0.94950
<keras.callbacks.History at 0x1f78055f908>
```

이 콜백은 accuracy가  
3번 증가하지 않으면  
학습을 중단시킴

총 epoch은 5000번 이지만 123, 123,  
125번째의 accuracy가 더 이상 증가하  
지 않아 학습을 종료함

# CNN 모델 구현하기

Keras





# CNN 모델 구현하기 코드(1/2)

Keras

```
1. import keras
2. from keras.datasets import mnist
3. from keras.models import Sequential
4. from keras.layers import Dense, Dropout, Flatten
5. from keras.layers import Conv2D, MaxPooling2D
6.
7. (x_train, y_train), (x_test, y_test) = mnist.load_data()
8.
9. x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
10. x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
11.
12. x_train = x_train.astype('float32')
13. x_test = x_test.astype('float32')
14. x_train /= 255
15. x_test /= 255
16. print('x_train shape:', x_train.shape)
17. print(x_train.shape[0], 'train samples')
18. print(x_test.shape[0], 'test samples')
19.
20. # convert class vectors to binary class matrices
21. y_train = keras.utils.to_categorical(y_train, 10)
22. y_test = keras.utils.to_categorical(y_test, 10)
23.
```

# CNN 모델 구현하기 코드(2/2)

Keras

```
24. model = Sequential()
25. model.add(Conv2D(32, (3, 3), # kernel_size=(3,3)
                    input_shape=(28, 28, 1), activation='relu',))
26. model.add(MaxPooling2D(pool_size=(2, 2)))
27. model.add(Dropout(0.1))
28.
29. model.add(Conv2D(64, (3, 3), activation='relu'))
30. model.add(MaxPooling2D(pool_size=(2, 2)))
31. model.add(Dropout(0.25))
32.
33. model.add(Flatten())
34. model.add(Dense(128, activation='relu'))
35. model.add(Dropout(0.5))
36. model.add(Dense(10, activation='softmax'))
37.
38. model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])
39.
40. hist = model.fit(x_train, y_train, validation_data=(x_test, y_test),
                    batch_size=128, epochs=2, verbose=1)
41.
42. score = model.evaluate(x_test, y_test, verbose=0)
43. print('Test loss:', score[0])
44. print('Test accuracy:', score[1])
```

# 모델 학습 과정 표시하기

```
1. %matplotlib inline
2. import matplotlib.pyplot as plt

3. fig, loss_ax = plt.subplots()

4. acc_ax = loss_ax.twinx()

5. loss_ax.plot(hist.history['loss'], 'y', label='train loss')
6. loss_ax.plot(hist.history['val_loss'], 'r', label='val loss')

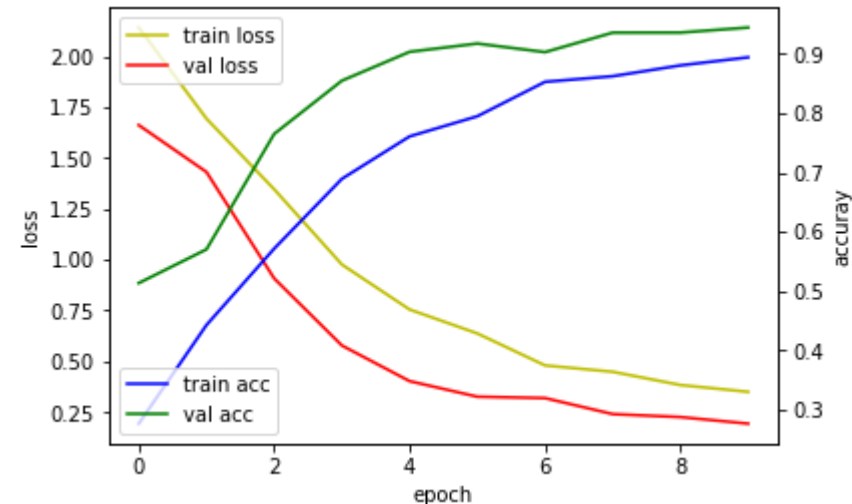
7. acc_ax.plot(hist.history['acc'], 'b', label='train acc')
8. acc_ax.plot(hist.history['val_acc'], 'g', label='val acc')

9. loss_ax.set_xlabel('epoch')
10. loss_ax.set_ylabel('loss')
11. acc_ax.set_ylabel('accuracy')

12. loss_ax.legend(loc='upper left')
13. acc_ax.legend(loc='lower left')

14. plt.show()
```

batch\_size=10000, epoch=10





---

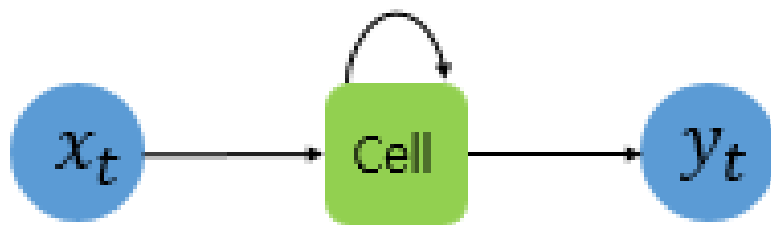
## 2절. 순환 신경망(RNN)



## 2.1. 순환 신경망(RNN)

2절. 순환 신경망(RNN)

- RNN(Recurrent Neural Network)은 시퀀스(Sequence) 모델임
  - 입력과 출력을 시퀀스로 처리하는 모델
  - 번역기를 생각해보면 입력은 번역하고자 하는 문장. 즉, 시퀀스입니다. 출력에 해당하는 번역된 문장 또한 시퀀스입니다.
  - 시퀀스 모델들을 처리하기 위해 고안된 모델들을 시퀀스 모델이라고 함
  - RNN은 시퀀스 모델의 가장 대표적이고 기본적인 모델
- RNN은 은닉층에서 활성화 함수를 통해 나온 결과값을 출력층 방향으로 보내면서, 다시 은닉층의 다음 계산의 입력으로 보내는 특징을 갖고 있음

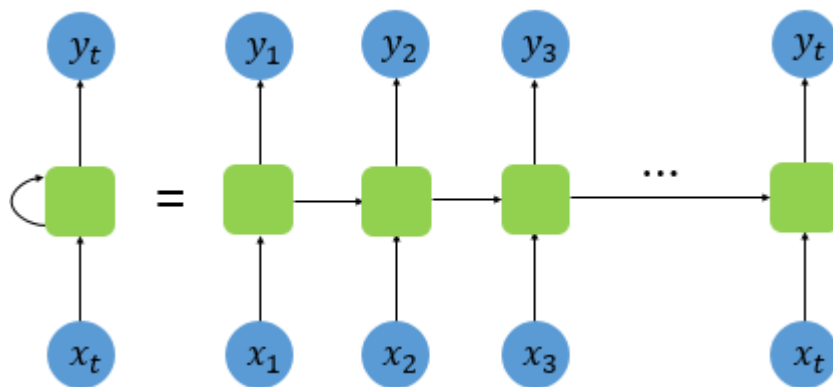


편향  $b$ 도 은닉층과 출력층의 입력으로 존재하지만 앞으로의 신경망의 그림에서 편향  $b$ 는 생략함

# 은닉 상태

## 2절. 순환 신경망(RNN)

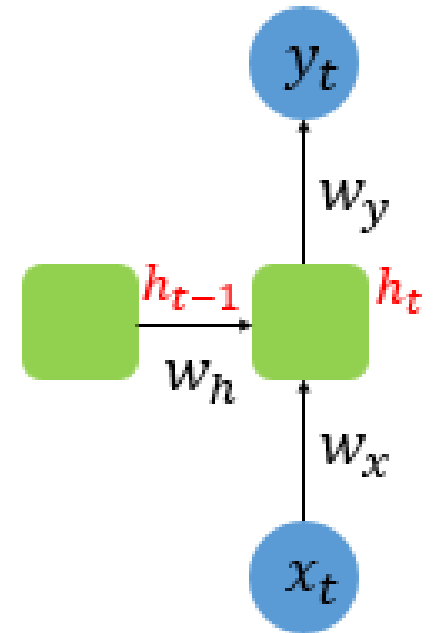
- 은닉층의 메모리 셀은 각각의 시점(time-step)에서 바로 이전 시점에서의 은닉층의 메모리 셀에서 나온 값들을 계속해서 자신의 입력으로 보내는 재귀적 활동을 하고 있음
  - 현재 시점을  $t$ 로 표현하고, 이전 시점을  $t-1$ , 다음 시점을  $t+1$ 와 같은 형식으로 표현함
  - 이는 현재 시점  $t$ 에서의 메모리 셀이 갖고있는 값은 과거의 메모리 셀들의 값에 영향을 받은 것임을 의미함
- 메모리 셀이 다음 시점  $t+1$ 에 다시 자신에게 보내는 이 값을 은닉 상태(hidden state)라고 함
  - 다시 말해 현재 시점  $t$ 의 메모리 셀은 이전 시점  $t-1$ 에서의 메모리 셀이 보낸 은닉 상태값을 다시 계산을 위한 입력값으로 사용



# RNN의 은닉층, 출력층에 대한 수식

2절. 순환 신경망(RNN)

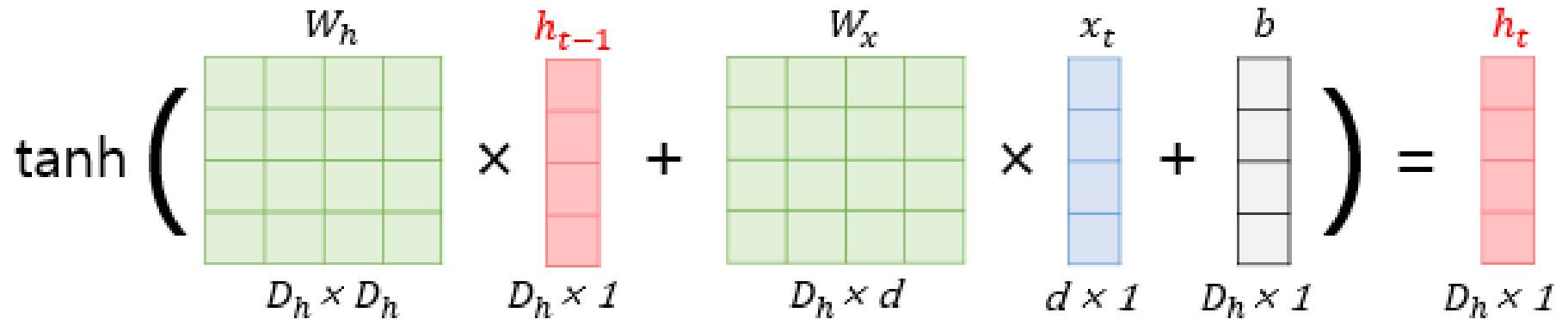
- 현재 시점  $t$ 에서의 은닉 상태값을  $h_t$ 라고 정의
- 은닉층의 메모리 셀은  $h_t$ 를 계산하기 위해서 총 두 개의 가중치를 갖게 됨
  - 입력층에서 입력값을 위한 가중치  $W_x$ 이고,
  - 이전 시점  $t-1$ 의 은닉 상태값인  $h_{t-1}$ 을 위한 가중치  $W_h$
- 식
  - 은닉층 :  $h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$
  - 출력층 :  $y_t = f(W_y h_t + b)$   
단,  $f$ 는 비선형 활성화 함수 중 하나.



# RNN의 은닉층, 출력층에 대한 수식

2절. 순환 신경망(RNN)

- 배치 크기가 1이고,  $d$ 와  $D_h$  두 값을 모두 4로 가정하였을 때, RNN의 은닉층 연산을 그림으로 표현



The diagram illustrates the hidden layer calculation of an RNN. It shows the following components and their dimensions:

- $W_h$ : A green 4x4 matrix with dimension  $D_h \times D_h$ .
- $h_{t-1}$ : A red 4x1 vector with dimension  $D_h \times 1$ .
- $W_x$ : A green 4x4 matrix with dimension  $D_h \times d$ .
- $x_t$ : A blue 4x1 vector with dimension  $d \times 1$ .
- $b$ : A gray 4x1 vector with dimension  $D_h \times 1$ .
- $h_t$ : A red 4x1 vector with dimension  $D_h \times 1$ .

The calculation is represented by the equation:

$$\tanh \left( W_h \times h_{t-1} + W_x \times x_t + b \right) = h_t$$



## 2.2. 양방향 순환 신경망(Bidirectional Recurrent Neural Network)

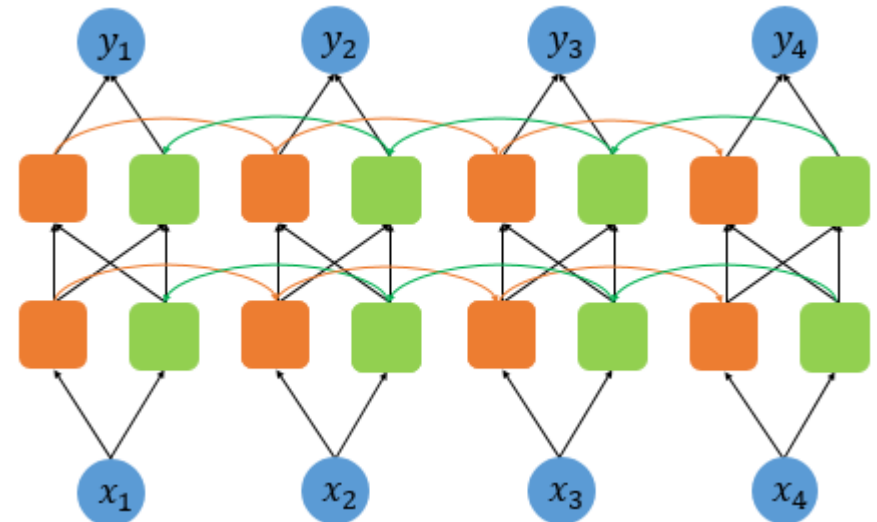
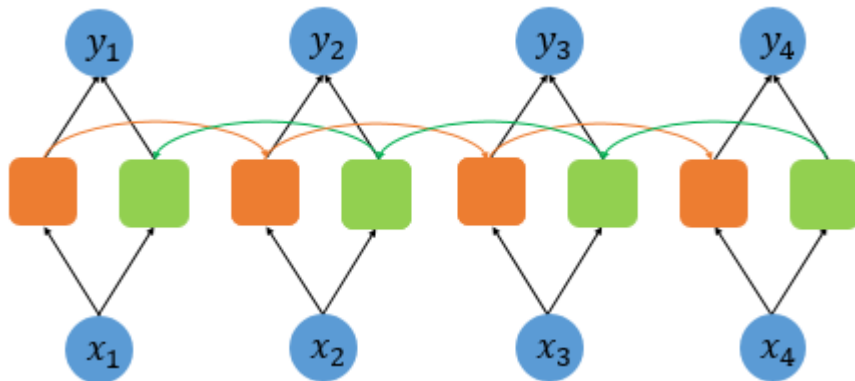
2절. 순환 신경망(RNN)

- 시점  $t$ 에서의 출력값을 예측할 때 이전 시점의 데이터뿐만 아니라, 이후 데이터로도 예측할 수 있다는 아이디어에 기반

- 영어 빈칸 채우기 문제

Q) Exercise is very effective at [            ] belly fat.

- 1) reducing
- 2) increasing
- 3) multiplying



# keras.layers.Embedding

- Embedding()을 통해 생성하는 임베딩 층(embedding layer) 또한 인공 신경망의 층의 하나이므로 model.add()로 추가해야 함
- `keras.layers.Embedding(input_dim, output_dim, embeddings_initializer='uniform', embeddings_regularizer=None, activity_regularizer=None, embeddings_constraint=None, mask_zero=False, input_length=None)`
- 매개변수
  - **input\_dim**: int > 0. Size of the vocabulary, i.e. maximum integer index + 1.
  - **output\_dim**: int >= 0. Dimension of the dense embedding.
  - **embeddings\_initializer**: Initializer for the embeddings matrix.
  - **embeddings\_regularizer**: Regularizer function applied to the embeddings matrix.
  - **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation").

## 2.3. 문맥을 예측해서 다음 단어 예측해보기

2절. 순환 신경망(RNN)

```
1 text="""경마장에 있는 말이 뛰고 있다\n
2 그의 말이 법이다\n
3 가는 말이 고와야 오는 말이 곱다\n"""
```

```
1 from keras_preprocessing.text import Tokenizer
2 t = Tokenizer()
3 t.fit_on_texts([text])
4 encoded = t.texts_to_sequences([text])[0]
```

토큰화와 정수 인코딩

```
1 vocab_size = len(t.word_index) + 1
2 # 케라스 토큰라이저의 정수 인코딩은 인덱스가 1부터 시작하지만,
3 # 케라스 원-핫 인코딩에서 배열의 인덱스가 0부터 시작하기 때문에
4 # 배열의 크기를 실제 단어 집합의 크기보다 +1로 생성해야하므로 미리 +1 선언
5 print('단어 집합의 크기 : %d' % vocab_size)
```

단어 집합의 크기 : 12

```
1 print(t.word_index)
```

{'말이': 1, '경마장에': 2, '있는': 3, '뛰고': 4, '있다': 5, '그의': 6, '법이다': 7, '가는': 8, '고와야': 9, '오는': 10, '곱다': 11}

## 2.3. 문맥을 예측해서 다음 단어 예측해보기

2절. 순환 신경망(RNN)

```
1 print(t.word_index)
```

```
{'말이': 1, '경마장에': 2, '있는': 3, '뛰고': 4, '있다': 5, '그의': 6, '법이다': 7, '가는': 8, '고와야': 9, '오는': 10, '곱다': 11}
```

```
1 sequences = list()
2 for line in text.split('\n'): # \n을 기준으로 문장 토큰화
3     encoded = t.texts_to_sequences([line])[0]
4     for i in range(1, len(encoded)):
5         sequence = encoded[:i+1]
6         sequences.append(sequence)
7
8 print('훈련 데이터의 개수: %d' % len(sequences))
```

훈련 데이터의 개수: 11

```
1 print(sequences)
```

```
[[2, 3], [2, 3, 1], [2, 3, 1, 4], [2, 3, 1, 4, 5], [6, 1], [6, 1, 7], [8, 1], [8, 1, 9], [8, 1, 9, 10], [8, 1, 9, 10, 1], [8, 1, 9, 10, 1, 11]]
```

```
1 print(max(len(l) for l in sequences))
```

[2, 3]은 [경마장에, 있는]에 해당되며 [2, 3, 1]은 [경마장에, 있는, 말이]에 해당됨  
모든 훈련 데이터에 대해서 맨 우측에 있는 단어에 대해서만 y로 분리하면 X와 y의 쌍(pair)이 됨

## 2.3. 문맥을 예측해서 다음 단어 예측해 보기

2절. 순환 신경망(RNN)

```
1 from keras.preprocessing.sequence import pad_sequences
2 sequences = pad_sequences(sequences, maxlen=6, padding='pre')
```

Using TensorFlow backend.

```
1 print(sequences)
```

```
[[ 0  0  0  0  2  3]
 [ 0  0  0  2  3  1]
 [ 0  0  2  3  1  4]
 [ 0  2  3  1  4  5]
 [ 0  0  0  0  6  1]
 [ 0  0  0  6  1  7]
 [ 0  0  0  0  8  1]
 [ 0  0  0  8  1  9]
 [ 0  0  8  1  9 10]
 [ 0  8  1  9 10  1]
 [ 8  1  9 10  1 11]]
```

- pad\_sequences()는 모든 데이터에 대해서 0을 추가하여 길이를 맞춰줌
- maxlen의 값으로 6을 주면 모든 데이터의 길이를 6으로 맞춰주며, padding의 인자로 'pre'를 주면 길이가 6보다 짧은 데이터의 앞을 0으로 채움

## 2.3. 문맥을 예측해서 다음 단어 예측해보기

2절. 순환 신경망(RNN)

```
1 import numpy as np
2 sequences = np.array(sequences)
3 X = sequences[:, :-1]
4 y = sequences[:, -1]
5 # 리스트의 마지막 열을 제외하고 저장한 것은 X
6 # 리스트의 마지막 열만 저장한 것은 y
```

```
1 print(X)
```

```
[[ 0  0  0  0  2]
 [ 0  0  0  2  3]
 [ 0  0  2  3  1]
 [ 0  2  3  1  4]
 [ 0  0  0  0  6]
 [ 0  0  0  6  1]
 [ 0  0  0  0  8]
 [ 0  0  0  8  1]
 [ 0  0  8  1  9]
 [ 0  8  1  9 10]
 [ 8  1  9 10 11]]
```

X와 y의 분리

## 2.3. 문맥을 예측해서 다음 단어 예측해보기

2절. 순환 신경망(RNN)

```
1 print(y)
```

```
[ 3  1  4  5  1  7  1  9 10  1 11]
```

```
1 from keras.utils import to_categorical
2 y = to_categorical(y, num_classes=vocab_size)
```

```
1 print(y)
```

```
[[0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]]
```

훈련 데이터를 훈련  
시키기 전에 y에 대해서  
원-핫 인코딩을 수행

## 2.3. 문맥을 예측해서 다음 단어 예측해보기

2절. 순환 신경망(RNN)

```
1 from keras.layers import Embedding, Dense, SimpleRNN
2 from keras.models import Sequential
3
4 model = Sequential()
5 # y를 제거하였으므로 이제 x의 길이는 5
6 model.add(Embedding(vocab_size, 10, input_length=5))
7 model.add(SimpleRNN(32))
8 model.add(Dense(vocab_size, activation='softmax'))
9 model.compile(loss='categorical_crossentropy', optimizer='adam',
10               metrics=['accuracy'])
11 model.fit(X, y, epochs=200, verbose=2)
```

Epoch 180/200

- 0s - loss: 0.1980 - acc: 1.0000

Epoch 181/200

- 0s - loss: 0.1948 - acc: 1.0000

Epoch 182/200

- 0s - loss: 0.1917 - acc: 1.0000

Epoch 183/200

- 0s - loss: 0.1886 - acc: 1.0000

RNN 모델에 데이터를  
훈련시킴

중간부터 정확도가  
100%가 나옴



## 2.3. 문맥을 예측해서 다음 단어 예측해보기

2절. 순환 신경망(RNN)

```
1 def sentence_generation(model, t, current_word, n): # 모델, 토큰라이저, 현재 단어, 반복할 횟수
2     init_word = current_word # 처음 들어온 단어도 마지막에 같이 출력하기 위해 저장
3     sentence = ''
4     for _ in range(n): # n번 반복
5         encoded = t.texts_to_sequences([current_word])[0] # 현재 단어에 대한 정수 인코딩
6         encoded = pad_sequences([encoded], maxlen=5, padding='pre') # 데이터에 대한 패딩
7         result = model.predict_classes(encoded, verbose=0)
8         # 입력한 X(현재 단어)에 대해서 Y를 예측하고 Y(예측한 단어)를 result에 저장.
9         for word, index in t.word_index.items():
10             if index == result: # 만약 예측한 단어와 인덱스와 동일한 단어가 있다면
11                 break # 해당 단어가 예측 단어이므로 break
12             current_word = current_word + ' ' + word # 현재 단어 + ' ' + 예측 단어를 현재 단어로 변경
13             sentence = sentence + ' ' + word # 예측 단어를 문장에 저장
14         # for문이므로 이 행동을 다시 반복
15     sentence = init_word + sentence
16     return sentence
```

```
1 print(sentence_generation(model, t, '경마장에', 4))
2 # '경마장에' 라는 단어 뒤에는 총 4개의 단어가 있으므로 4번 예측
```

경마장에 있는 말이 뛰고 있다

모델이 정확하게 예측하고  
있는지 문장을 생성하는  
함수를 만들어서 실제로 출력

```
1 print(sentence_generation(model, t, '그의', 2)) # 2번 예측
```

그의 말이 법이다



---

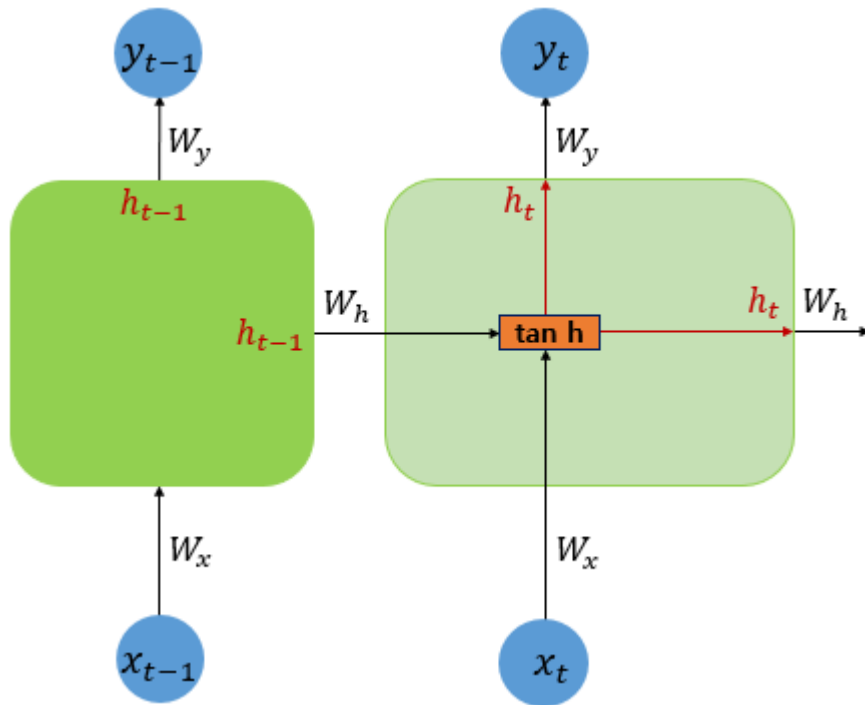
## 3절. 장단기 메모리(LSTM)



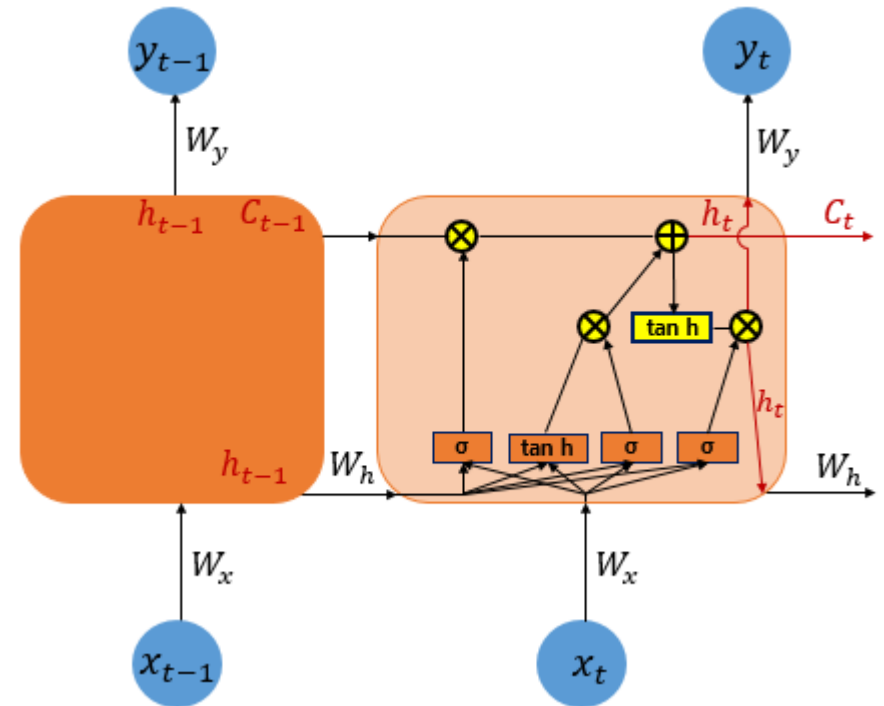
# RNN vs. LSTM

3절. 장단기 메모리(LSTM)

Vanilla RNN



LSTM(Long Short-Term Memory)



# keras.layers.LSTM

- `keras.layers.LSTM(units, activation='tanh', recurrent_activation='sigmoid', use_bias=True, kernel_initializer='glorot_uniform', recurrent_initializer='orthogonal', bias_initializer='zeros', unit_forget_bias=True, kernel_regularizer=None, recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, recurrent_constraint=None, bias_constraint=None, dropout=0.0, recurrent_dropout=0.0, implementation=2, return_sequences=False, return_state=False, go_backwards=False, stateful=False, unroll=False)`

<https://keras.io/layers/recurrent/#lstm>

# LSTM

3절. 장단기 메모리(LSTM)

- 기존 코드와 크게 다른 점이 없고 모형을 만드는 클래스가 바뀌었음

```
1 from keras.layers import Embedding, Dense, LSTM
2 from keras.models import Sequential
3
4 model = Sequential()
5 model.add(Embedding(vocab_size, 10, input_length=max_len-1))
6 # y를 제거하였으므로 이제 X의 길이는 기존 데이터의 길이 - 1
7 model.add(LSTM(128))
8 model.add(Dense(vocab_size, activation='softmax'))
9 model.compile(loss='categorical_crossentropy', optimizer='adam',
10               metrics=['accuracy'])
11 model.fit(X, y, epochs=200, verbose=2)
```



---

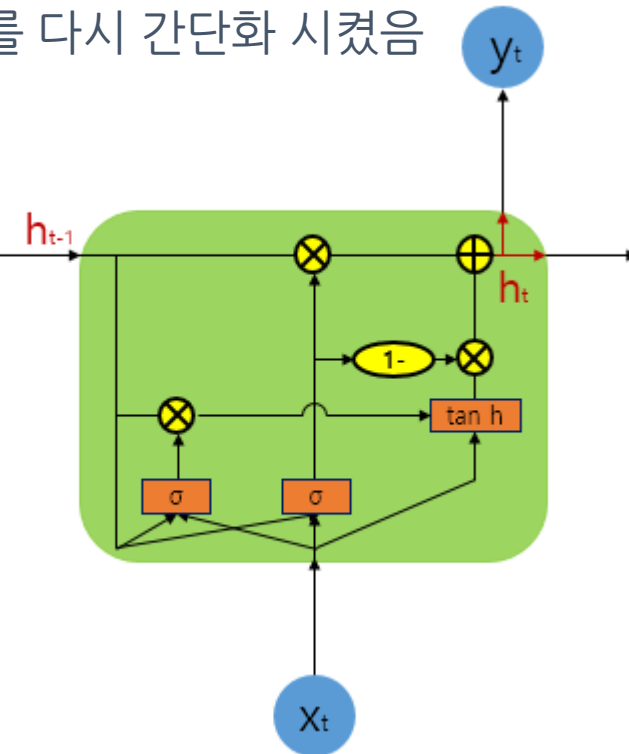


## 4절. 게이트 순환 유닛(GRU)

## 4절. 게이트 순환 유닛(GRU)

4절. 게이트 순환 유닛(GRU)

- GRU(Gated Recurrent Unit)는 2014년 뉴욕대학교 조경현 교수님이 집필한 논문에서 제안
- GRU는 LSTM의 장기 의존성 문제에 대한 해결책을 유지하면서, 은닉 상태를 업데이트하는 계산을 줄였음
  - GRU는 성능은 LSTM과 유사하면서 복잡했던 LSTM의 구조를 다시 간단화 시켰음
- 주의
  - 반드시 LSTM 대신 GRU를 사용하는 것이 좋지는 않음
  - GRU와 LSTM 중 어떤 것이 모델의 성능면에서 더 좋은지 단정지어 말할 수 없으며, 기존에 LSTM을 사용하면서 최적의 하이퍼파라미터를 찾아낸 상황이라면 굳이 GRU로 바뀌서 사용할 필요는 없음
  - 사실 아직까지는 GRU보다 LSTM이 좀 더 많이 사용되는 편이고, 또한 일부 논문에서는 여전히 LSTM이 GRU보다 더 좋은 성능을 보였다고 주장함



# keras.layers.GRU

- `keras.layers.GRU(units, activation='tanh', recurrent_activation='sigmoid', use_bias=True, kernel_initializer='glorot_uniform', recurrent_initializer='orthogonal', bias_initializer='zeros', kernel_regularizer=None, recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, recurrent_constraint=None, bias_constraint=None, dropout=0.0, recurrent_dropout=0.0, implementation=2, return_sequences=False, return_state=False, go_backwards=False, stateful=False, unroll=False, reset_after=False)`

<https://keras.io/layers/recurrent/#gru>

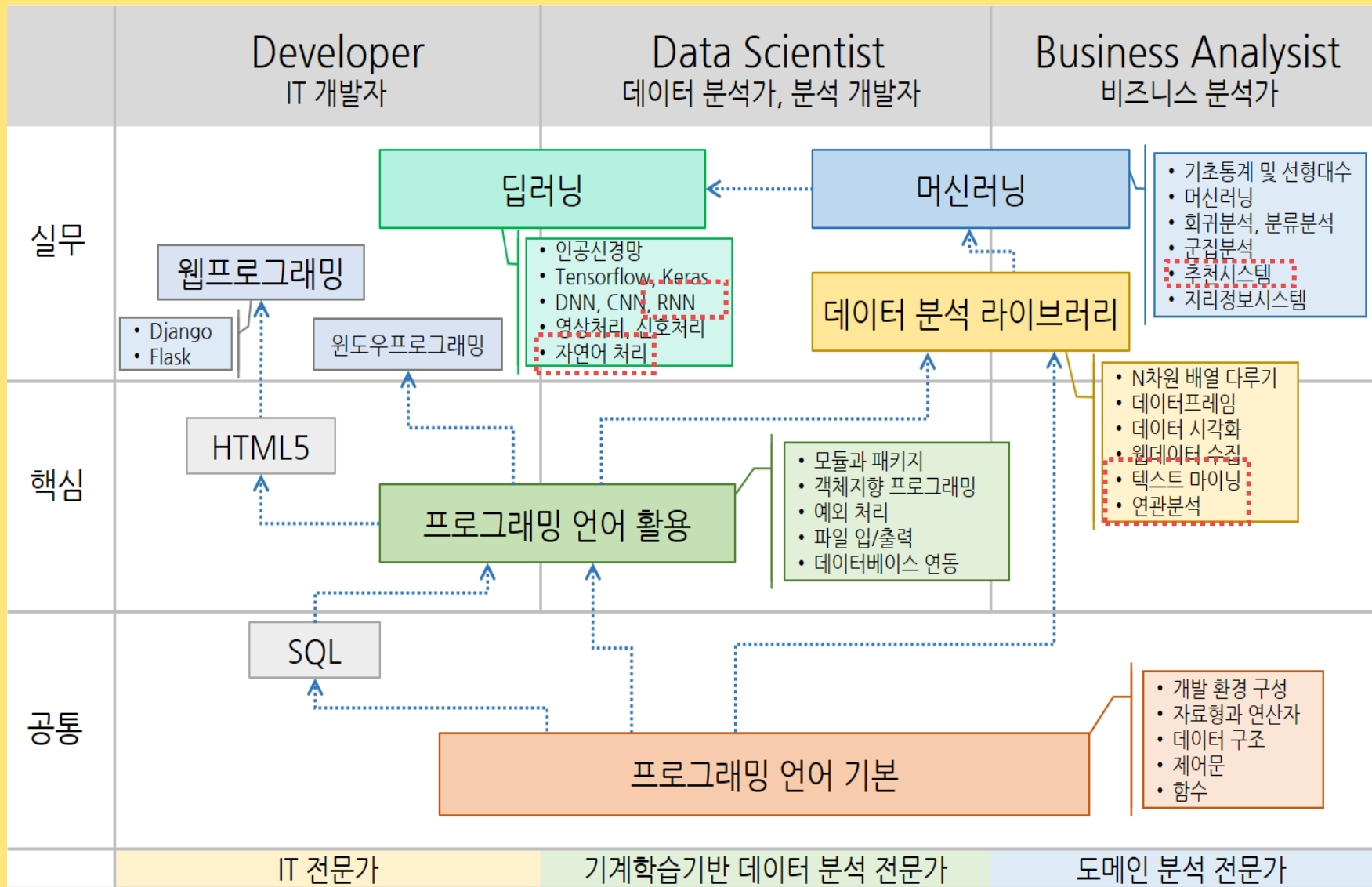


# 케라스에서의 GRU(Gated Recurrent Unit)

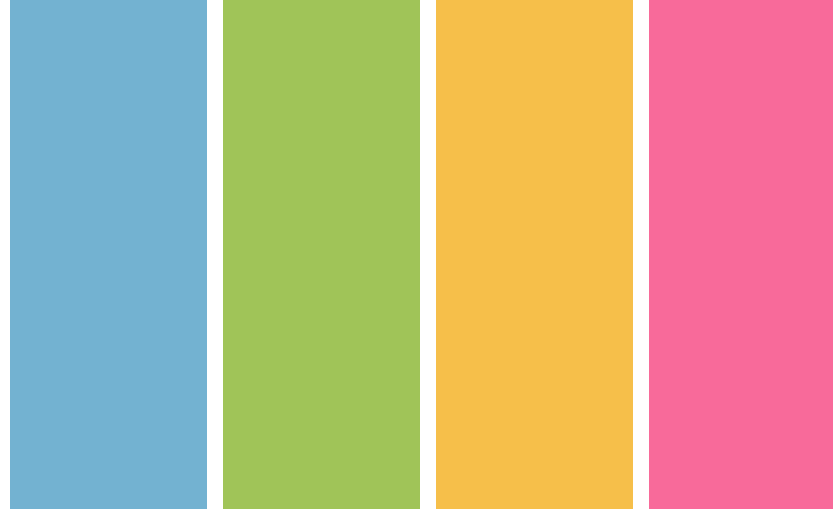
4절. 게이트 순환 유닛(GRU)

- 사용 방법은 SimpleRNN이나 LSTM과 같으며 단지, 이들의 이름 대신 GRU로 바꿔주기만 하면 됨
- 예
  - `model.add(GRU(hidden_size, input_shape=(timesteps, input_dim)))`

# 파이썬 학습 로드맵



hjk7902@gmail.com



**hjk7902@gmail.com**

---

언어인지 알고리즘

# 시험 - RNN 모델 구현하기

Keras

